QATAR UNIVERSITY

COLLEGE OF ENGINEERING

PEDESTRIAN DETECTION USING MOTION SALIENCY AIDED CONVOLTIONAL NEURAL NETWORK

BY

ALI ABDUL HAFIZ FARHAT

A Thesis Submitted to

the Faculty of the College of

Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Electrical Engineering

June 2018

COMMITTEE PAGE

The members of the Committee approve the Thesis of Ali Abdul Hafiz Farhat defended on 08/05/2018.

[illegible]

Dr. Faycal Bensaali
Thesis/Dissertation Supervisor

Prof. Abbes Amira
Thesis/Dissertation Co-Supervisor

Prof. Abdenour Hadid
Committee Member

Prof. Serkan Kiranyaz
Committee Member

Approved:

Khalifa Al-Khalifa, Dean, College of Engineering

ABSTRACT

FARHAT, ALI., Masters : June : 2018, Masters of Science in Electrical Engineering

Title: Pedestrian Detection Using Motion Saliency Aided Convoltional Neural Network

Supervisor of Thesis: Dr. Faycal Bensaali.

Co-Supervisor of Thesis: Prof. Abbes Amira.

Pedestrian and crowd analysis is one of the oldest problems in the area of computer vision and image processing. Researchers have proposed several algorithms that analyze crowds for different purposes aiming to improve their security and safety. The recent advancements in the areas of machine learning and computer devices resulted in a revolution in the proposed algorithms for such problem. The use of Convolutional Neural Networks in the proposed algorithms boosted the performance of the state-of-the-art algorithms. This research project studies the impact of integrating of motion saliency along with the CNN based pedestrian detector. A detailed literature review was conducted to draw the pathway of the project, discussing different approaches of motion flow algorithms and CNN based detectors. Background subtraction based on Gaussian Mixture Model motion flow algorithm was used to extract the motion saliency information. Faster R-CNN based on AlexNet was considered and integrated with the motion flow algorithm. Two different approaches of integration were proposed and tested: (1) Motion Masked Faster R-CNN method, where the motion information is used to generate a mask for the scene, keeping the regions of interest where motion is found and people may be available; (2)Motion Corrected Faster R-CNN method, where the motion information is utilized after the Faster R-CNN detector locates all pedestrian in the frame to correct the false detections. The report discusses the hardware and software setups used in this thesis project to develop and implement the proposed detectors. PETS2009 dataset was used to test the proposed detectors, which are trained on Caltech pedestrian dataset. It was found that the integration of motion information improves the precision of the Faster R-CNN detector by 7% - 30%, where the error rate dropped by 6% - 34%. A detailed discussion of further improvement pathways is provided in the report. Limitations of current implementation of the proposed detectors are discussed. The evaluation of the proposed detectors is compared to state-of-the-art detectors in the literature. Models’ performance was discussed, noting the challenges that degraded their performance and how to overcome them.

DEDICATION

To my grandparents, parents, brothers and sisters

ACKNOWLEDGMENTS

First, I would like to thank my advisors Dr. Faycal Bensaali, and Prof. Abbes Amira for their support, guidance and encouragement that helped me develop and polish my technical and non-technical skills. It was an honor to be part of their research group.

Second, my sincere thanks go to Ilyas Farhat, Mohamed Sami Zitouni, and Omar Hommos for their lovely online meetings and calls that demonstrated their continuous passionate support, technical and non-technical guidance.

I would like to thank the KINDI Center and the Electrical Engineering Department for their continuous support. I would like to thank the KINDI Center for providing me with some of the equipment used in this work and their opportunity where I was part of their lovely teem. I would like to thank Dr. Gabriele Oligeri and Dr. Abdullatif Shikfa for their nice and nurturing collaborative experience that helped me widen my research from various perspectives. Also, I would like to the thank the Electrical Engineering Department for allowing me to work with them and their support throughout the past two years.

Special thanks go to the rest of my colleague and lab mate Sharief Saleh for the enjoyable period that I spent with him during my studies and work at Qatar University. Also I would like to thank my friends Abdulhadi Al-Qahtani, Ali Al-Zawqari, Omar Abu Selo, and Ahmed Musleh for their continuous support and for being always there for help.

Lastly, I would like to say that no matter how much I thank my parents, Abdul Hafiz and Subhiah Farhat, and my siblings, Salman, Hajar, Haroun, Omran, Aalaa, and Ilyas, it would not be sufficient. Their patience, compassion and advices, where they were mostly unfollowed, and that what urges me to better myself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACF	Aggregated Channel Features
AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
AP	Average Precision
BCN	Binary Classification Network
CFN	Channel Feature Network
CNN	Convolutional Neural Networks
ConvNets	Convolutional Neural Networks
CPU	Central Processing Unit
DT-GMM	Dynamic Texture based Gaussian Mixture Model
eGPU	external Graphical Processing Unit
FC	Fully Connected
FCN	Fully Convolutional Network
FPGA	Field Programable Gate Arrays
FPPI	False Positive Per Image
FPPW	False Positive Per Window
FPR	False Positive Rate
FPS	Frames Per Second
GMM	Gaussian Mixture Model
GPU	Graphical Processing Units
HOG	Histogram of Oriented Gradients
ILSVRC	IMAGENET Large Scale Visual Recognition Challenge
IoU	Intersection over Union
LBP	Local Binary Patterns
LDCF	Locally Decorrelated Channel Filters
mAP	mean Average Precision
MC Faster R-CNN	Motion Corrected Faster R-CNN
MM Faster R-CNN	Motion Masked Faster R-CNN

MR	Miss Rate
MRE	Mean Relative Error
MS COCO	Microsoft Common Objects in COntext
MSE	Mean Squared Error
PASCAL VOC	PASCAL Visual Object Challenge
PCIe 3.0	Peripheral Component Interconnect Express Generation 3.0
R-CNN	Regional-based Convolutional Neural Network
RJMCMC	Reversible Jump Markov Chain Monte Carlo
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RoI	Regions of Interest
RPN	Region Proposal Network
RPN-BF	Regional Proposal Network – Boosted Forest
SDS-R-CNN	Simultaneous Detection and Segmentation R-CNN
SENet	Squeeze-and-Excitation Networks
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
TPR	True Positive Rate

CHAPTER 1: INTRODUCTION

The field of image processing and computer vision has been drastically evolving in the past decade, especially after the significant progress achieved in the area of machine learning and Artificial Intelligence (AI). Besides, the development in the fields of electronics and computer devices played an essential role in the development of computer vision algorithms. The latest Graphical Processing Units (GPUs) provided the ability to propose, design, train, and test very complicated algorithms as the bottleneck of implementing computationally intensive algorithms has been addressed [1]. Therefore, performing the required calculations for such algorithms in a reasonable amount of time became viable.

Recently, the implementation of Convolutional Neural Networks (CNN); also known as ConvNets, was the game changer in AI and computer vision. The use of CNN improved the performance of the proposed computer vision algorithms as it provided the ability to perform more complex image analysis and recognition tasks. This was clearly demonstrated in the IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) that was introduced in 2010 instead of the PASCAL Visual Object Challenge (PASCAL VOC) [2, 3]. The image classification challenge of the ILSVRC has an approximately 1.5 million images of 1000 object classes instead of having 20 object classes and 1000 images in the PASCAL VOC [2]. Figure 1.1 summarizes the recognition rate improvements from 2010 to 2017 for the winners of the classification challenge [2, 4-11].

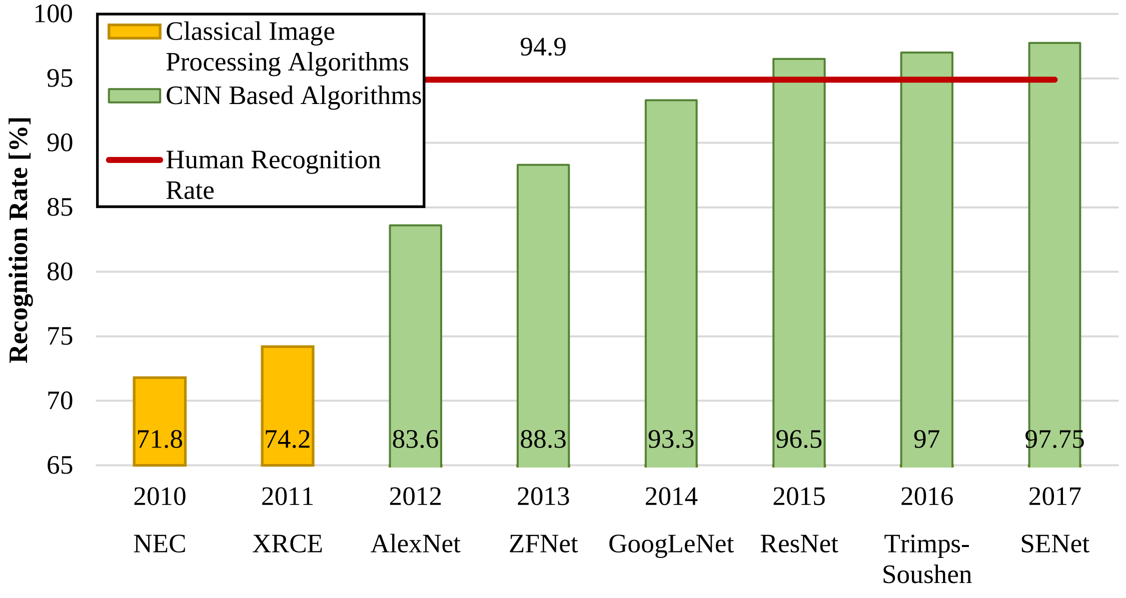


Figure 1.1. The winners' recognition rate of the ILSVRC classification challenge from 2010 to 2017

The algorithm that won the competition of the ILSVRC in 2010 is based on a Scale-Invariant Feature Transform (SIFT) feature extractor and a stochastic Support Vector Machine (SVM) classifier achieving a 71.8% recognition rate [2, 4]. In 2011, the winner utilized a high dimensional image signatures and a one-vs-all linear SVM. This increased the recognition rate to reach 74.2% [2, 5]. However, the winning algorithm architecture of the 2012 ILSVRC, known as AlexNet, is an 8-layer CNN that boosted the recognition rate to 83.6% [2, 6]. AlexNet consists of 60 million parameters and was trained using an efficient GPU implementation. ZFNet, the winner in 2013, is the average of several CNNs. This algorithm improved the recognition rate to settle at 88.3% [2, 7]. Indeed, the increase of the recognition rate did not stop at this point with the proposal of the 22-layer

GoogLeNet in 2014 [2, 8]. The increase of the network depth increased the recognition rate further to reach 93.3%. Nonetheless, the result achieved by the 152-layer ResNet proposed in 2015 by Microsoft Asia research team overcame the human eye ability in classification by reaching 96.5% where the human's recognition rate is 94.9% [9]. Later in 2016, Trimps-Soushen CNN improved the results to reach 97% through fusing five different networks [10]. Furthermore, the winner in 2017 introduced the Squeeze-and-Excitation Networks (SENet) where the performance has been improved further to 97.75% [11].

Therefore, it is very easy to observe the significant improvement in the performance of computer vision and image processing classification algorithms after utilizing the power of CNNs. Yet, computer vision and image processing algorithms are not limited to classification purposes only. There are several applications that require different algorithms such as object detection and instance segmentation. The proposed algorithms in these applications started to adopt CNN based algorithms since they have demonstrated better performance in classification.

1.1 *Background*

Visual analysis and recognition of crowd's and pedestrian's image sequences is of importance for several applications including surveillance, homeland security, intelligent environments, and autonomous driving [12]. The main motivation of such work is related to security and safety of both crowds and pedestrians through preventing any disasters and tragic accidents such as the bombing at the Boston Marathon and the Hajj stampede [13]. Subsequently, this has been the main focal point of many researchers in the field of computer vision and image processing where they have been working on such problems for more than ten years [12, 14]. In 2022, Qatar will host the FIFA World Cup, and it is very important to have a reliable crowd analysis system that emphasizes on the safety of the crowds and prevents any losses in lives.

The applications of pedestrian and crowd analysis are many and include target tracking, counting, anomaly detection, behavioral pattern recognition and crowd modelling [12, 13]. The literature shows significant progress in the autonomous crowd and pedestrian analysis. Yet, several unaddressed shortcomings in the proposed work limited its performance such as occlusions, articulation of the human body, variation in viewpoints, and modelling and inference [12, 15].

According to [12], crowd analysis consists of three main stages: crowd modelling, crowd monitoring, and crowd management. Crowd modelling is about constructing robust crowd representations to understand scenes where crowd monitoring analyzes visual data statistically to design real-time decision systems. Moreover, crowd management's main objective is to ensure the safety of the crowd through efficient analysis of the crowds' strategic, tactical, and operational data. Therefore, crowd modelling is the essential stage in any robust crowd analysis system since it is the primary stage that every advanced analysis is based on.

There are several parameters are used to classify crowd modelling techniques into distinct categories [12]. One of the used parameters is the level where crowd modelling techniques are applied and it consists of two categories: 1) micro-level and 2) macro-level. In the micro-level, crowd modelling techniques concentrate on crowd's individual entities to construct an understanding of the scene's semantics. In contrast, the macro-level techniques are based on the holistic view of the scene where it may use the flow to understand its dynamics. Since the micro-level focus on the individual entities of the crowd, it is considered to be more complex. The micro-level techniques might suffer from occlusions, targets' appearance similarities and more computational complexities. Therefore, more studies in the literature focus on the macro-level techniques. Recently, some groups started to combine both levels to model the crowds and understand the behavior of its entities at the same time [12]. Figure 1.2 shows the schematic diagram of crowd analysis and its categories.

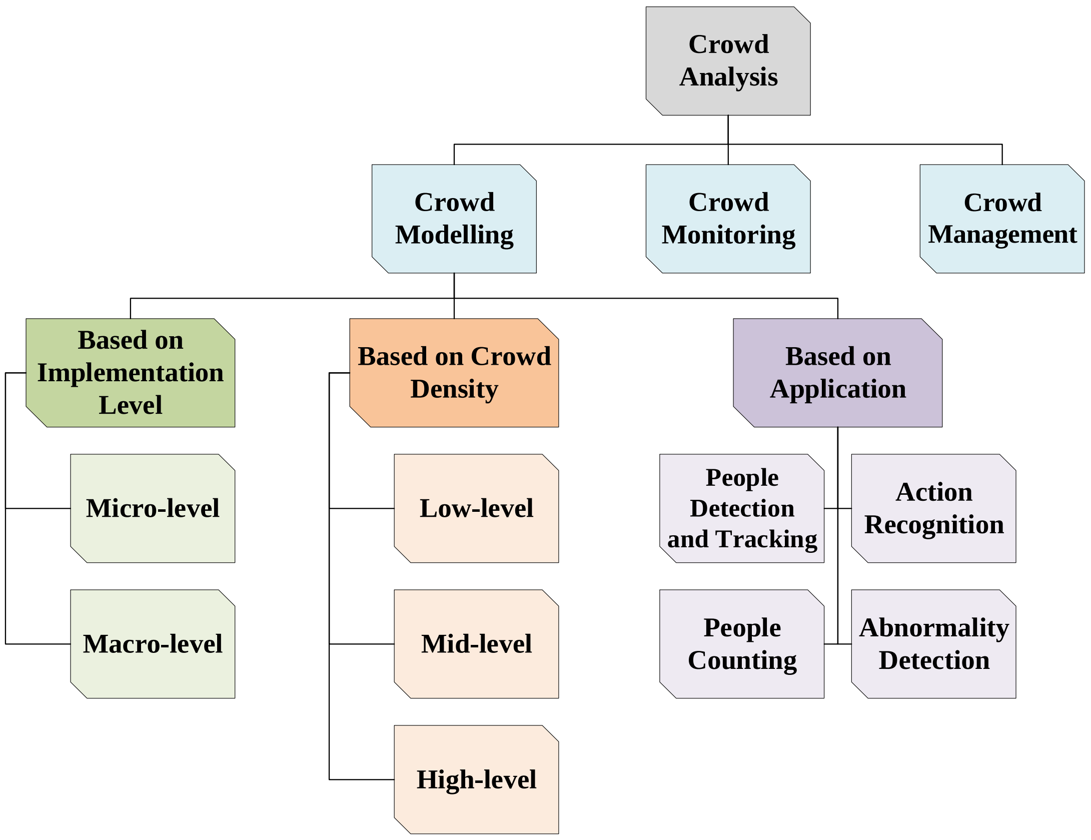


Figure 1.2. The schematic diagram of crowd analysis and its distinct categories

Another parameter used to categorize crowd analysis is the density of the crowd [12]. In fact, crowd modelling has always been depending on the density and it is categorized into three levels 1) low-level, 2) mid-level, and 3) high-level [12]. In the low level, the used techniques study the motion to characterize the features that detects individuals such as optical flow and background modelling. In the mid-level, the techniques are based on pattern recognition algorithms. The use of SVM and Adaptive Boosting (AdaBoost) classifiers on extracted features such as Histogram of Oriented Gradients (HOG) are very common. In this level and based on similar feature extractors, partial occluded scenes have been successfully analyzed. However, the use of dynamic texture models and the Lagrangian based techniques at the high level is becoming more popular recently [12].

Furthermore, crowd modelling techniques are also categorized based on the application they are designed for [12, 16]. There are several applications that requires crowd modelling such as 1) people detection and tracking, 2) people counting, 3) action recognition and behavioral analysis, and 4) abnormality detection [12, 16]. In addition, there are other parameters used to classify crowd modelling techniques into different, but less popular categories such as: offline versus online crowd modelling, and static versus dynamic crowd modelling [12, 16].

1.2 ***Thesis Scope***

The scope of this thesis focuses on the study of enhancing the CNN based pedestrian detector that depends on appearance saliency to detect people through integrating motion saliency information in visual surveillance. This shall allow the pedestrian detector results to be enhanced through integrating the motion information (temporal information) extracted from a sequence of frames. As a result, the detections will be more accurate as the false detections could be eliminated. It is worth noting that this approach will process the visual information at two levels. The motion information will model the crowd at the macro-level through extracting the holistic view of the crowd. On the other hand, the micro-level analysis where the individual entities of the crowd are detected is accomplished through the CNN based pedestrian detector. The impact of this approach will be verified through evaluating the performance of the proposed detector with and without motion saliency integration on several surveillance videos that have different settings (e.g. different illumination). The surveillance scenarios investigated in this work assume that the background of the scene is static and the camera is stationary. In addition, the crowd density will be assumed to be low. The proposed approach will be evaluated and benchmarked with other state-of-the-art algorithms against the same surveillance videos. Moreover, this thesis will focus on reporting all state-of-the-art pedestrian detectors in the literature. It is worth noting that having a robust pedestrian detector is very important as it is the starting point of any further analysis, such as people tracking, people identification, and behavior analysis. Even though these analysis challenges are very related to the pedestrian detection, it is considered to be out of the scope of this work.

1.3 *Thesis Objectives*

The main objectives of this thesis are:

1) Study and review existing algorithms proposed to perform pedestrian detection in the literature.
2) Propose a CNN based pedestrian detector that targets surveillance applications.
3) Integrate the proposed CNN based pedestrian detector with a motion flow algorithm to extract and utilize the motion saliency information to improve the performance.
4) Evaluate the performance post integrating the CNN based pedestrian detector with the motion flow algorithm to process surveillance videos. This will include reporting the detection accuracy, processing time, and any interesting and significant results.

1.4 *Thesis Outline*

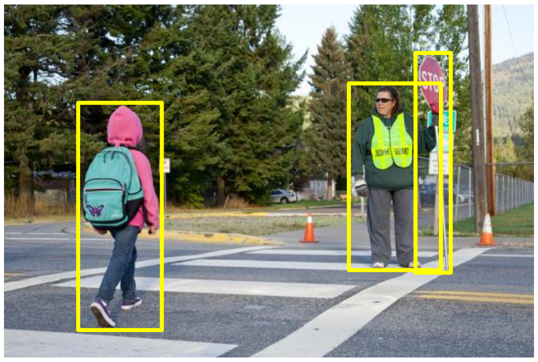
The remaining chapters of this thesis are structured as follows: In Chapter 2, the literature survey is reported. The chapter discusses, the performance metrics used to assess the pedestrian detectors, and the different motion flow algorithms and compare them. In addition, it will report and categorize the proposed algorithms for pedestrian detection in two categories: 1) classical pedestrian detectors, 2) CNN based pedestrian detectors. Finally, the two families of pedestrian detectors are compared and benchmarked. The proposed motion saliency based CNN pedestrian detectors and the training process are reported in Chapter 3. Chapter 4 summarizes the findings of this research project, discusses the results and states the limitations of the proposed detectors. Finally, Chapter 5 draws the conclusions of this work, and comments on the future work.

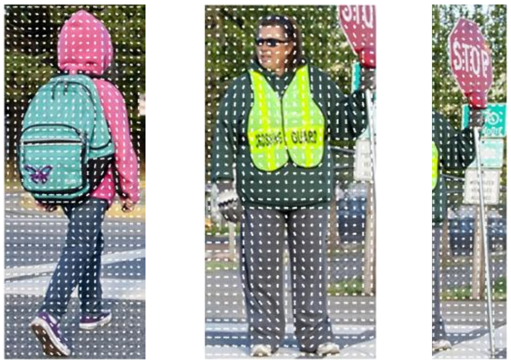
CHAPTER 2: LITERATURE REVIEW

The pedestrian detection problem in the field of computer vision and image processing is not a new problem where several researchers have tackled this problem thoroughly [17, 18]. A typical pedestrian detector consists of three stages [19]. Figure 2.1 summarizes the three stages of a typical pedestrian detector. The first stage is to propose bounding boxes that might contain pedestrians. These bounding boxes are also known as Regions of Interest (RoI). Afterwards in the second stage, the features of each proposed bounding box are extracted. Finally, classify all the proposed bounding boxes whether they contain a person or not using the extracted features. To perform these steps, several algorithms have been proposed. The main difference among them is the ability to extract more features that help in classifying the bounding boxes, and how these features are selected and classified [19].

This chapter is divided into five sections: Section 2.1 starts with the performance analysis and metrics used to benchmark pedestrian detection algorithms. Section 2.2 reviews the motion flow algorithms. Section 2.3 reviews classical pedestrian detectors algorithms while Section 2.4 concentrates on the CNN based pedestrian detection algorithms. Section 2.5 compares between the two families of pedestrian detectors.

(a) The input image


(b) The input image showing the proposed bounding boxes


(c) Extracting the features of each proposed bounding box (e.g. HOG features)

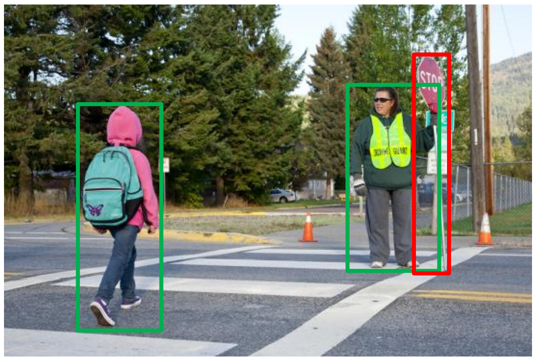

(d) Classifying the proposed bounding boxes and showing the final result (green means it is a person and red means it is not a person)

Figure 2.1. Image processing stages in a typical pedestrian detection system

2.1 ***Performance Analysis***

In this section, how to evaluate the work and benchmark it against other proposed work in the literature is reported. This would allow a better understanding of the reported algorithms. Qualitative and quantitative evaluation methods have been reported. The qualitative evaluation is based on results' visual inspection. However, the quantitative evaluation depends on computing different metrics [12]. For the purpose of pedestrian detection, there are various metrics used including but not limited to: Receiver Operating Characteristic (ROC) curve, recall and precision, accuracy, and error rates [12, 20].

It is worth noting that there are two methods to calculate the performance evaluation metrics, based on individual pixels or bounding boxes. The first method is more likely to be used with background modelling based algorithms. The second method is used in case of the appearance learning based algorithms. In addition, the datasets ground truth data availability and format, algorithm implementation mechanism, and the benchmarking protocol used affects which method is used to benchmark the proposed algorithm. However, regardless of which method is used, the variation in the quantitative evaluation metric of the proposed algorithm would be minimal [12].

2.1.1 Receiver Operating Characteristic Curve

The ROC curve is a plot that illustrates the performance of binary classifiers while varying the threshold of discrimination [21]. The plot consists of two quantities: True Positive Rate (TPR) and False Positive Rate (FPR). Equations 2.1 – 2.2 show how to calculate these two quantities.

$$TPR = \frac{TP}{TP + FN} \tag{2.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.2}$$

; where TP and FP stand for true positive and false positive, respectively, whereas TN and FN denote true negative and false negative, respectively.

To use this quantitative evaluation metric, it is assumed that the detection algorithm would take an image as an input and return a boundary box and its corresponding score of confidence for each detected person. Then, by using the generated boundary boxes and the ground truth boundary boxes of the dataset, the area of overlap between the boundary boxes and the area of their union are computed. Afterwards, the Intersection over Union (IoU) metric is used to determine if this detection was correct or not [20]. The IoU threshold was initially set by the PASCAL VOC criteria to be 50% [3]. In other words, if the IoU value is 50% or greater, it is considered a correct detection; otherwise, it is a false detection. Equation 2.3 summarizes the IoU metric [20].

$$IoU = \frac{Area(BB_D \cap BB_T)}{Area(BB_D \cup BB_T)} > 0.5 \tag{2.3}$$

; where BB_D corresponds to the detected boundary box and BB_T denotes the ground truth boundary box.

In case there are unmatched BB_D, they are counted as false positives, and in case there are unmatched BB_T, they are counted as false negatives. Afterwards, to compare and benchmark the detectors, the Miss Rate (MR) is plotted against the False Positive Per Image (FPPI) on a log-log scale while varying the confidence threshold of detection [20]. As the confidence threshold reduces, the MR would decrease while the number of false positives increases. At the end, the detector is evaluated by finding the log-average MR. This is calculated based on averaging the MR of the detector at nine FPPI rates that are evenly spaced in the log scale in the range from 10^{-2} to 10^{0} [20]. However, since for some detectors the curve might end before reaching the given FPPI rate, the minimum MR of the curve is used in that case [20]. The main target for this plot is to minimize the area under the curve since it corresponds to the MR.

Alternatively, if the pedestrian detector is based on binary classifiers, the MR could be found per window instead of per image. The plot in this case would be the MR against the False Positive Per Window (FPPW). This is usually used to benchmark the performance of different classifiers instead of detectors using positive and negative cropped windows [20]. In addition, it is used to evaluate different systems that generates RoI automatically [20]. It is worth noting that the performance evaluation would have different values when evaluated based on per window instead of per image. This is due several reasons including the fact that the tested RoIs are different in the per window case from the per image case [20]. Additionally, there are several factors selected when a classifier is being converted to a detector such as selecting the spatial strides, and the nonmaximal suppression to merge the nearby detections which affect the performance in the per image case [20]. However, it is assumed that having a better performance in the per window evaluation means having better performance in detection.

Since the ultimate aim of pedestrian and crowd detectors is to localize the pedestrian and crowds in an image, the per image evaluation is usually used [20]. As stated earlier, the case where the per window is used does not consider the performance of the detection as it isolates and concentrates on the performance of the classifier. Therefore, the use of per image evaluation is not limited to the pedestrian detection problem, it is usually used for any object detection problem as it provides a satisfactory measure of the algorithm’s detection error [20].

2.1.2 Recall and Precision

The recall, also known as sensitivity, is the fraction of relevant instances that are successfully retrieved. On the other hand, precision is known as the positive predictive value and it is the fraction of the retrieved instances that are relevant [12, 21]. Based on these two quantities, an additional quantity known as the F-measure is calculated. The F-measure is the harmonic mean of the recall and the precision. These quantities are calculated as illustrated by Equations 2.4 – 2.6 [21].

$$Recall = TPR = \frac{TP}{TP + FN} \tag{2.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

$$F_{measure} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.6}$$

The use of the recall and precision does not stop at this level. A curve known as the precision/recall curve is plotted to evaluate the object detectors in general for each class of objects [3]. The main idea of this curve is to define two parameters, the recall and the precision as follows. For a specific class, the recall is defined as the set of positive samples that satisfy a specific threshold where the precision is the set of the positive samples that belongs to the specified class and satisfy the same threshold [3]. Afterwards, a metric known as the Average Precision (AP) is found through averaging the precision at 11 recall values that are evenly spaced (0, 0.1, 0.2, … 1). Subsequently, the AP summarizes the whole precision/recall curve in just a one parameter. Therefore, to have a better performance in detection, it is required to have a precision at the different levels of recalls. The advantage of using the AP is its sensitivity and interpretability that visualize the performance of the detection algorithm at different recall levels especially low values [3].

Now, in case the detector is designed for multi class objects (e.g. car, pedestrian, plane, etc.), there will be an AP for each class. Therefore, a new metric known as the mean Average Precision (mAP) is used which is basically the mean of the AP per class [3]. It is worth noting in the pedestrian detection problem, the AP and mAP would be equivalent since there is one class of objects.

2.1.3 Accuracy

The accuracy is calculated as shown in Equation 2.7 [12].

$$Acccuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.7}$$

2.1.4 Error Rates

There are several error rates calculated and used in the evaluation of crowd and pedestrian detection proposed algorithms [12]. The most used error quantities are the Mean Squared Error (MSE), the Root Mean Square Error (RMSE), and the Mean Relative Error (MRE) [12, 22]. To calculate these quantities, Equations 2.8 – 2.10 are used.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(D_i - T_i)^2 \tag{2.8}$$

$$RMSE = \sqrt{MSE} \tag{2.9}$$

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|D_i - T_i|}{T_i} \tag{2.10}$$

; where D is the estimated value, T is the true value, and n is the number of frames.

2.2 *Motion Flow Algorithms*

In visual analysis of crowds, motion flow algorithms are used for scene understanding or for surveillance purposes through modelling and visualizing the dynamics of the crowd in the scenes. These motion algorithms provide temporal information since the motion features are detected using consecutive frames [23]. There are three types of motion levels analyzed using such algorithms: (a) global, (b) local and (c) persistent motion levels [12]. (a) The global motion is called global since it uses all the pixels of the image to estimate the motion. One of its applications is the removal of the camera motion (motion compensation) to be able to concentrate on the motion of objects. To model global motion, parametric transformations are used. (b) Local motion is applied on blocks of the image instead of all the pixels at once. One of the main targets of this approach is to track individual targets where the information of these targets should be available beforehand. (c) The concept of persistent motion concentrates on the persistent and collective dynamics in a visual scene as it was introduced in [24]. Despite that the persistent motion algorithms have their own category, they depend on other global and local motion models to model the motion flow [12].

Since the main idea of this work is to enhance the detection of the pedestrians, the review of motion algorithms would be concentrating more on the local motion algorithms. The local motion algorithms are usually used to track individual targets as stated earlier which would help in enhancing the performance of the detection.

2.2.1 Optical Flow

The optical flow motion algorithm is based on estimating the change of relative motion between an observer and the visual scene [12]. The detection in this algorithm is based on estimating the optical flow/velocity vectors for every pixel in the image so that motion in two consecutive frames is detected. Later, a clustering process based on the distribution of the flow vector field is used to have the complete movement information [25, 26]. This field characterizes the velocity of the moving edges, objects and surfaces in the visual scene and the direction of movement. Based on the magnitude of these vectors, it is easy to identify and segment the motion part of these two frames based on the zero/non-zero vector field estimated earlier. This estimation could be accomplished in three-dimensions in addition to time as the fourth dimension, however and for simplicity, it is usually mapped to a two-dimensional coordinate system and time is considered to be the third dimension [27]. It is worth noting that most of the optical flow algorithms are considered to be computationally slow and does not meet real-time processing speed [23, 26].

To use the optical flow algorithm, there are several constraints for the algorithm to estimate the flow vector field. The first constraint is known as brightness constancy where the brightness (intensity) of a point in the scene does not change despite that its location might change. In addition, it is assumed that the movement of a point between two consecutive frames is very small, which is known as spatial coherence. Equation 2.11 shows the equations for these two constraints [27].

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \qquad (2.11)$$

; where $I(x, y, t)$ is the intensity of the pixel at the position (x, y) at time instant t and $I(x + \delta x, y + \delta y, t + \delta t)$ is the intensity of the same pixel at the new near position $(x + \delta x, y + \delta y)$ at the time instant $t + \delta$.

To get the optical flow vector field, Taylor series is applied on the right hand side of Equation 2.11, and after neglecting the high order terms with additional simplifications, Equations 2.12 – 2.13 would be obtained where Equation 2.13 is equivalent to Equation 2.12 but it is written using vector representation [27].

$$I_x \cdot V_x + I_y \cdot V_y = -I_t \qquad (2.12)$$

$$\nabla I^T \cdot \vec{V} = -I_t \qquad (2.13)$$

; where I_x, I_y, and I_t are the x, y, and time derivatives of the brightness intensity, respectively, ∇I is the spatial gradient of the brightness intensity, and $\vec{V}$ is the velocity vector of the image pixel.

Equation 2.13 is used to compute the optical flow and it is known as the two-dimensional motion constraint equation or the gradient constraint equation. It is worth noting that ∇I and I_t are easy to compute using the sequence of frames but the vector $\vec{V}$ cannot be computed without additional equations. The most used methods to solve this equation and obtain the velocity vector are known as Lucas-Kanade and Horn-Schunck [27].

2.2.2 Background Subtraction

Another approach used to extract the motion information from a sequence of frames captured by a static camera is known as background subtraction. It is one of the most basic and reliable methods used to perform moving object detection and foreground analysis in a sequence of frames. The idea of this approach is to construct a background model where the current frame is subtracted from the background model as shown in Equation 2.14. Then, by thresholding $F(t)$, the segmented foreground is acquired as shown in Equation 2.15. The main drawback of this approach is that it is not able to handle dynamic background; hence, the camera should be static [12, 26].

$$F_n = I_n - B \tag{2.14}$$

$$Foreground = |F_n| > T = |I_n - B| > T \tag{2.15}$$

; where I_n is the current frame, B is the background model, F_n is the image that shows the pixels of frame I_n that experienced a change in its intensity level from the background model B, and T is the threshold level.

According to [28], there are several techniques to implement the background subtraction algorithm where the main difference among them is how the background model is constructed. The techniques include direct differencing, mean or median filtering, temporal median filter, Gaussian Mixture Model (GMM), and kernel density estimation. However, the use of GMM in crowd analysis have been dominating [12].

2.2.3 Frame Differencing

The frame differencing algorithm is very simple where the previous frame is subtracted from the current frame. The computations of this algorithm is very simple as shown in Equations 2.16 – 2.17 [26]. Similar to the background subtraction algorithm, a threshold might be used to detect the foreground of the scene and filter the difference of the two consecutive frames (Equation 2.17). The main drawback of this algorithm is its poor ability to detect the contour of the objects detected in the foreground which reduces the object detection accuracy [26].

$$F_n = I_n - I_{n-1} \tag{2.16}$$

$$Foreground = |F_n| > T = |I_n - I_{n-1}| > T \tag{2.17}$$

; where I_n is the current frame, I_{n-1} is the previous frame, F_n is the image that shows the pixels of frame I_n that experienced a change in its intensity level from the previous frame I_{n-1}, and T is the threshold level.

2.2.4 Comparison of Motion Flow Algorithms

As stated earlier, these three different motion detection algorithms have various advantages and disadvantages. Table 2.1 summarizes these points for the aforementioned algorithms based on the reported work in [25, 26].

Table 2.1 Comparison Between the Different Motion Flow Algorithms

Algorithm	**Accuracy**	**Computational Time**	**Comments**
Optical Flow	Moderate	High	▪ Complex computations and requires more memory ▪ Not preferred for real-time processing ▪ Very sensitive to noise ▪ Provides complete movement information ▪ Can handle dynamic backgrounds
Background Subtraction	Moderate	Moderate	▪ Simpler computations and requires lower memory ▪ Might require a buffer to store recent pixel values based on the used technique (e.g. median) ▪ Cannot handle dynamic background
Frame Differencing	High	Low to Moderate	▪ The simplest algorithm ▪ Requires a background frame at the initial step ▪ Suffers from noise ▪ Difficult to obtain accurate outline of moving objects ▪ Can process low resolution videos ▪ Can adapt with dynamic background

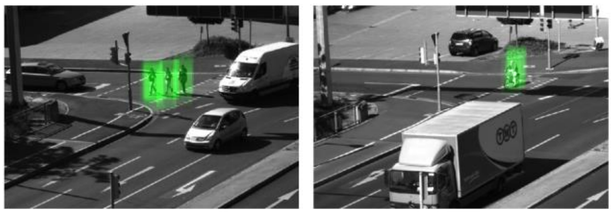
2.3 *Classical Pedestrian Detectors*

The techniques used in pedestrian detection are basically derived from the state-of-the-art algorithms proposed in the field of computer vision for multiple target tracking. In addition, the use of motion detection techniques is also common in this application [12]. In this section, different algorithms are discussed where they depend on different hand crafted features and not on CNNs.

2.3.1 Histogram of Oriented Gradients Features Based Pedestrian Detectors

The work in [29] proposed and implemented a pedestrian detector on a Field Programable Gate Arrays (FPGA) and GPU. The algorithm is based on two stages, the first stage is where background subtraction and foreground analysis are used to propose RoIs based on geometric features. Afterwards, HOG features are extracted and a kernel SVM classifier are used to categorize the proposed RoIs earlier into two classes: pedestrian or non-pedestrian. The advantage of using a kernel SVM is the ability to find non-linear decision boundaries through performing linear separation in a high dimensional feature space. The implemented kernel SVM utilizes a Gaussian kernel function. The final implementation of the system takes less than 100 ms to process 1000 windows, processing around 10 Frames Per Second (FPS). The system was tested using the INRIA [30] dataset where it showed a similar performance to the proposed algorithm in [30]. In addition, an outdoor experiment was conducted to verify the performance of the proposed algorithm. Figure 2.2 shows the experimental setup and some examples of the detection results. The results from the outdoor experiment reported an accuracy of 95.4% per frame.

(a) The outdoor setup of the experiment



(b) Samples of the detected pedestrians from the images of the outdoor experiment

Figure 2.2. The outdoor experiment setup and examples for the proposed algorithm in [29]

Furthermore, the work in [22] approached the pedestrian detection problem through combining appearance and motion algorithms (two are appearance based and one motion based algorithms). The outputs of the algorithms are combined to enhance the performance of pedestrian and crowd detection further. The first appearance algorithm performs head detection. Both HOG and Local Binary Patterns (LBP) features are used in addition to the multiple kernel learning classifier. The second appearance algorithm performs upright person detection through utilizing HOG features and an SVM classifier. The two appearance algorithms were originally proposed in [30, 31]. On the other hand, the proposed motion detection algorithm depends on the Dynamic Texture based Gaussian Mixture Model (DT-GMM). Afterwards, the initial motion detection results are used to enhance the performance of the head and upright person detectors. This would reflect back on the final results where the individuals of the crowd are localized and detected accurately. This approach does not improve only detection accuracy, but also reduce the false detections that has been found in the initial steps. The work used three video sequences from the PETS2009 [32] dataset to test the proposed approach. The MREs of the video sequence S1_L2_14-06, which contains highly dense crowd moving under significant occlusion, were found to be 9.64% and 6.25% for the head and upright person detectors, respectively. For another video sequence, S2_L2_14-55 where it contains movement of sparse moderate density crowd under varying lighting conditions, the MREs were found to be 10.17% and 1.61% for the aforementioned detectors, respectively.

2.3.2 Informed Haar-Like Features Based Pedestrian Detector

The proposed algorithm in [17] is based on multi-modal and multi-channel Informed Haar-like features. The use of such features allowed the algorithm to identify the characteristics of the human body parts' while being robust to changes in clothing and environmental settings. Binary and ternary modalities are the two template modalities used for the Haar-like features. The use of the ternary modality is to have the ability of representing more complex features of the pedestrians' silhouette through representing its corner regions. The features found are based on ten channels (three channels for the LUV colors, one channel for the gradient magnitude information, and six channels for HOG). These channels are found to be the most informative channels in [33] and was used in [34]. The use of color and gradient features provided the ability to overcome the variations in clothing. The algorithm used an AdaBoost classifier with a 2000 weak classifiers. The INRIA and Caltech [20] datasets have been used to benchmark the system. Reported results showed that the log-average MR for the INRIA and Caltech datasets are 14.43% and 34.60%, respectively.

2.3.3 Body-Parts Pedestrian Detectors

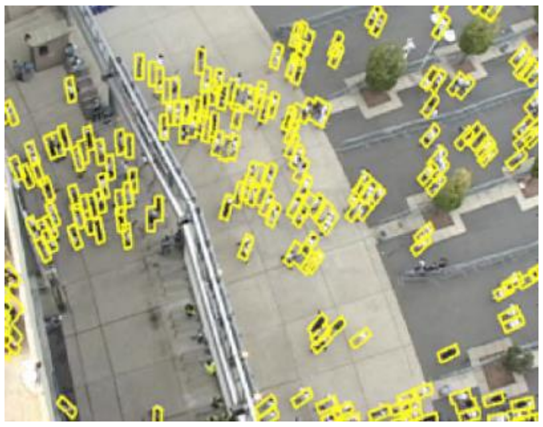
The researchers in [35] proposed a density aware people detector, using the proposed regression based density estimator in [36]. The approach implemented does not depend on the full body of a pedestrian. Instead, it uses part of the body, the head region, to detect a pedestrian which improved the immunity of the algorithm to occlusions. In addition, the algorithm roughly estimates the viewpoint of the camera. This improves the result through identifying the constraints on the size of a pedestrian in the image. To train and test the algorithm, the researchers have used videos from the Internet besides their own collected dataset. The detector training was accomplished through using manually annotated images with both positive and negative samples for the head regions. The negative samples were created using the same training images, where the overlap with the positive samples is minimal. The detection AP values reported for 1 pedestrian in 4 m^2 was 72% where it decreased to reach around 48% for 9 pedestrians in 4 m^2.

Similarly, the work in [37] uses parts of the body to detect pedestrians. The followed approach consists of two-layers. The first layer is based on the model proposed in [38] where it utilizes a deformable template to represent part and global appearances. The second layer uses a mixture model that takes the part scores and the appearance as features to model the pedestrians and handle the occlusions. To train and test the proposed approach, the PETS2009 and the UCSD crowd [39] datasets were used. From the PETS2009, the S1_L2 sequence, which contains highly dense crowd, was used for training. The two sequences S2_L2 and S2_L3 were used to test the algorithm. The sequence S2_L2 contains a medium density crowd where the sequence S2_L3 contains a high density crowd. The reported AP on the two sequences were 75.5% and 65.5%, respectively. From the UCSD, the sequence S1 was used for training where the sequence S2 was used for testing. The reported AP is 68.3%. It is worth noting that the sequences from the UCSD dataset were manually annotated by the researchers.

2.3.4 Combined Pedestrian Detector

Pedestrian detectors that utilize different algorithms to detect pedestrians based on specific criteria are categorized in this report as combined pedestrian detectors. For example, for every video setting, there is a specified algorithm to detect the pedestrians.

The work presented in [40] performs visual analysis of small groups in pedestrian crowds. Two algorithms are proposed based on the elevation angle of the captured videos. In case the videos are captured from a high elevation angle, the pedestrians are detected through implementing a Reversible Jump Markov Chain Monte Carlo (RJMCMC) as shown in Figure 2.3.a. This is used to process the binary segmentation produced by the adaptive background subtraction. However, if the videos are of high resolution, the pedestrians are detected based on HOG features as proposed in [30]. Figure 2.3.b shows an example of the high resolution images with low elevation angle. The HOG features are calculated on the output of a background subtraction mask that helps in detecting moving pedestrians while suppressing the gradients in the regions where it is most likely have no pedestrians. This allowed to reduce the false positives in such regions. To test this work, the researchers have collected their own dataset and created its ground truth data. The reported accuracy of the detection algorithm on the collected dataset was 96%.


(a) A sample of pedestrian detection from a high elevation angle using RJMCMC

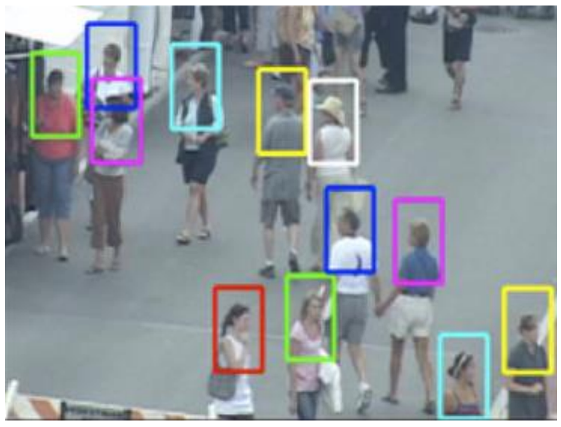

(b) A sample of pedestrian detection from a low elevation angle using RJMCMC

Figure 2.3. Examples from the pedestrian detection algorithms implemented in [40]

2.3.5 Aggregated Channel Features Based Pedestrian Detector

The work in [41] proposes the Aggregated Channel Features (ACF) pedestrian detector. Based on [33] and similar to [17, 34], the algorithm uses the same ten channels. Then, the fast feature pyramid is computed based on an efficient method. The efficient method computes the pyramid of features based on one scale per octave. An octave is the interval between two scales where one of them is the double of the other. The features at the intermediate scales are found from the features of the nearest scale per octave. Hence, the speed of computation increases while the accuracy is slightly affected. To perform pedestrian detection, a decision tree is used where it was trained and combined using AdaBoost. The ACF algorithm processes 32 FPS using the fast feature pyramid features and it would drop to 12 FPS when the exact feature pyramids are calculated, which is not suitable for real-time processing. The proposed ACF pedestrian detector was tested using four different datasets: INRIA, Caltech, TUD-Brussels [42], and ETH [43]. The accuracy of the proposed ACF pedestrian detector among the four sets was found by using the log-average MR between 10^{-2} and 10^{0} FPPI. The average of the log-average MR among the four datasets was found to be 40% when the exact feature pyramid is computed and 41% when the fast feature pyramid is computed.

2.3.6 Locally Decorrelated Channel Features Based Pedestrian Detector

Afterwards, the work presented in [44] concentrated on improving the performance of boosted detectors such as the ACF pedestrian detector [41]. Despite the evolution of these boosted detectors, the use of orthogonal decision trees (decisions are based on single feature splits) is still the most popular. The main reason behind this dominance is their simplicity and high efficiency. For example, using oblique decision trees (decision are based on multiple feature splits) increases the computational cost of training and testing. Yet, they demonstrated effectiveness in processing high dimensional data with correlated features [44, 45].

However, the work presented in [46] inspired the idea of decorrelating the channel features used to detect pedestrians (or objects in general) before applying an orthogonal decision tree algorithm. This is critical since orthogonal decision trees do not perform well while processing high dimensional data with highly correlated features [44]. Thus, the proposed algorithm would avoid the use of oblique decision trees. Initial tests in [44] have shown that the use of decorrelation filters with orthogonal decision trees outperforms the oblique decision trees.

Subsequently, the ACF detector was modified by applying four decorrelation linear filters per channel to the 10 channels of the ACF detector. Hence, the total number of locally decorrelated channel features is 40. Then, to simplify the computations and maintain similar efficiency to the ACF detector, the 40 channel features are downsampled by a factor of 2. This downsampling reduces the size of the 40 channel features to an equivalent size of the 10 original channel features while having minimal impact on the results. It is worth noting that the implementation of the decorrelating filters is the only addition to the ACF detector. Hence, the new detector has been called the Locally Decorrelated Channel Filters (LDCF) pedestrian detector. After testing the LDCF detector on the INRIA and the Caltech pedestrian datasets, the log-average MR is found to be 14% and 25%, respectively.

2.4 *CNN Based Pedestrian Detectors*

When the CNN object detectors are studied, the family of Regional-based Convolutional Neural Network (R-CNN) cannot be ignored. This is due to the performance of the Faster R-CNN where it achieved the best results on the object detection challenges such as PASCAL VOC [47] and Microsoft Common Objects in COntext (MS COCO) [48] [49]. Even though the family of R-CNN networks were proposed to perform multi-class object detection, they can be finetuned to perform specific class object detection. Therefore, the family of the R-CNN are reported in this section to understand their main building blocks and evolution.

2.4.1 Regional-based CNN

The development of the R-CNN started in [50] where it boosted the performance of object detection from 35.1% mAP for the work in [51] to 53.7% on the PASCAL VOC 2010. The proposed architecture for the R-CNN is shown in Figure 2.4. Selective search is used to generate class independent region proposals. Selective search was used to propose RoI for the purpose of comparing the proposed R-CNN with the state-of-the-art algorithms at that time as they used selective search. Afterwards, AlexNet [6] was used to generate a 4096 dimensional feature vector for each of the proposed regions. At the end, linear SVMs are used to classify the detected objects. To process one image on a GPU, it takes around 23 seconds, which might vary based on the class of the object being detected.

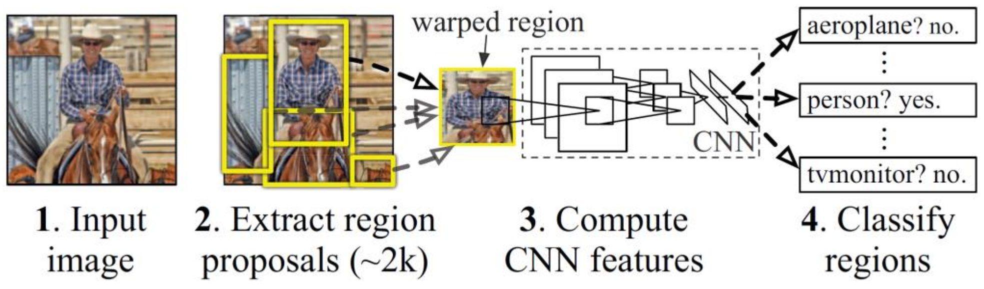


Figure 2.4. The proposed architecture of the R-CNN [50]

2.4.2 Fast R-CNN

The work of R-CNN did not stop at this step where another approach called Fast R-CNN was proposed in [52]. The proposed architecture of the Fast R-CNN is shown in Figure 2.5. The proposed network processes the input image to extract a feature map using several convolutional layers from the VGG-16 network [53]. Afterwards, the RoI pooling layer extracts a feature vector from the feature map that has a fixed length for each object proposal in the image. The extracted vector is then processed by the Fully Connected (FC) layers. At the end, two output layers are used to detect the class of the object using a softmax layer and the refined bounding box is regressed by the bounding box regressor layer. The reported mAP for the PASCAL VOC 2010 was 66.1% where the mAP for the PASCAL VOC 2012 was 65.7%. It takes around 0.32 seconds to process one image using the Nvidia K40 GPU where it takes 47 seconds for the R-CNN to process one image on the same GPU as reported in [52].

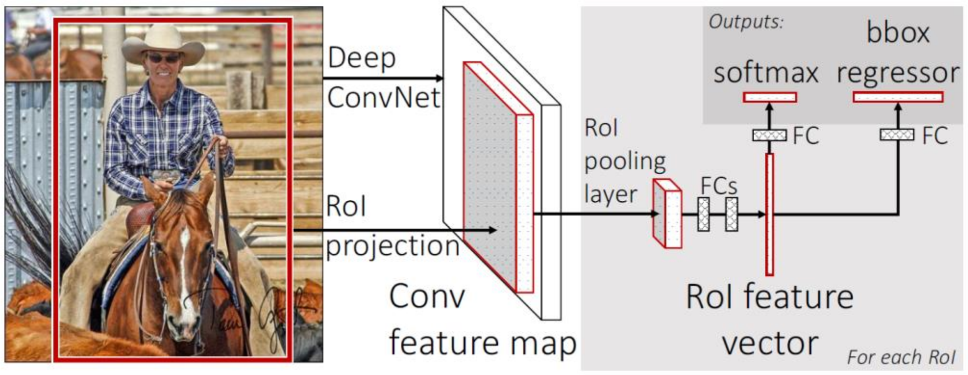


Figure 2.5. The proposed architecture of the Fast R-CNN [52]

2.4.3 Faster R-CNN Pedestrian Detector

The work in [54] has further enhanced the performance by proposing the Faster R-CNN architecture shown in Figure 2.6. The proposed architecture consists of two networks. The first network is the Region Proposal Network (RPN) which is a Fully Convolutional Network (FCN). It takes an input image and proposes boundary boxes and their corresponding confidence score. This is accomplished through sliding a window on the input image. Then anchors of different sizes and scales are used to propose regions inside the sliding window. The different sizes and scales anchors are used to address the problem of multi-scale and several aspect ratios. The proposed RPN uses nine anchors that have three different sizes and three different scales. The second network is the Fast R-CNN detector proposed in [52]. The contribution of this work lies in the idea that both RPN and Fast R-CNN networks share several convolutional layers to make it more efficient. However, this would complicate the training process of the Faster R-CNN network as each network will modify the shared convolutional layer differently.

The Faster R-CNN mAP testing results on the PASCAL VOC 2007 and 2012 were reported to be 78.8% and 75.9%, respectively. It is found that the implementation of the Faster R-CNN (based on VGG-16) on the Nvidia K40 GPU takes 0.2 seconds to process one image. However, if the implementation was based on the ZFNet [7], it takes approximately 0.06 seconds to process one image. In other words, it is capable of processing 5 or 17 FPS depending on which CNN is used. This made the Faster R-CNN implementation one step closer to the real-time processing assuming that a camera takes 25 FPS which translates to processing each frame or image in 0.04 seconds (or 40 milliseconds).

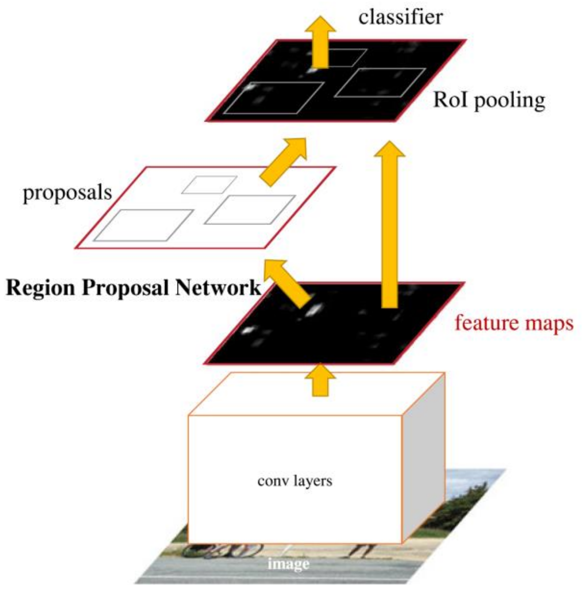


Figure 2.6. The proposed architecture of the Faster R-CNN [54]

After the proposal of Faster R-CNN, the researchers in [55] implemented a pedestrian detector using the Faster R-CNN architecture. The reported log-average MR on the Caltech dataset was 20.2%. This MR would drop to 16.2% in case the ‘atrous convolution’ presented in [56] is used. On the other hand, the work also implements a RPN only to perform pedestrian detection where it demonstrated better performance as the log-average MR decreased to reach 14.9%. Furthermore, the log-average MR decreases further in case the RPN and R-CNN networks are combined to reach 13.1%.

Therefore and after these findings, the researchers in [55] investigated why Faster R-CNN has an outstanding performance in multi-class object detection problems while its performance is not at the same level in pedestrians detection. Furthermore and as stated earlier, it was found that the performance of the RPN as a pedestrian detector is better than the Faster R-CNN as a standalone pedestrian detector. The accuracy loss in Fast R-CNN is due to its classifier. The first reason for the degradation in accuracy is the size of the objects to be detected. The size of the pedestrian instances is small and with the use of a pooling layer, the extracted low resolution feature map is found to be not discriminative which cause the reduction in accuracy of the Fast R-CNN classifier. In addition, most of the false detections are usually due to the hard background instances, which causes the Faster R-CNN pedestrian detector to suffer.

2.4.4 Regional Proposal Network – Boosted Forest Pedestrian Detector

Subsequently, the researchers in [55] proposed the RPN-Boosted Forest (RPN-BF) architecture shown in Figure 2.7. It addresses the first point through using the crafted features at the RPN’s first layers where the resolution is high to increase the size of the feature map. In addition, it utilizes a cascaded BF to classify the proposed bounding boxes by the RPN. The use of BF improves the ability to classify hard negative background instances that are proposed by the RPN. The proposed architecture depends on the deep features extracted by the RPN and uses the BF for classification. As a result, it is considered more computational efficient and it overcomes the older algorithms that depend on hand crafted features and BF.

The implemented RPN is based on the VGG-16 network that is pretrained on the IMAGENET dataset. Since the proposed RPN aims to detect pedestrians only, the modifications applied in [57] were respected. The aspect ratio of the nine anchors used have been fixed to a single aspect ratio of 0.41 (width/height). This ratio is used since it is found to be the average aspect ratio of the pedestrians [57]. Thus, the RPN proposes bounding boxes and their corresponding confidence scores which are passed to the BF in addition to the extracted features.

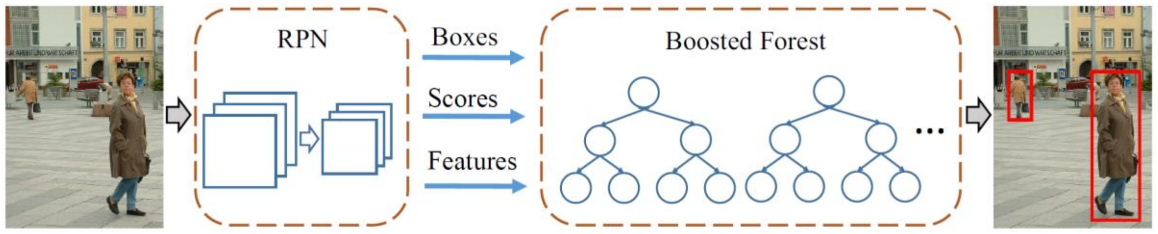


Figure 2.7. The proposed architecture of the RPN-BF [55]

The cascaded BF is trained using the outputs of the RPN (proposed regions, confidence scores, and features). The hyper-parameters are set according to [58] and the RealBoost algorithm is used in training [59]. The BF processes the proposed bounding boxes in a biased fashion where the produced score by the RPN is used as an initial score.

The RPN-BF testing was accomplished among four datasets: Caltech, INRIA, ETH, and KITTI [60]. The log-average MR reported for the Caltech dataset is 9.6% where it took 0.5 seconds to process one image (2 FPS) on the Nvidia Tesla K40 GPU. The reported log-average MR for the INRIA and ETH datasets were 6.9% and 30.2%, respectively. For the KITTI dataset, the reported mAP on moderate was 61.15%.

2.4.5 The HyperLearner Pedestrian Detector

Another pedestrian detector architecture is proposed in [18]. It is known as HyperLearner. It is based on integrating channel features into the CNN based pedestrian detector. The work tested three different group of channels to be integrated to improve the performance: apparent-to-semantic channels, temporal channels, and depth channels.

The apparent-to-semantic channels include in addition to the same ten channels used in [17, 34], the edge, the segmentation, and the heatmap channels. The edge channel is extracted using the holistically-nested edge detection network proposed in [61], where a FCN generates the segmentation channel which is blurred to create the heatmap channel. For the temporal channels, optical flow is used to extract temporal features through using the consecutive frames of a video. However, the depth channels are generated using the employed depth sensors to create depth information. It is worth noting that edge and semantic segmentation channel features demonstrated more promising results when integrated with the CNN based pedestrian detector.

The proposed HyperLearner consists of four networks as shown in Figure 2.8. First, the input image is fed to the body network, a modified VGG-16 network pretrained on IMAGENET, to generate the aggregated activation maps. Later, a Channel Feature Network (CFN), which is a FCN, predicts the channel feature map through processing the aggregated activation maps. Afterwards, the proposed Faster R-CNN (RPN and Fast R-CNN) in [54] are used to propose RoI and perform pedestrian detection.

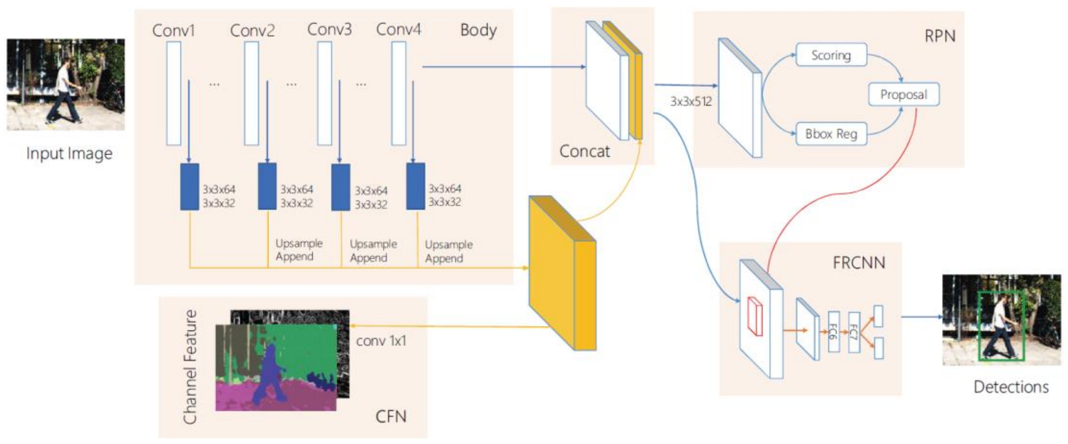


Figure 2.8: The proposed HyperLearner pedestrian detector [18]

To train the proposed HyperLearner, four stages are required. In the first stage, only the CFN is trained while fixing the body network (since it is a pretrained VGG-16 network). Then, the RPN is trained while fixing both the body network and the CFN. In the third stage, the body network, the CFN, and RPN are fixed while the Faster R-CNN network is trained. However, in the fourth stage, all the networks are trained.

To evaluate the HyperLearner, the following three datasets were used: KITTI, Cityscapes [62], and Caltech datasets. On the KITTI dataset, integrating the edge channel boosted the results slightly higher than the segmentation features to reach a mAP of 71.51% on moderate. For the Cityscapes dataset, the AP reported was 87.67% for the 720p frames resolution. Furthermore, the log-average MR reported for the Caltech dataset (based on the new annotations) was 5.5%. The proposed network implementation using a single Nvidia TITAN X takes 0.25 seconds to process one 720p frame which is equivalent to 4 FPS.

2.4.6 Simultaneous Detection and Segmentation R-CNN Pedestrian Detector

Another work that investigated the impact of using semantic segmentation on the pedestrian detection problem was reported in [63]. The proposed algorithm allows joint learning on semantic segmentation and pedestrian detection while having limited or minimal impact on the network. Hence, the proposed architecture is known as Simultaneous Detection and Segmentation R-CNN (SDS-R-CNN) and it is shown in Figure 2.9. It contains two networks: a RPN and Binary Classification Network (BCN). The contribution in the architecture lies in the semantic segmentation infusion layer. This shall generate feature maps based on semantic masks which improve the ability to perform pedestrian classification.

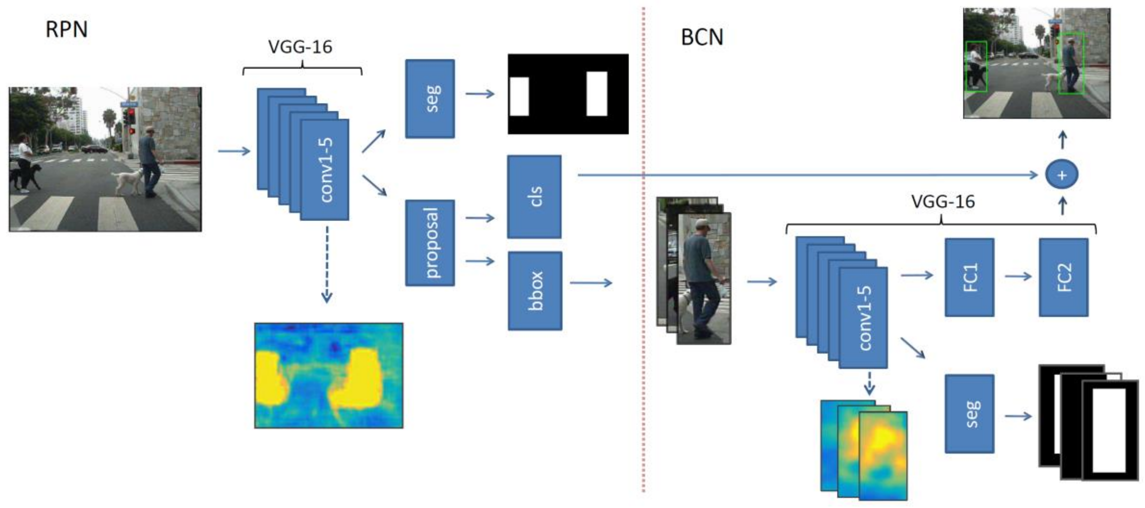


Figure 2.9: The proposed SDS-R-CNN pedestrian detector [63]

The RPN implemented is the same one proposed in the work of [54] where the same modifications suggested in the work of [55] has been followed. The RPN is based on the VGG-16 network pretrained on IMAGENET. The BCN is used to classify the proposals of the RPN and refine their scores. Another VGG-16 network is used in the BCN to avoid performance degradation in the pedestrian detection capability as shown in [55]. This would reduce the efficiency of the proposed algorithm as the computations will be calculated twice. However, the advantage would be the ability to detect the hard samples passed by the RPN. In addition, shared networks usually provide similar scores which is not helpful when the networks are fused.

The SDS-R-CNN evaluation used Caltech and KITTI datasets. The log-average MR reported for the Caltech dataset is 7.36% where the reported mAP on moderate for the KITTI dataset is 63.05%. The network takes 0.21 seconds (4.76 FPS) to process one frame of resolution 960×720 using a single Nvidia TITAN X.

2.5 *Comparison and Overview of the Pedestrian Detectors in the Literature*

Table 2.2 and Table 2.3 summarize the two categories of algorithms stated in the literature earlier. In general, it is clear that most of the classical algorithms does not process the raw input image. However, these algorithms extract specific features from the input images and process the extracted features to perform pedestrian detection such as the 10 channels used by the ACF detector. However, the CNN based algorithms take the input image and extract deep convolutional features. The extracted features are selected through the supervised learning of the networks. Thus, the CNN based algorithms have clearly outperformed the classical algorithms as they are using more optimal features to discriminate the contents of the images.

All in all, the performance gain in the CNN based algorithms over the classical algorithms came with the price of higher processing time. Even though using shallower CNN reduced the time, the performance was affected while not satisfying the real-time requirement as well. In addition, it could be noticed that after the proposal of the Faster R-CNN, the proposed CNN based pedestrian detection are using more than raw images to extract the features and detect pedestrians. For example, the HyperLearner best performance was accomplished when the edge and semantic segmentation channel features were integrated in the process of pedestrian detection. Similarly, the SDS-R-CNN utilized the semantic masks to boost the performance. These additional features are extracted through adding more layers or even networks to the proposed pedestrian detector. As a result, the processing time of the detector increased.

However, what if such additional features could be extracted using simpler algorithms. In fact, the idea of this work is inspired from the work presented in [22], and the review reported in [55]. As stated earlier that there are two reasons for the accuracy of a Faster R-CNN network to drop, where the second reason stated that the drop is related to the hard background instances detected as pedestrians. However, and since the application targeted in this work is visual surveillance, the effect of this limitation on the proposed pedestrian detector could be minimized if not completely eliminated. This could be accomplished through utilizing a similar idea to the one reported in [22]. The use of a motion flow algorithm to detect initial RoI in the images before performing pedestrian detection will provide an idea of where the people could be located in the image. Therefore, any detections of hard background instances could be corrected through using the motion flow RoI. Yet, it is worth noting that this approach might introduce additional constraints and limitations to the proposed detector depending on the selected motion flow algorithm, and the method of integrating the motion flow information in the detection process.

Table 2.2 Summary of the Classical Image Processing Pedestrian Detectors Proposed in the Literature

Algorithms	**Features**	**Dataset**	**Performance Metric**	**Performance Measure**	**Processing Speed [FPS]**
[40]	RJMCMC (or) HOG	Own dataset	Accuracy	96%	-
[35]	Parts of the Body (Head Region)	Own dataset + Internet Videos	AP	48% - 72%	-
[37]	Deformable Template	PETS2009	AP	65.5% - 75.5%	-
		UCSD Crowd		68.3%	
[29]	Background Subtraction + HOG	INRIA	-	-	~10
		Own dataset	Accuracy	95.4%	
[17]	Informed Haar	INRIA	Log-average Miss Rate	14.43%	-
		Caltech		34.60%	
[22]	HOG + LBP + DT-GMM (Head Detector)	PETS2009	MRE	9.64% - 10.17%	-
	HOG + DT-GMM (Pedestrian Detector)			6.25% - 1.61%	
[41]	ACF - Fast Feature Pyramid	INRIA	Log-average MR	17%	32
		Caltech		45%	
		TUD-Brussels		52%	
		ETH		51%	
	ACF - Exact Feature Pyramid	INRIA	Log-average MR	17%	12
		Caltech		43%	
		TUD-Brussels		50%	
		ETH		50%	
[44]	LDCF	INRIA	Log-average MR	14%	-
		Caltech		25%	

Table 2.3 Summary of the CNN Based Pedestrian Detectors Proposed in the Literature

Algorithms	**CNN Architecture**	**Dataset**	**Performance Metric**	**Performance Measure**	**Processing Speed [FPS]**
[55]	Faster R-CNN	Caltech	Log-average MR	20.2%	-
	Faster R-CNN (atrous convolution)	Caltech		16.2%	-
	RPN	Caltech		14.9%	-
	RPN + R-CNN	Caltech		13.1%	-
	RPN + BF	Caltech		9.6%	2
		Caltech (new annotations)		7.3%	
		INRIA		6.9%	
		ETH		30.2%	
		KITTI (moderate)	mAP	61.15%	
[18]	HyperLearner	Caltech (new annotations)	Log-average MR	5.5%	4
		Cityscapes	AP	87.67%	
		KITTI (moderate)	mAP	71.51%	
[63]	SDS-R-CNN	Caltech	Log-average MR	7.36%	~4
		KITTI (moderate)	mAP	63.05%	

CHAPTER 3: PROPOSED MOTION SALIENCY AIDED CNN PEDESTRIAN DETECTOR

In this chapter, the design of the proposed Motion Saliency Aided CNN pedestrian detector is reported. Section 3.1 shows an overview of the proposed motion saliency aided pedestrian detector. Section 3.2 discusses the hardware and software setup used in this study to implement the algorithms developed. Sections 3.3 reports the proposed motion flow algorithm and Section 3.4 discusses the proposed CNN pedestrian detector.

3.1 *An Overview of the Motion Saliency Aided Faster R-CNN Pedestrian Detector*

As stated earlier that the main aim of this work is to investigate the potential improvement of the performance of a CNN based pedestrian detector after integrating it with a motion flow algorithm. Thus, one of the key points investigated in this work is how the motion saliency information could be used to enhance the performance. A typical CNN based pedestrian detector takes the raw image as input and process it to detect the pedestrians. However, in this work, this will be slightly modified to integrate the motion information. In this work, two methods of motion integration are used. The first approach integrates the motion information in a pre-processing stage; where a sequence of frames is processed to perform motion analysis and mask the raw image before processing it by the Faster R-CNN detector as illustrated in the block diagram in Figure 3.1. Hence, this approach is called Motion Masked Faster R-CNN (MM Faster R-CNN) pedestrian detector. On the other hand, the second approach integrates the motion information after processing the image using the Faster R-CNN pedestrian detector. Hence, the detection results are corrected based on the motion information at a post-processing stage as shown in Figure 3.2. Therefore, this approach is called Motion Corrected Faster R-CNN (MC Faster R-CNN) pedestrian detector. The choice of the Faster R-CNN architecture to perform pedestrian detection is discussed later in Section 3.4.

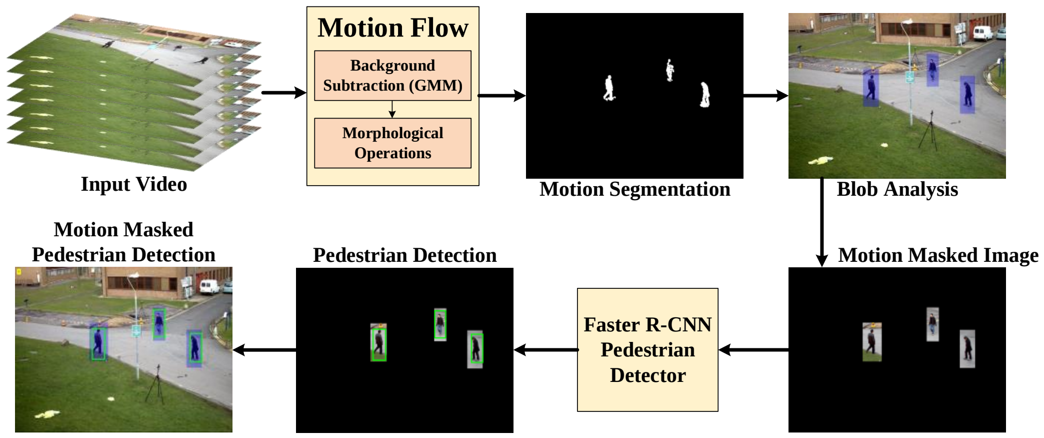


Figure 3.1. The block diagram of the MM Faster R-CNN pedestrian detector

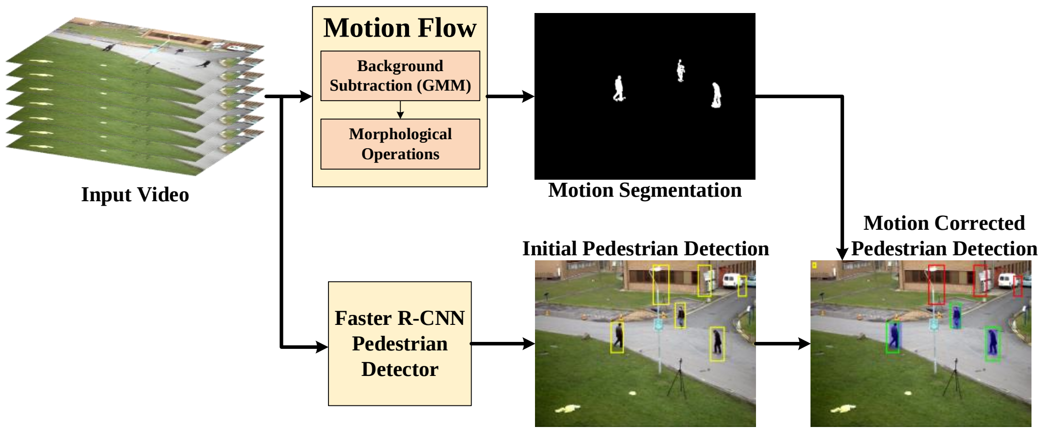

Motion Flow
Background Subtraction (GMM)
Morphological Operations
Input Video
Motion Segmentation
Faster R-CNN Pedestrian Detector
Initial Pedestrian Detection
Motion Corrected Pedestrian Detection

Figure 3.2. The block diagram of the MC Faster R-CNN pedestrian detector

3.2 *Hardware and Software Setup*

In this work, the proposed algorithms are developed on the Dell XPS 9560. The laptop is running Microsoft Windows 10 Operating System, on an Intel Core i7 7700HQ processor, 16 GB of RAM, and a 4 GB GDDR5 Nvidia GeForce GTX 1050 (640 CUDA core, memory bandwidth of 112 GB/s) [64, 65]. In addition, a 12 GB GDDR5X Nvidia TITAN Xp (3840 CUDA core, memory bandwidth of 547.7 GB/s) is connected as an external Graphical Processing Unit (eGPU) through the USB-Type C Thunderbolt 3 connection. It is worth noting that the Dell XPS 9560 Thunderbolt 3 port utilizes 2 lanes of Peripheral Component Interconnect Express Generation 3.0 (PCIe 3.0) instead of a typical Thunderbolt 3 which utilize 4 lanes. This limits the bandwidth of the connection to 20 Gbps instead of 40 Gbps [64]. To develop, train, implement and test the algorithms, MathWorks MATLAB software environment is used since the author is familiar with the software package. Figure 3.3 summaries the hardware and software systems used.

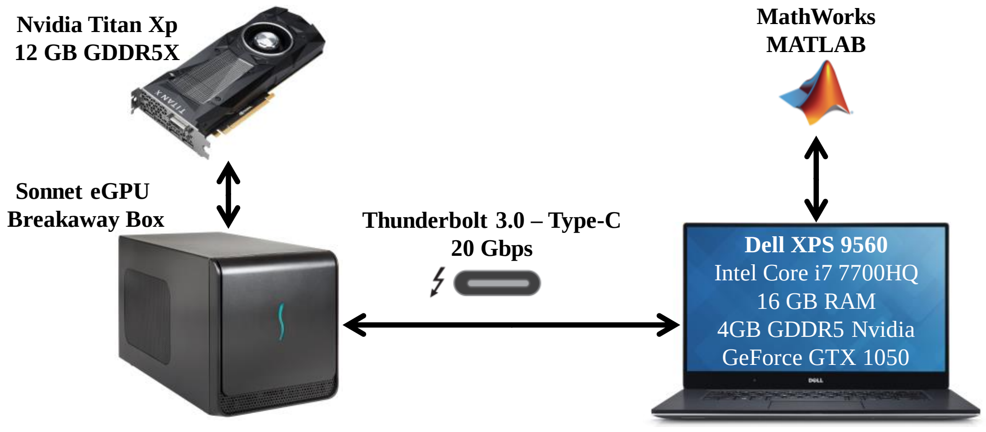


Figure 3.3. The hardware and software setup used in this work

3.3 *Proposed Motion Flow Algorithm*

As reported in the literature review section, there are several motion flow algorithms used to detect the moving objects and perform foreground analysis in frame sequences or videos. In this work, the aim of using the motion flow algorithm is to perform foreground extraction and region detection to perform holistic crowd representation. This is based on the assumption that pedestrians do not stand still completely in the scene which indicates that people are in this part of the image. Based on the literature, the most common algorithm used in visual analysis of crowds is based on background subtraction and GMM, due to its reliability and high processing speed [12]. Yet, that does not mean this is the best algorithm in terms of performance. Actually, some algorithms might provide better motion information such as optical flow [27]; however, the computational cost of such algorithms increases the processing time and complicates the system further while having minimal or no impact on the result required in this work.

The use of GMM to implement background subtraction is to use K Gaussian distributions to model each pixel in the background. The number K is selected to be a small number and it is assumed that the different Gaussians are used to represent different colors. Thus, each pixel in the scene is modelled using the mixture of K Gaussian distributions, and the probability of having a pixel with a certain value X_n is shown in Equation 3.1 [66].

$$p(X_n) = \sum_{i=1}^{K} \omega_i \cdot \eta(X_n, \mu_i, \Sigma_i) \tag{3.1}$$

; where ω_i is the estimated weight of the ith Gaussian distribution, μ_i is the mean of the ith Gaussian distribution, Σ_i is the covariance matrix, and η is the Gaussian probability density function as shown in Equation 3.2:

$$\eta(X_n, \mu_i, \Sigma_i) = \frac{1}{(2\cdot\pi)^{\frac{n}{2}}\cdot|\Sigma|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(X_n-\mu_i)^T\Sigma^{-1}(X_n-\mu_i)\right)} \tag{3.2}$$

Figure 3.4 illustrates the proposed motion flow algorithm. First, background subtraction using GMM is applied to the input sequence of frames to detect the foreground. Afterwards, the detected foreground is filtered using morphological operations. This will remove any small regions where motion has been detected. At the end, blob analysis is applied to detect the moving objects in the scene as shown in the figure.

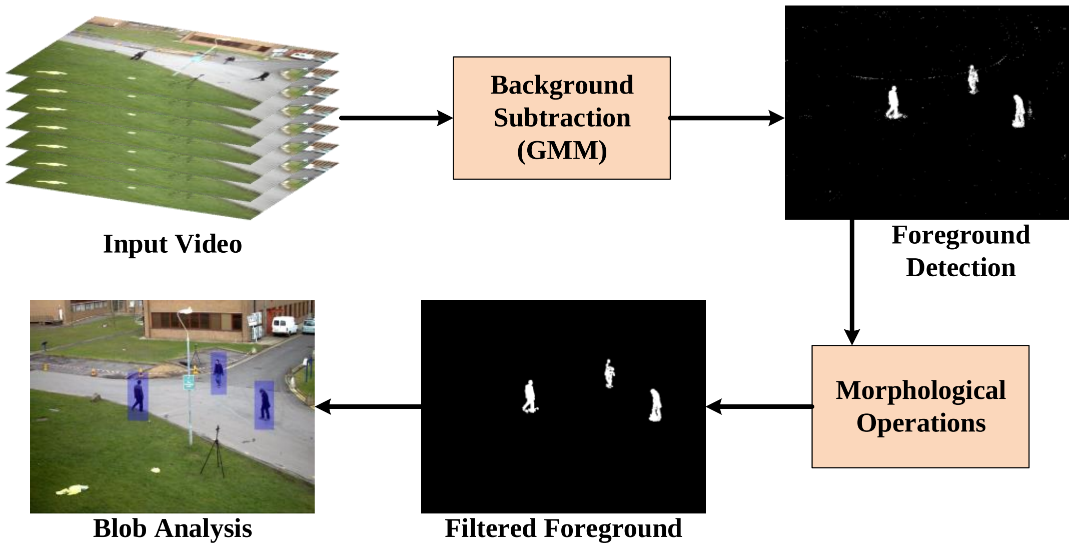


Figure 3.4. The block diagram of the proposed motion flow algorithm

3.4 *Proposed Faster R-CNN Based Pedestrian Detector*

As reported in the literature, there are several CNNs proposed to perform pedestrian detection. Some of these detectors process the raw images to perform pedestrian detection where other detectors process the raw images using additional convolutional layers to extract additional feature channels such as the HyperLearner [18, 63]. This step of extracting additional feature channels enhanced the proposed CNNs performance while its processing speed decreased. However, in this work, the aim is to avoid using these additional convolutional layers and use a motion flow algorithm to enhance the performance while having minimal impact on the processing speed. Therefore, the Faster R-CNN architecture is used in this work to perform pedestrian detection since it is the current state-of-the-art in object detection and it is one of the fastest CNNs [49].

To be able to use the Faster R-CNN as a pedestrian detector, it is required to build the network and finetune it. First, to create the Faster R-CNN, it is required to have a classification network that is used to extract convolutional features from the raw image. These features then are used by the RPN to generate the proposals. At the end, the classifier of the Faster R-CNN classifies the proposed regions by the RPN network to generate the final detections of the pedestrians.

To build the Faster R-CNN, first the VGG-16 classification network pretrained on IMAGENET is used since it is utilized in almost all the reported pedestrian detectors in the literature. However, with the current hardware and software setup, there was no enough memory to perform the finetuning for the Faster R-CNN despite the use of 12 GB eGPU and very small mini-batch size of 8. The reason behind this is not clear but one of the reasons might be due to the way MATLAB utilize the memory of the GPU during the training process since other frameworks such as Caffe requires a 12 GB GPU memory to train a Faster R-CNN network based on VGG-16 [55]. Therefore, the next shallower classification network pretrained on IMAGENET and available on MATLAB was used to create the Faster R-CNN pedestrian detector. AlexNet was used and the training process was not interrupted this time. The use of AlexNet should not affect the objectives of this work since all the CNNs work in a similar fashion and the outcomes found using any of them shall have similar impact on the others. Yet, the results of this work might not be competing with the state-of-the-art solutions since a shallow network is used to extract the features.

3.4.1 Training the Faster R-CNN Pedestrian Detector

To create a Faster R-CNN network in MALTAB using a pretrained classification CNN, transfer learning was used. Transfer learning allows the user to reuse the required part of the CNN without the need to retrain the whole network. In this work, the aim is to take the AlexNet CNN shown in Table 3.1 and use transfer learning to reuse the pretrained convolutional layers while removing the FC layers to create a Faster R-CNN pedestrian detector. The reason is that the convolutional layers are trained to extract convolutional features, visual features in this case, where the FC layers work as a classifier. However, the pretrained AlexNet FC layers are designed to classify the images into 1000 objects where in this work the aim is to perform classification into two objects only (pedestrian, not a pedestrian). Therefore, Layers 17 to 25 have been removed.

Afterwards, new layers have been added to the network instead of the layers that have been removed. Therefore, a new modified AlexNet has been created to perform pedestrian detection. The modified AlexNet architecture is shown in Table 3.2. As shown in the table, the output size of the FC layers ‘fc7’ (layer no. 20) and ‘fc8’ (layer no. 23) have been adjusted to 3072 instead of 4096. The reason behind this is to reduce the size of the minimum anchor box of the RPN. In case the size has been reduced further (e.g. 2024), the performance of the detector was affected. However, in case the 4096 was used, several pedestrians were missed and not detected. Thus, the use of 3072 was the sweet spot. Furthermore, an additional FC layer was added ‘fc9’ (layer no. 26). In any classification network, the size of the last FC layer should be equal to the number of classes. Since, the purpose of this network is to detect pedestrians, then, there are two classes ‘Pedestrian’, and ‘Background’. However, instead of jumping from a 3072 FC layer ‘fc7’ (layer no. 20) to a 2 FC layer (layer no. 26), an intermediate 1000 FC layer (layer no. 23) was introduced. Therefore, the total number of Layers in the modified AlexNet is 28.

As the architecture of the classification network is prepared, the MATLAB function trainFasterRCNNObjectDetector [67] was used to train the Faster R-CNN pedestrian detector. The training parameters reported in [55] were used when applicable. The RPN anchor boxes have been set to a single aspect ratio since the network performs single class detection. The scale and the number of the anchor boxes, and the size of the minimum anchor box were [55]selected by MATLAB after statistically analyzing the ground truth labels of the training dataset. This improved the detection performance on the Caltech dataset when compared to the case where these parameters have been selected manually. Yet, the size of the minimum anchor box was larger in this case when compared with [55] due to the different network architecture. To finetune the network and modify the architecture as stated earlier, the Caltech pedestrian dataset has been used. The dataset consists of around 250,000 images divided into 11 sets. In the training process, the images of the first 6 sets are used as suggested by [20]. The images are resized so that the shorter edge has 720 pixels and the number of strongest regions is set to 1000 proposals similar to the work in [55]. It is worth noting that the learning rate of the new FC layers 'fc6' to 'fc9' has been increased when compared to the rest of the layers. The main reason is that the rest of the layers were pretrained on IMAGENET where the new layers were not trained. Therefore, their learning rate has been increased as suggested by MATLAB to be 20 times the pretrained layers [67]. Afterwards, the training of the Faster R-CNN pedestrian detector started and it took around 28 hours to complete where the eGPU was used during the training process.

Table 3.1 The Architecture of MATLAB's AlexNet Classification Network Pretrained on IMAGENET

Layer No.	Layer Name	Layer Type	Comments
1	'data' Image Input	Input Layer	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1]
11	'relu2'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	Softmax
25	'output'	Classification Output	Cossentropyex with 'tench', 'goldfish', and 998 other classes

Table 3.2 The Architecture of the Modified AlexNet Classification Network to Perform Pedestrian Detection

Layer No.	**Layer Name**	**Layer Type**	**Comments**
1	'data' Image Input	Input Layer	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1]
11	'relu2'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0]
17	'fc6'	Fully Connected	3072 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	3072 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'relu8'	ReLU	ReLU
25	'drop8'	Dropout	50% dropout
26	'fc9'	Fully Connected	2 fully connected layer
27	'prob'	Softmax	Softmax
28	'output'	Classification Output	Cossentropyex

CHAPTER 4: RESULTS AND DISCUSSION

In this chapter, the findings and results acquired are discussed and reported. Section 4.1 reports the reflections on the setup used during the development and implementation of this work. Section 4.2 reports the results of this work in terms of processing time and performance metrics where Sections 4.3 discusses the reported results and the conclusions of this work.

4.1 *Reflections on the Hardware and Software Setup*

In this work, the setup shown in Figure 3.3 has been used where there are some reflections to be noted based on the implementation of the proposed model.

4.1.1 Reflections on the Use of MATLAB

In this work, MATLAB was used to design the motion flow algorithm, train the Faster R-CNN pedestrian detector, and integrate them to implement the two motion integrated detectors. Despite that MATLAB has the following two toolboxes to create and train CNN using GPUs: Neural Network Toolbox [68] and Parallel Computing Toolbox [69], it is worth noting the following observations:

- During the training process of CNN object detectors such as Faster R-CNN, the use of a validation set of images during the training process is not viable. Hence, to identify if the implemented CNN architecture is promising or overfitting, it is required to complete the training process which might take from several hours to several days depending on the architecture of the network, size of the training dataset, and the GPU used to execute the training process.

- During the training process on MATLAB, the program was using the Central Processing Unit (CPU) to perform some preparation steps in the training process despite selecting the GPU as an execution environment. These preparation steps include generating the proposals using the trained RPN before training the Fast R-CNN network. Subsequently, the training time increased by several days or weeks depending on the size of the training set. To train a Faster R-CNN based on AlexNet using 100 images, it took 2 hours to complete the training using the CPU to generate the proposals. This issue was noticed while using MATLAB R2018a. However, running the same code on MATLAB R2017a completed the training process in 10 minutes only using the GPU in all the stages.

Thus, it is recommended to use a framework specialized in deep learning and CNN such as Caffe, TensorFlow, Torch; especially since they provide advanced tools to control the training process, the network and the work in general. In contrast, MATLAB would be easier to use than the other environments for initial testing and as a learning stage about CNNs.

4.1.2 Reflections on the Use of eGPU

To train the Faster R-CNN pedestrian detector, the Nvidia TITAN Xp was used as an eGPU. This was used since its specifications (e.g. memory, memory bandwidth, etc.) is better than the internal GPU of the Dell XPS laptop. Thus, it should perform training for larger network architectures and at a faster rate as well. Yet, the performance will not be identical to a regular desktop computer that utilizes the TITAN Xp as a GPU. The drop in performance reported is in the range from 15% to 20% due to the use of Thunderbolt 3 as a connection [70]. In addition, an additional drop of 10% to 15% was noticed since the eGPU is connected through a 20 Gbps Thunderbolt 3 connection. However, the performance would still be better than the internal GPU since the TITAN Xp memory is 12 GB. In addition, these numbers were reported when the eGPU is used for Gaming and not performing computations such as in this case. Thus, the exact performance drop might be slightly different.

During the training process of the CNNs, it was noticed that the preparation stages take longer time when the training was executed on the eGPU. This is due to the data transfer through the Thunderbolt connection. However, the training process time was much shorter. Therefore, for large training datasets, it is recommended to perform the training process using the eGPU as the total time would be less. For testing, the internal GPU was used, since the number of frames used in testing is small. It was found that it takes 1.4 seconds (~0.71 FPS) to process one frame using the eGPU where the internal GPU process the same frame in ~0.11 seconds (~9 FPS), which reflects that internal GPU is faster by a factor of ~12.7.

4.2 *Results*

After preparing the proposed motion flow and CNN based pedestrian detectors, their implementations are integrated as illustrated in Figure 3.1 and Figure 3.2 to create the two proposed motion saliency aided Faster R-CNN based pedestrian detectors. To assess these algorithms, the metrics used in evaluation are precision, error rates and AP. The precision, Equation 2.5, provides the percentage of the correct detections out of the total detections of the detector. The error rates, Equations 2.8 and 2.10, quantify the ability of the proposed detectors to reduce false detections where the AP quantifies how the detection ability of the original detector is affected. In other words, if the number of false positives has been reduced while reducing the detection ability, the proposed algorithms are said to have drawbacks. For example, having a detector with zero detections would have a zero error but its detection ability is still zero. Therefore, it is very essential to examine the detection ability after integrating the motion information.

4.2.1 Processing Time

After implementing the algorithms, the processing time has been analyzed and reported in Table 4.1 for a frame resolution of 960×720. The reported timings are based on implementing the algorithms on the Dell XPS 9560 without the use of the external GPU since it takes longer time as discussed in Section 4.1.2. The motion flow algorithm implementation was processed on the CPU. However, the CNN pedestrian detection performs the processing using the Nvidia GTX 1050. In addition, the table shows the processing time for some other algorithms from the literature where these timings are based on CPU implementations. The MATLAB implementations of the ACF and the LDCF detectors are provided by the authors in [71]. However, the implementation of the HOG features based pedestrian detector is provided by MATLAB [72].

From the table, proposed motion information integrated algorithms dropped the speed by approximately 1 FPS from the Faster R-CNN. The difference between the two motion based methods is how motion information is integrated. The MM Faster R-CNN approach uses the motion to mask the image before processing it through the Faster R-CNN detector. The MC Faster R-CNN approach uses the motion information after processing the image through the Faster R-CNN detector to correct the detections. Thus, it performs lower number of computations which made it faster.

Table 4.1 Comparison Between the Processing Time for Different Pedestrian Detectors

Pedestrian Detectors	**Algorithm (Execution Environment)**	**Algorithm Average Processing Time [Sec.]**	**Total Processing Time [Sec.]**	**Processing Speed [FPS]**
Faster R-CNN [Proposed]	Faster R-CNN (GPU)	-	0.10865	9.2
MM Faster R-CNN [Proposed]	Motion Flow (CPU)	0.01512	0.12592	8.1
	Faster R-CNN (GPU)	0.10865		
MC Faster R-CNN [Proposed]	Motion Flow (CPU)	0.01329	0.12194	8.2
	Faster R-CNN (GPU)	0.10865		
HOG [30]	HOG (CPU)	-	0.18616	5.3
ACF [41]	ACF (CPU)	-	0.21859	4.5
LDCF [44]	LDCF (CPU)	-	1.18296	0.8

The reported processing time per frame for the pedestrian detectors in the table ranges from around 0.1 to 1.2 seconds. The slowest detector is the CPU implementation of the LDCF detector where the processing time reached 1.18 seconds due to the additional decorrelation filtering stage added on top of the ACF detector, which increased the processing time of the original ACF detector by approximately 1 second. On the other hand, the processing speed of the HOG features and the ACF detectors is very similar where both process one frame in ~0.2 seconds on CPU. The implementation of the Faster R-CNN detector using the GPU caused the Faster R-CNN based pedestrian detector to be the fastest detector. By comparing these timings, the processing speed has been increased by up to 10 times from the processing time of the LDCF detector; which has the slowest implementation.

4.2.2 Video 1 Testing Results

The first video used to test the proposed algorithms is taken from the publicly available PETS2009 dataset. The video sequence is the S2_L1_12-34 view 1 which contains a low density sparse crowd with variability in crowd occlusions. The original video resolution is 768×576 where it is processed at the resized resolution of 960×720. In addition, the video was manually annotated to create the ground truth bounding boxes to compute the evaluation metrics.

Figure 4.1 depicts the results of detecting the pedestrians using the three different algorithms: Faster R-CNN, MM Faster R-CNN, and MC Faster R-CNN. The green bounding boxes represent the detections of the algorithms, the red bounding boxes corresponds to discarded detections through motion correction, and blue rectangular regions represents motion detected regions.

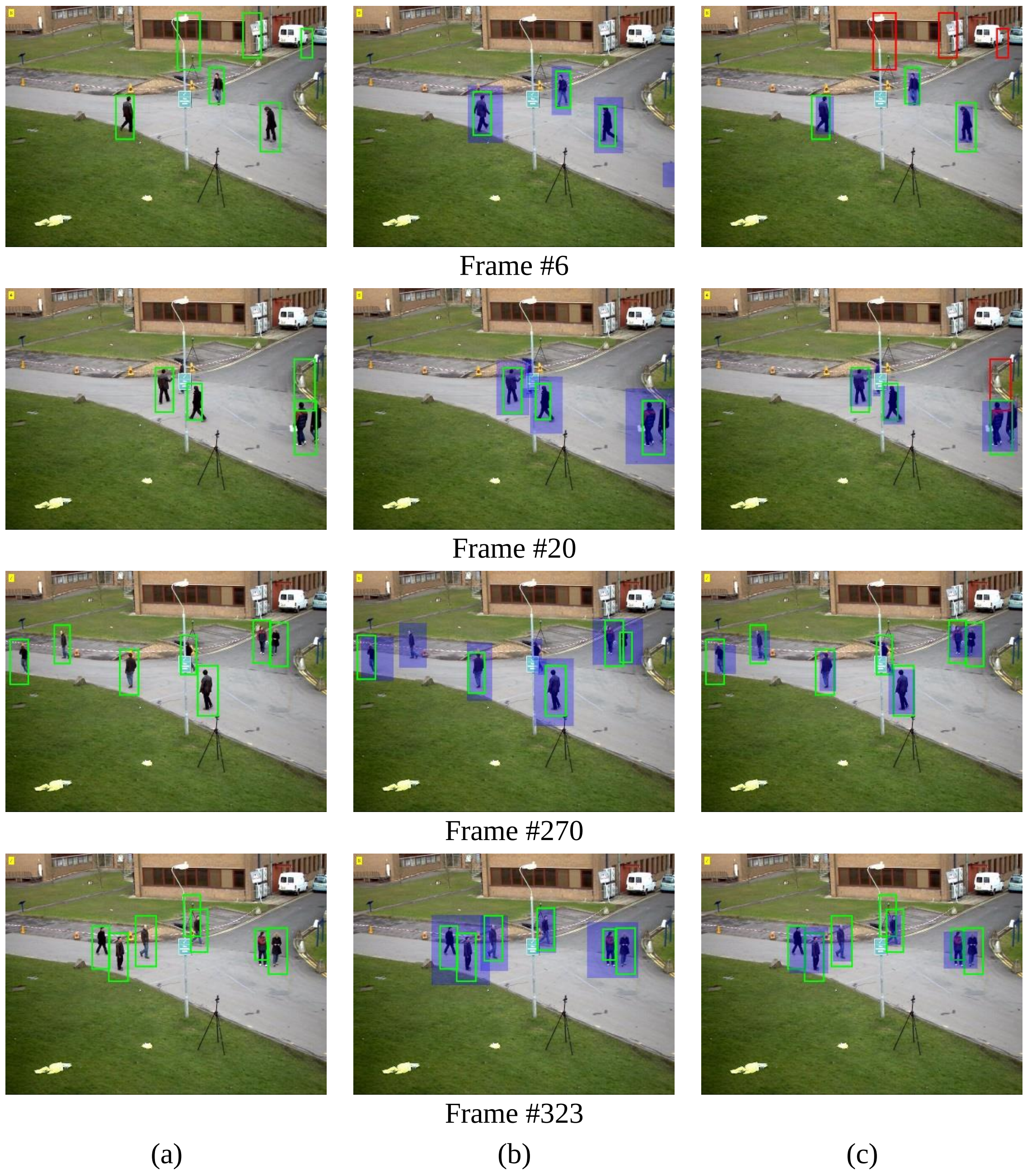


Figure 4.1. Video 1 detection samples using (a) Faster R-CNN pedestrian detector, (b) MM Faster R-CNN pedestrian detector, and (c) MC Faster R-CNN pedestrian detector

From the figure, frame #6 and frame #20 clearly demonstrate two examples of how the integrated motion information improved the detection in the two proposed approaches. However, frame #270 demonstrates an example where the MM Faster R-CNN detections degraded the performance while the MC Faster R-CNN maintained the same performance of the Faster R-CNN detector. On the contrary, frame #323 illustrates an example where the MM Faster R-CNN outperforms the Faster R-CNN and the MC Faster R-CNN detectors.

In addition, Table 4.2 summarizes the evaluation metrics for the proposed algorithms and three algorithms from the literature. The AP metric reported in the table has been extracted from the precision/recall curves plotted in Figure 4.2 where the IoU threshold is 0.5.

Table 4.2 Comparison on Video 1 (PETS2009 S2L1 View 1) Showing the Precision, Error Rates and AP for Different Pedestrian Detectors

Pedestrian Detectors	**Precision [%]**	**MRE [%]**	**MSE**	**AP [%]**
Faster R-CNN [Proposed]	74.13	25.09	3.52	69.84
MM Faster R-CNN [Proposed]	84.62	14.93	1.27	69.50
MC Faster R-CNN [Proposed]	81.25	18.63	1.89	70.74
HOG [30]	66.38	32.70	7.03	74.83
ACF [41]	88.85	9.83	1.09	91.30
LDCF [44]	89.61	9.09	1.11	90.72

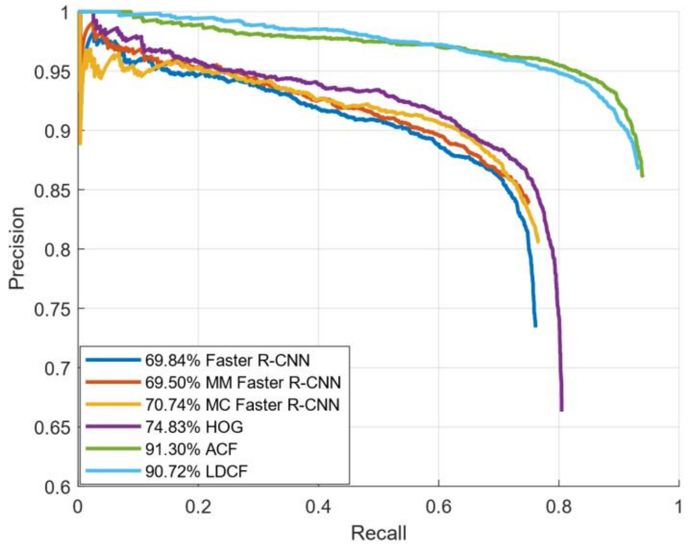


Figure 4.2. Comparison on Video 1 (PETS2009 S2L1 View 1) showing the AP for different pedestrian detectors using an IoU threshold of 0.5 to determine true positive

4.2.3 Video 2 Testing Results

Similar to the first video, the second video used to test the proposed algorithms is taken from the publicly available PETS2009 dataset. The video sequence is the S3_MF_12-34 view 2 which contains a low density crowd with variability in occlusions. The original video resolution is 768×576 where it is processed at the resized resolution 960×720.

Figure 4.3 depicts the results of detecting the pedestrians using the three different algorithms: Faster R-CNN, MM Faster R-CNN, and MC Faster R-CNN. The green bounding boxes represent the detections of the algorithms, the red bounding boxes corresponds to discarded detections through motion correction, and blue rectangular regions represents motion detected regions.

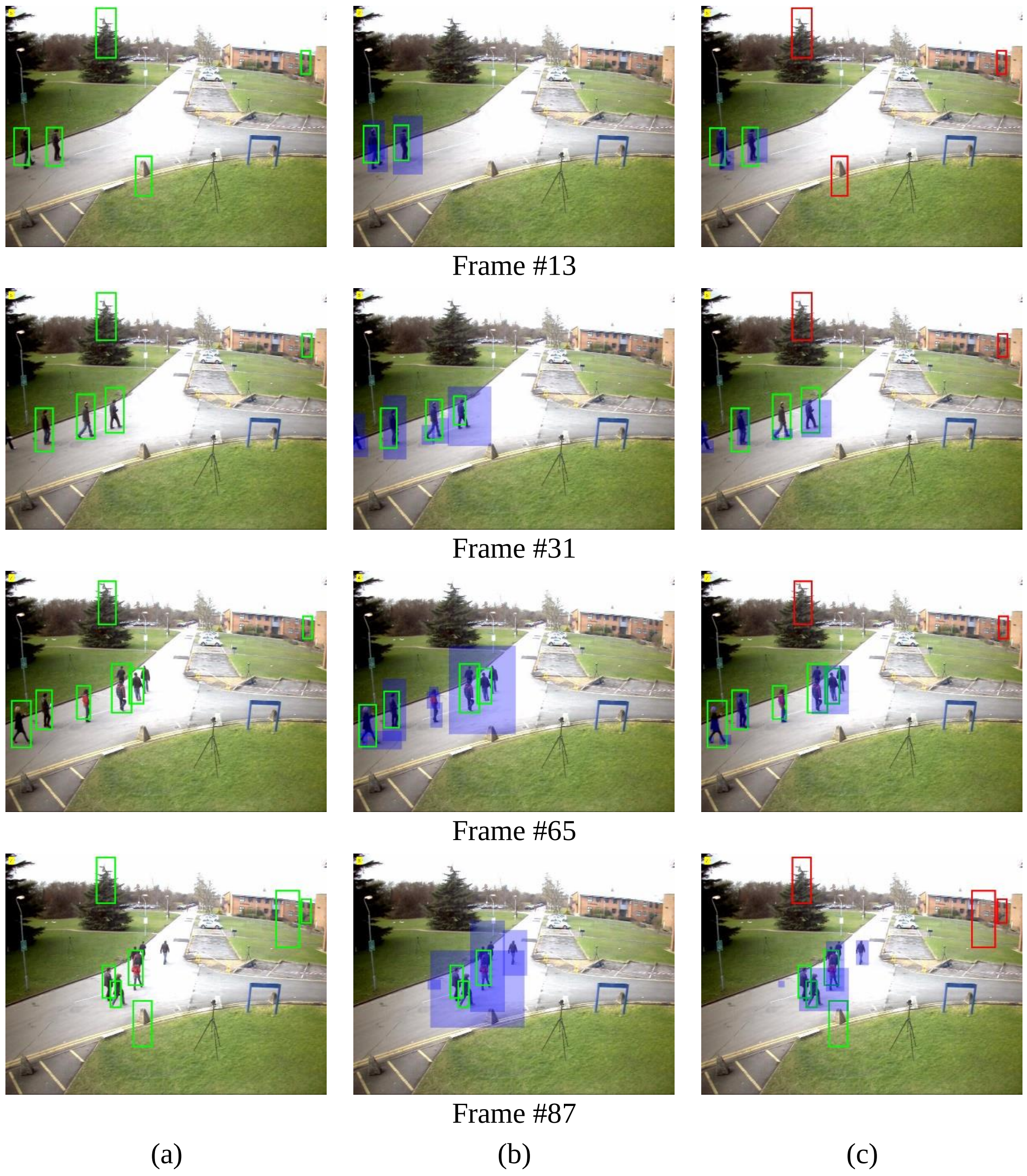


Figure 4.3. Video 2 detection samples using (a) Faster R-CNN pedestrian detector, (b) MM Faster R-CNN pedestrian detector, and (c) MC Faster R-CNN pedestrian detector

From the figure, frame #13 and frame #31 clearly demonstrate two examples of how the integrated motion information improved the detection in the two proposed approaches. However, frame #65 demonstrates an example where the MM Faster R-CNN detections has been affected. One of the pedestrians that is detected by the Faster R-CNN and MC Faster R-CNN detectors has not been detected. Additionally, the MM Faster R-CNN was able to correct one false detection in this frame. In contrast, frame #87 illustrates an example where the MM Faster R-CNN outperforms the Faster R-CNN and the MC Faster R-CNN detectors.

Table 4.3 summarizes the evaluation metrics for the proposed algorithms and three algorithms from the literature. The AP metric reported in the table has been extracted from the precision/recall curves plotted in Figure 4.2.

Table 4.3 Comparison on Video 2 (PETS2009 S3MF View 2) Showing the Precision, Error Rates and AP for Different Pedestrian Detectors

Pedestrian Detectors	**Precision [%]**	**MRE [%]**	**MSE**	**AP [%]**
Faster R-CNN [Proposed]	20.00	61.64	14.44	46.49
MM Faster R-CNN [Proposed]	50.00	27.22	1.31	49.76
MC Faster R-CNN [Proposed]	50.00	26.99	1.12	49.17
HOG [30]	20.00	59.59	18.60	65.39
ACF [41]	66.67	12.14	0.81	89.90
LDCF [44]	60.00	10.87	0.85	85.60

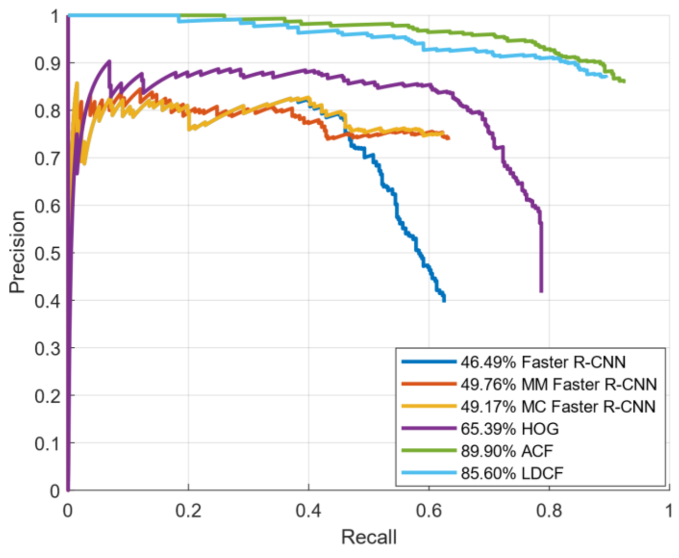


Figure 4.4. Comparison on Video 2 (PETS2009 S3MF View 2) showing the AP for different pedestrian detectors using an IoU threshold of 0.5 to determine true positive

4.3 *Discussion of Results*

The reported results for videos 1 and 2 in Table 4.2 and Table 4.3 clearly demonstrates that the motion saliency integration to the Faster R-CNN pedestrian detector improved the ability of the detector to identify false detections. Furthermore, this integration did not affect the ability of the Faster R-CNN detector to detect pedestrian as reflected by the values of the AP. On the contrary, the AP values has been improved slightly due to the reduction in the number of false positives.

For the MM Faster R-CNN, the MRE has been reduced by approximately 10% and 34% for the first and second videos, respectively. Meanwhile, the precision of the detector has been increased from 74% to 84% and from 20% to 50% demonstrating further improvements for video 1 and 2, respectively. This shows that MM Faster R-CNN is now detecting more true positive detections out of the total number of detections; which means the number of false detections has been reduced since the AP values was close to the original detector's AP.

Similarly, the MC Faster R-CNN detector demonstrated similar performance where the MRE has been reduced by approximately 6% and 35% for the first and second videos, respectively. In this case, the precision and the AP on the two videos has been improved emphasizing on the fact that the MC Faster R-CNN detector results have been improved when compared with the original Faster R-CNN detector.

Therefore, integrating the motion information with the Faster R-CNN detector either before or after detecting the pedestrians using the Faster R-CNN detector, improves the ability of the network to reduce false positives. However, the only drawback of this approach is the increase in the processing time due to the motion information extraction. Yet, implementing such an algorithm affects the speed of the detector slightly as the motion integration in this case increased the processing time by less than 20 ms. Therefore, it is recommended to integrate a CNN based pedestrian detector with motion flow algorithm to enhance the performance since the processing time of the network is larger than the motion flow algorithm.

By comparing the proposed motion integrated pedestrian detectors with other detectors from the literature, it is noted that the main difference is related to the AP reported. The main reason for the AP of the Faster R-CNN, MM Faster R-CNN, and MC Faster R-CNN detectors to be the lowest is due to the selection of AlexNet. AlexNet is a shallow network which consists of 5 convolutional layers and 3 FC layers. When the network is used in a Faster R-CNN architecture, it resulted in having anchor boxes of larger size when compared to the pedestrians in the surveillance videos used in testing. As a result, this led to the following two cases: First, the Faster R-CNN detector would be detecting the pedestrians with large bounding boxes that have low IoU with ground truth bounding boxes. Second, the network would miss the detection of the pedestrians in case the size of the anchors is way larger than the size of the pedestrians. This could be verified by varying the IoU threshold to determine true positive detections and computing the AP at these different thresholds. In case the AP increased slightly, then the Faster R-CNN detector is not detecting the pedestrians; however, if the increase in AP was notable, then the Faster R-CNN is detecting pedestrians with large bounding boxes. Table 4.4 reports the AP of different algorithms while changing the IoU threshold to determine true positive detections.

As demonstrated in the table, the AP of the three detectors that depends on the Faster R-CNN proposed in this work has been increased by approximately 5% - 10% when the IoU threshold is reduced by 5% every time for the two videos. Thus, this demonstrates that the size of the bounding boxes of the detections is quite large and when the AP is computed at an IoU threshold of 0.5, many of these detections are considered as false detections. Figure 4.5 shows examples of large bounding boxes while detecting the pedestrians. Similarly, the HOG features based pedestrian detector AP increases when the threshold of true positive detections decreases which leads to a similar conclusion. However, this is not the case for the ACF and LDCF detectors where their AP increased by approximately 1% for every reduction by 5% in the threshold. In fact, this is anticipated since both detectors depend on pyramid of features which causes their results to be more accurate when compared with the rest of the detectors.

Table 4.4 AP of the Different Algorithms While Varying the IoU Threshold to Determine True Positive Detections

Pedestrian Detectors	**AP [%] on Video 1 (PETS2009 S2L1 View 1)**			**AP [%] on Video 2 (PETS2009 S3MF View 2)**		
	@IoU=0.5	**@IoU=0.45**	**@IoU=0.40**	**@IoU=0.5**	**@IoU=0.45**	**@IoU=0.40**
Faster R-CNN [Proposed]	69.84	78.73	83.98	46.49	57.50	69.22
MM Faster R-CNN [Proposed]	69.50	77.40	81.67	49.76	61.84	71.59
MC Faster R-CNN [Proposed]	70.74	78.80	84.61	49.17	60.71	72.92
HOG [30]	74.83	88.19	88.29	65.39	78.29	83.65
ACF [41]	91.30	92.62	93.57	89.90	91.08	91.63
LDCF [44]	90.72	92.12	92.93	85.60	87.16	87.98

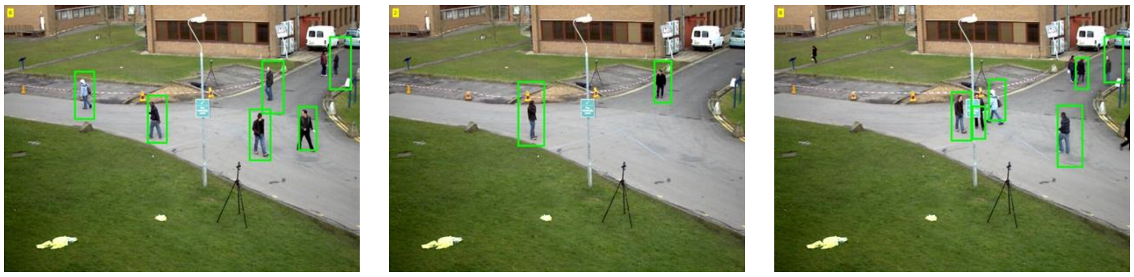


Figure 4.5. Examples of pedestrian detections using the proposed Faster R-CNN where the size of the bounding boxes is larger than the pedestrians

Therefore, based on the reported results and design, the following summarizes the constraints of the proposed motion saliency aided CNN pedestrian detectors: First, since the detectors are utilizing a motion flow algorithm that is based on background subtraction, the stationary camera became one of the constraints of the algorithm. Additionally, the proposed detectors are designed to detect pedestrians of low density crowds. The processing speed reported is slightly higher than 8 FPS; which means that the proposed detectors process videos in real time in case their frame rate dropped to lower than 8 FPS such as the PETS 2009 dataset scenarios used for testing.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 *Conclusions*

Pedestrian and crowd analysis consists of three main stages: crowd modelling, crowd monitoring and crowd management. Each of these stages tackles specific properties of the crowd to ensure its safety and security. The scope of this work is related to the crowd modelling stage where it concentrates on pedestrian detection of low density crowds. This work proposes to integrate the motion saliency with the CNN based pedestrian detector to enhance the performance of the pedestrian detector used in surveillance applications through reducing the false detections.

To develop the proposed model, background subtraction based on GMM has been implemented as a motion flow algorithm. It was selected due to its simplicity, reliable performance and being widely used in the literature in similar applications. Faster R-CNN has been selected as a CNN pedestrian detector since it resembles the state-of-the-are solution for object detection problems. Faster R-CNN pedestrian detector has been implemented based on a pretrained AlexNet CNN on IMAGENET dataset.

The integration of the motion information was performed at two different stages resulting in two proposed detectors: MM Faster R-CNN and MC Faster R-CNN. MM Faster R-CNN detector masks the images based on the extracted motion information and then process the masked image using the Faster R-CNN detector. MC Faster R-CNN detector processes the image using the Faster R-CNN detector and then utilizes the motion information to correct the detections. Two videos from the PETS2009 dataset were used to test the two proposed motion integrated pedestrian detectors.

Results have demonstrated that the number of false positive detections have been reduced drastically. For MM Faster R-CNN, the MRE is reduced by 10% and 34% for the two videos while the precision has been increased to reach 84% and 50% compared to 74% and 20% for the Faster R-CNN detector without motion integration. MM Faster R-CNN maintained the AP of the Faster R-CNN detector without motion integration for the first video while it was increased by 3% for the second video. Similar performance was observed for MC Faster R-CNN detector where the MRE has been reduced by 6% and 35% for the first and second videos, respectively. The precision has been increased to reach 81% and 50% where the AP was improved by 1% and 3% for the first and second videos, respectively. The integration of the motion flow algorithm decreased the processing speed of the two proposed detectors reaching ~8 FPS instead of ~9 FPS for the original Faster R-CNN detector when executed on Nvidia GTX 1050 GPU.

5.2 *Future Work*

The research results of this thesis show a potential in improving the CNN pedestrian detector performance through integrating motion saliency. The use of a deeper network architecture instead of AlexNet would further improve the performance and allowing for smaller anchor boxes. Implementing a deeper network on a deep learning framework through using a high end GPU or cluster of GPUs is necessary to accurately perform crowd modelling and to compete with current state-of-the-art results in this area. In addition, this would allow the detector to better perform in higher density crowds.

The motion flow algorithm performance can be further improved through analyzing the detected motion regions through joining nearby small regions or ignoring the small regions if they were apart.

Extending the scope of the work to include people tracking, counting and density estimation is the next phase of this work.

REFERENCES

[1] E. László, P. Szolgay, and Z. Nagy, "Analysis of a GPU based CNN implementation," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, 2012, pp. 1-5.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. 115, pp. 211-252, 2015/12/01 2015.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision,* vol. 88, pp. 303-338, June 01 2010.

[4] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, *et al.*, "Large-scale image classification: Fast feature extraction and SVM training," in *CVPR 2011*, 2011, pp. 1689-1696.

[5] J. Sanchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *CVPR 2011*, 2011, pp. 1665-1672.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25,* pp. 1097-1105, 2012.

[7] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ed Cham: Springer International Publishing, 2014, pp. 818-833.

[8] C. Szegedy, L. Wei, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1-9.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[10] IMAGENET, "IMAGENET Large Scale Visual Recognition Challenge 2016 (ILSVRC2016)," 2016.

[11] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *CoRR*, vol. abs/1709.01507, 2017.

[12] M. S. Zitouni, H. Bhaskar, J. Dias, and M. E. Al-Mualla, "Advances and trends in visual crowd analysis: A systematic survey and evaluation of crowd modelling techniques," *Neurocomputing*, vol. 186, pp. 139-159, 2016/04/19/ 2016.

[13] A. Dehghan and M. Shah, "Binary Quadratic Programing for Online Tracking of Hundreds of People in Extremely Crowded Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1-1, 2017.

[14] S. Yao, S. Pan, T. Wang, C. Zheng, W. Shen, and Y. Chong, "A new pedestrian detection method based on combined HOG and LSS features," *Neurocomputing*, vol. 151, pp. 1006-1014, 2015/03/03/ 2015.

[15] V.-D. Hoang and K.-H. Jo, "Joint components based pedestrian detection in crowded scenes using extended feature descriptors," *Neurocomputing*, vol. 188, pp. 139-150, 2016/05/05/ 2016.

[16] J. C. S. J. Junior, S. R. Musse, and C. R. Jung, "Crowd Analysis Using Computer Vision Techniques," *IEEE Signal Processing Magazine,* vol. 27, pp. 66-77, 2010.

[17] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed Haar-Like Features Improve Pedestrian Detection," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 947-954.

[18] J. Mao, T. Xiao, Y. Jiang, and Z. Cao, "What Can Help Pedestrian Detection?," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6034-6043.

[19] C. Li, X. Wang, and W. Liu, "Neural features for pedestrian detection," *Neurocomputing,* vol. 238, pp. 420-432, 2017/05/17/ 2017.

[20] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, pp. 743-761, 2012.

[21] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters,* vol. 27, pp. 861-874, 2006/06/01/ 2006.

[22] M. S. Zitouni, J. Dias, M. Al-Mualla, and H. Bhaskar, "Hierarchical Crowd Detection and Representation for Big Data Analytics in Visual Surveillance," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 1827-1832.

[23] T. Bouwmans, C. Silva, C. Marghes, M. S. Zitouni, H. Bhaskar, and C. Frelicot, "On the role and the importance of features for background modeling and foreground detection," *Computer Science Review,* vol. 28, pp. 26-91, 2018/05/01/ 2018.

[24] D. Lin, E. Grimson, and J. Fisher, "Modeling and estimating persistent motion with geometric flows," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2010, pp. 1-8.

[25] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A Survey on Object Detection and Tracking Methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, 2014.

[26] S. Manchanda and S. Sharma, "Analysis of computer vision based techniques for motion detection," in *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence),* 2016, pp. 445-450.

[27] A. Agarwal, S. Gupta, and D. K. Singh, "Review of optical flow technique for moving object detection," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016, pp. 409-413.

[28] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1-4.

[29] S. Bauer, S. Köhler, K. Doll, and U. Brunsmann, "FPGA-GPU architecture for kernel SVM pedestrian detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* - *Workshops*, 2010, pp. 61-68.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1.

[31] T. Zhou, J. Yang, A. Loza, H. Bhaskar, and M. Al-Mualla, "Crowd modeling framework using fast head detection and shape-aware matching," 2015, p. 22.

[32] J. Ferryman and A. Shahrokni, "PETS2009: Dataset and challenge," in *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009, pp. 1-6.

[33] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," *Proceedings of the British Machine Conference,* pp. 91.1-91.11, 2009.

[34] P. Dollar, S. Belongie, and P. Perona., "The Fastest Pedestrian Detector in the West," *Proceedings of the British Machine Vision Conference,* pp. 68.1--68.11, 2010.

[35] M. Rodriguez, I. Laptev, J. Sivic, and J. Y. Audibert, "Density-aware person detection and tracking in crowds," in *2011 International Conference on Computer Vision*, 2011, pp. 2423-2430.

[36] V. Lempitsky and A. Zisserman, "Learning To Count Objects in Images," *Advances in Neural Information Processing Systems,* pp. 1324--1332, 2010.

[37] J. Yan, Z. Lei, D. Yi, and S. Z. Li, "Multi-pedestrian detection in crowded scenes: A global view," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3124-3129.

[38] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627-1645, 2010.

[39] V. Rabaud and S. Belongie, "Counting Crowded Moving Objects," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, pp. 705-711.

[40] W. Ge, R. T. Collins, and R. B. Ruback, "Vision-Based Analysis of Small Groups in Pedestrian Crowds," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, pp. 1003-1016, 2012.

[41] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast Feature Pyramids for Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, pp. 1532-1545, 2014.

[42] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 794-801.

[43] A. Ess, B. Leibe, and L. V. Gool, "Depth and Appearance for Mobile Scene Analysis," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1-8.

[44] W. Nam, P. Dollar, and J. H. Han, "Local Decorrelation for Improved Pedestrian Detection," *Advances in Neural Information Processing Systems,* pp. 424-432, 2014.

[45] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht, "On Oblique Random Forests," Berlin, Heidelberg, 2011, pp. 453-469.

[46] B. Hariharan, J. Malik, and D. Ramanan, "Discriminative Decorrelation for Clustering and Classification," Berlin, Heidelberg, 2012, pp. 459-472.

[47] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision,* vol. 111, pp. 98-136, January 01 2015.

[48] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, *et al.*, "Microsoft COCO: Common Objects in Context," Cham, 2014, pp. 740-755.

[49] G. Han, X. Zhang, and C. Li, "Revisiting Faster R-CNN: A Deeper Look at Region Proposal Network," Cham, 2017, pp. 14-24.

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580-587.

[51] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, pp. 154-171, September 01 2013.

[52] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440-1448.

[53] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[54] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137-1149, 2017.

[55] L. Zhang, L. Lin, X. Liang, and K. He, "Is Faster R-CNN Doing Well for Pedestrian Detection?," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., ed Cham: Springer International Publishing, 2016, pp. 443-457.

[56] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1-1, 2017.

[57] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian detection aided by deep learning semantic tasks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5079-5087.

[58] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning Complexity-Aware Cascades for Deep Pedestrian Detection," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3361-3369.

[59] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, pp. 337-407, 2000/04 2000.

[60] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354-3361.

[61] S. Xie and Z. Tu, "Holistically-Nested Edge Detection," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1395-1403.

[62] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213-3223.

[63] G. Brazil, X. Yin, and X. Liu, "Illuminating Pedestrians via Simultaneous Detection and Segmentation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4960-4969.

[64] Dell. (Accessed on October 2017). *XPS 15*. Available: www.dell.com/en-us/shop/dell-laptops/xps-15/spd/xps-15-9560-laptop

[65] Nvidia. (Accessed on April 2018). *GeForce Laptops*. Available: https://www.nvidia.com/en-us/geforce/products/10series/laptops/

[66] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection," in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, Eds., ed Boston, MA: Springer US, 2002, pp. 135-144.

[67] MathWorks. (2017, Accessed on February 2018). *trainFasterRCNNObjectDetector*. Available: https://www.mathworks.com/help/vision/ref/trainfasterrcnnobjectdetector.html

[68] MathWorks. (Accessed on February 2018). *Create, train, and simulate shallow and deep learning neural networks*. Available: https://www.mathworks.com/products/neural-network.html

[69] MathWorks. (Accessed on February 2018). *Perform parallel computations on multicore computers, GPUs, and computer clusters*. Available: https://www.mathworks.com/products/parallel-computing.html

[70] eGPU.io. (Accessed on April 2018). *PCI Express vs. Thunderbolt - How much performance drop of your GPU you will have if you put it in eGPU*. Available: https://egpu.io/build-guides/#perf

[71] P. Dollar, C. Wojek, B. Schiele, and, and P. Perona. (2009, Accessed on November 2017). *Caltech Pedestrian Detection Benchmark*. Available: www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

[72] MathWorks. (2012, Accessed on March 2018). *Detect upright people using HOG features*. Available: https://www.mathworks.com/help/vision/ref/vision.peopledetector-system-object.html