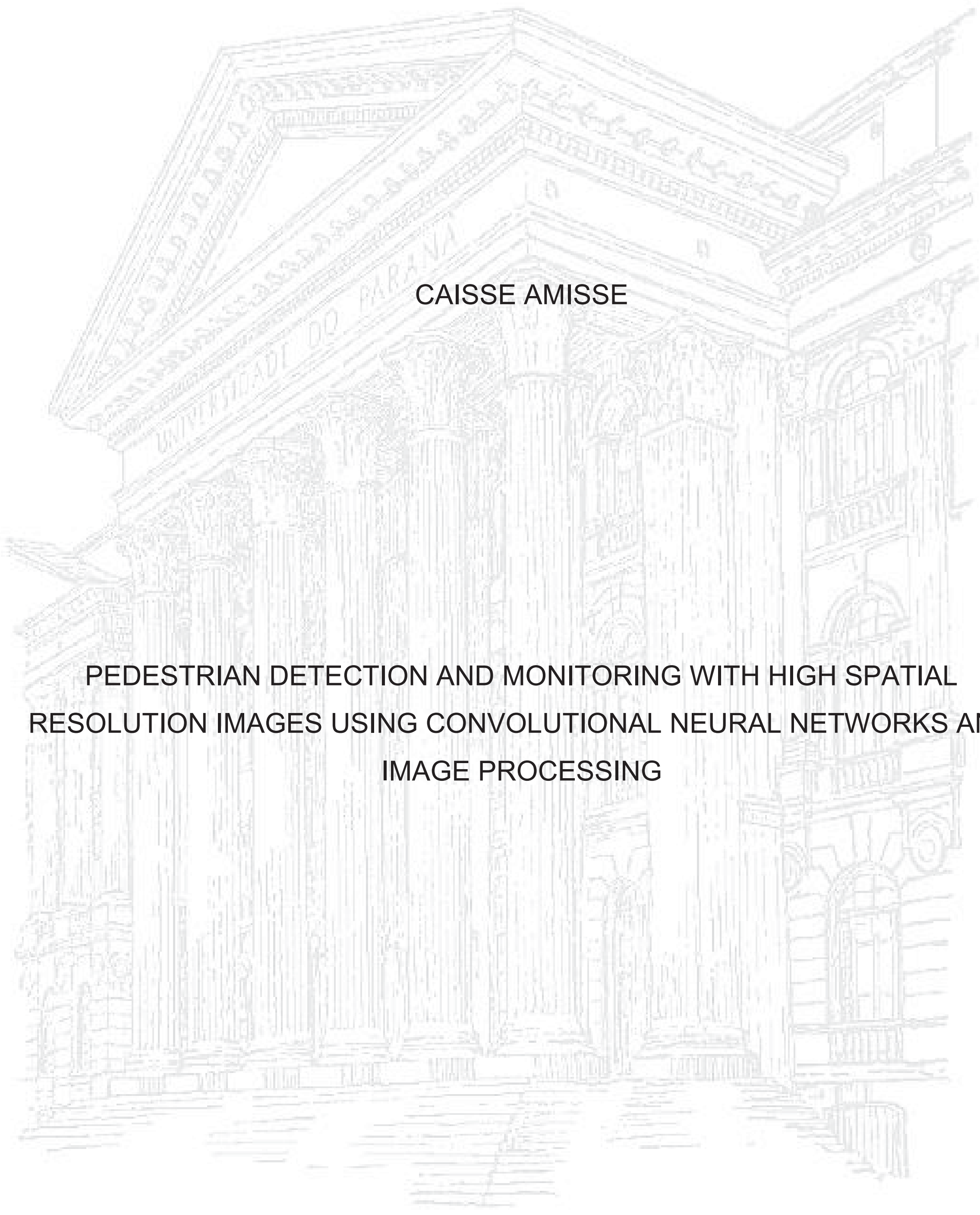
UNIVERSIDADE FEDERAL DO PARANÁ



CAISSE AMISSE

PEDESTRIAN DETECTION AND MONITORING WITH HIGH SPATIAL RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORKS AND IMAGE PROCESSING

CURITIBA

2020

CAISSE AMISSE

PEDESTRIAN DETECTION AND MONITORING WITH HIGH SPATIAL RESOLUTION IMAGES USING CONVOLUTIONAL NEURAL NETWORKS AND IMAGE PROCESSING

Tese apresentada ao Programa de Pós-Graduação em Ciências Geodésicas, Setor de Ciências da Terra da Universidade Federal do Paraná, como requisito parcial à obtenção do título de Doutor em Ciências Geodésicas.

Orientador: Prof. Dr. Jorge António Silva Centeno

CURITIBA
2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

A517p Amisse, Caisse
Pedestrian detection and monitoring with high spatial resolution images using convolutional neural networks and image processing [recurso eletrônico] / Caisse Amisse. – Curitiba, 2020.

Tese - Universidade Federal do Paraná, Setor de Ciências da Terra, Programa de Pós-Graduação em Ciências Geodésicas, 2020.

Orientador: Jorge António Silva Centeno.
.

1. Redes neurais (Computação). 2. Aprendizagem de máquina. I. Universidade Federal do Paraná. II. Centeno, Jorge António Silva. III. Título.

CDD: 006.31

Bibliotecária: Vanusa Maciel CRB- 9/1928

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS DA TERRA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO CIÊNCIAS
GEODÉSICAS - 40001016002P6

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em CIÊNCIAS GEODÉSICAS da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **CAISSE AMISSE** intitulada: **PEDESTRIAN DETECTION AND MONITORING WITH HIGH SPATIAL RESOLUTION IMAGES USING CONVOLUCIONAL NEURAL NETWORKS AND IMAGE PROCESSING**, sob orientação do Prof. Dr. JORGE ANTONIO SILVA CENTENO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 17 de Dezembro de 2020.

Assinatura Eletrônica
22/12/2020 15:18:52.0
JORGE ANTONIO SILVA CENTENO
Presidente da Banca Examinadora

Assinatura Eletrônica
23/12/2020 21:35:59.0
ALVARO MURIEL LIMA MACHADO
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
18/12/2020 13:27:13.0
HIDEO ARAKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
22/12/2020 13:51:06.0
JOÃO MARCELO BRASÃO PROTÁSIO
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARÁ)

Assinatura Eletrônica
22/12/2020 09:07:29.0
EDSON APARECIDO MITISHITA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Centro Politécnico - Caixa Postal 19001 - CURITIBA - Paraná - Brasil
CEP 81531-980 - Tel: (41) 3361-3153 - E-mail: cpgcg@ufpr.br
Documento assinado eletronicamente de acordo com o disposto na legislação federal Decreto 8539 de 08 de outubro de 2015.
Gerado e autenticado pelo SIGA-UFPR, com a seguinte identificação única: 66324
Para autenticar este documento/assinatura, acesse https://www.prppg.ufpr.br/siga/visitante/autenticacaoassinaturas.jsp e insira o codigo 66324

In memory of my Baba Sheikh Amisse Caisse

To my Mama Abiba Ali

With love and eternal appreciation

Acknowledgements

Alhamdulillah, I would like to start my words here by thanking Allah (SWT) for giving me this opportunity and the patience to complete this thesis. Peace and blessings be upon prophet Muhammad (PBUH), his family and his companions.

I am certain that this brief acknowledgement will not adequately express the depth of thanks and sincere gratitude owed to my supervisor Prof. Dr.-Ing. Jorge António Silva Centeno, for his guidance, unceasing encouragement and invaluable support throughout this research work. It has been an absolute privilege to work under his careful supervision and get benefited from his profound knowledge and expertise in Geosciences. This thesis has reached its present form thanks to his genuine interest, constructive suggestions, critical appraisal and contagious enthusiasm for research. Also, a special thanks go to Prof. Álvaro Muriel Lima Machado, and Prof. Hideo Araki for their constructive suggestions, ever available help and encouragement throughout the study.

I would also like to thank the PPGCG professors who had taught or advised me in one way or the other.

My sincere thanks also goes to CAPES for scholarships. Additionally, I would like to acknowledge the support I received from UniRovuma to attend my studies.

I also want to thank a lifetime of love and understanding from the most special women in my life Abiba Ali, Daua J. Adamo, Mery V.J. Muhove, Hawah, Dayarah, and Nusrah. Also, my special thanks to my lovely Lucman and Caisse, my brothers Azize, Amina, Atija, Fatima, Baquicha, Muamina, Rassul and Hamza. To my friends Abubacar, Abdulbassuar, Abdulssuamado, Mustafa, Assane, and Ivenso da S. V. Sualehe. Thank you all kindly for the support, understanding and words of encouragement from the beginning right until the end of this journey.

Finally, I would also like to humbly thank my colleagues, and fellow lab members at the PPGCG who have offered their guidance and support. Specially Mario E. J. Palma, Ilich H. C. Verástegui, Jaime C. Macuácua, Miguel R.Mugio, André F. Moiane, and Jairo Pinto.

Resumo

O rastreamento de pedestres é uma área de pesquisa bem estabelecido quando realizado em ambiente interno ou quando a câmera é estática. Mas quando a câmera está ativa, como no caso do ambiente externo, uma série de problemas e desafios emergem. Alguns desses desafios dizem respeito a: alterações de iluminação, oclusão, fundo desordenado e movimentos de pedestres.

Nesta tese, buscou-se combinar os métodos baseados em aprendizado profundo e técnicas de processamento de imagens para detectar e rastrear pedestres a partir de imagens de alta resolução espacial obtidas em um ambiente externo. O procedimento inicia aplicando redes neurais convolucionais para detectar pedestres a partir de uma série de imagens. Em seguida, um algoritmo de supressão de fundo é proposto para reduzir a influência da mudança de fundo. O método se baseia na segmentação da imagem, na análise da possível pose e em uma etapa de refinamento final baseada no relaxamento probabilístico. Uma vez extraídas as regiões, informação espúria removida e a pose separada, os atributos da pose são derivados e analisados para rastreamento. Portanto, dois conjuntos de imagens estão disponíveis, com e sem supressão do fundo. Esses conjuntos são usados para rastrear pedestres em série de imagens. O rastreamento é formulado como um problema de “matching” de atributos de um pedestre em quadros de imagens subsequentes, criando, portanto, uma correspondência entre pedestres em sequência de imagens. Para tanto, são comparados os histogramas de duas regiões contendo um pedestre em imagens diferentes. Três opções são analisadas: usando a distância euclidiana; usando Dynamic Time Warping e usando a correlação entre histogramas. Os melhores resultados de rastreamento foram obtidos usando a abordagem de correlação, com precisões acima de 80% e é capaz de lidar com problemas de mudanças na aparência (i.e, pose e forma) e oclusões parciais. No entanto, como lidar com oclusões totais, fundo muito desordenado, permanecem um desafio a ser abordado em trabalhos futuros.

Palavras-chave: detecção de pedestre; segmentação e rastreamento; modelos deep learning; processamento de imagem; supressão de fundo; relaxamento probabilístico; correspondência de histograma.

Abstract

Pedestrian tracking is a well-established research field when it is performed in an indoor environment or the camera is static. But when the camera is moving, as in the case of the outdoor environment, there are many open issues to be solved. Some of these issues concern: illumination changes, occlusion, cluttered background, and pedestrian movements.

In this thesis, deep learning-based methods and image processing technique frameworks are combined to detect and track pedestrians from high spatial resolution images obtained in an outdoor environment. The framework starts by applying deep convolutional neural networks to detect pedestrians from a series of image frames. Then a background suppression algorithm is proposed to reduce the influence of the changing background. The method is based on image segmentation, the analysis of the possible pose, and a final refinement step based on probabilistic relaxation. Once the regions are extracted spurious information is removed and the human figure is separated from the background, feature blobs from the human figures are derived. So, two sets of images are available, with and without background suppression. These sets are used to track the pedestrian in the image series. The tracking approach matches the extracted features of an individual pedestrian in subsequent frames, hence creating a correspondence of targets across multiple image frames. For this purpose, the histograms of two regions containing a pedestrian in different images are compared. Three options are compared: using the Euclidean Distance; using Dynamic Time Warping (DTW) and using the correlation between histograms. The best tracking results were obtained using the correlation approach, with accuracies above 80%, and addresses the problem of changes in appearance (i.e., pose and shape) and partial occlusions. However, full occlusions, more cluttered scenarios, remains a challenge to be addressed for future work.

Keywords: pedestrian detection, segmentation and tracking; deep learning models; image processing; background suppression; probabilistic relaxation; histogram matching.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS/ABBREVIATIONS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DTW	Dynamic Time Warping
FC	Fully Convolutional
IoU	Intersection Over Union
mAP	Mean Average Precision
NN	Neural Network
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RFCN	Region-based Fully Convolutional Network
R-CNN	Region-based Convolutional Neural Network
RPN	Region Proposal Network
ROIs	Regions Of Interest
SSD	Single Shot Detector
SLIC	Simple Linear Iterative Clustering

SUMMARY

1 INTRODUCTION

Pedestrian detection and monitoring from images are promising research fields, as it has become a viable solution with the current technology of imaging sensors in various configurations. Its products can support public security services, such as the police and or fire departments.

The increasing availability of imaging sensors in urban areas (e.g., street or indoor cameras, cameras mounted on vehicles, closed-circuit television systems, aircraft, and unmanned vehicle systems) for observing and monitoring humans have produced a huge amount of data images with detailed information. Visual analysis of such image series is tedious, so, automated methods are desired. The demand for automated methods to detect pedestrians in an image and track them along several images to support monitoring activities is growing.

Over the years, different techniques have been developed to ease automatic object/pedestrian detection and monitoring. Thus, researches have focused on the improvement of image processing techniques by making algorithms more accurate and robust for object/human detection (Dalal and Triggs, 2005; Dollar et al., 2009; Braun et al., 2019; Baabou et al., 2019; Zhai et al., 2020), segmentation (Milan et al., 2015; Minaee et al., 2020) and tracking (Stauffer and Grimson, 1999; Yilmaz et al., 2006; Bibby and Reid, 2008; Milan et al., 2015; Chen et al., 2016; Sun et al., 2020).

Although many approaches have been already proposed to track objects (Yilmaz et al., 2006; Yang et al., 2011; Pan et al., 2017; Li et al., 2018; Ciaparrone et al., 2020), there still challenges to address (Ruan and Wei, 2016). Cluttered background, illumination, and background geometry changes, partially or fully occlusion (caused by another person or tree), deformation of the person (the person changes his pose as he moves and his appearance also depends on the position and range of the imaging sensor), motion blur, and scale variation (when the person moves away from the camera, for example) are just some of the problems that are needed to be addressed in human detection and tracking. This thesis contributes to the solution to part of such challenges. It is proposed a framework that combines deep learning and image processing techniques to detect and track pedestrians from high-resolution images taken by a low-cost camera in an outdoor environment. The process starts by applying deep convolutional neural network models to extract a region of interest i.e., the location of the pedestrian in the image as a bounding box. Then, the separation of a human figure from the background is performed based on predefined pose fields and probabilistic relaxation. After the human figure is separated, the segments are analyzed for feature extraction. Finally, the tracking is formulated as a problem of feature matching/correspondence based on the comparison of the histograms of the regions containing a pedestrian using correlation, Dynamic Time Warping, and the Euclidean distance approaches.

1.1 HYPOTHESIS

State-of-the-art deep learning methods can detect pedestrians in high-resolution RGB images with enough accuracy and precision to extract features that enable pedestrian tracking along a sequence of image frames.

As the objects move along the scene, the background changes, which difficult tracking. This problem can be solved with background/foreground segmentation algorithms.

1.2 AIMS

This thesis aims to develop methodology to detect and monitor the movement of pedestrians from high spatial resolution images, obtained in an outdoor environment, using deep learning-based methods and image processing techniques framework. To achieve this main aim, specific objectives have been set as follows:

- Study the applicability of deep-learning models for object detection in image frames acquired by digital camera installed on mobile vehicles surveying urban areas.
- Propose a novel foreground pedestrian segmentation-based approach.
- Compare three different approaches to measure the similarity of regions of different images.
- Explore image processing methods for pedestrian tracking in a series of image frames.
- Propose histogram similarity measurement-based tracking technique for pedestrian tracking.

1.3 BACKGROUND AND RELEVANCE

Events or situations where large numbers of people are concentrated or moving are common in urban areas. Human mobility follows common patterns. Generally, it dependents on urban roads, sidewalks, as well as the necessity to reach certain locations. However, when an unusual event occurs, such as an accident, this routine can change drastically. The knowledge of people movements is important for institutions involved to security of society. For example, during sport events when fans and the general public gather and move around stadiums or arenas, looking for tickets to the venue and after the event, when they look for transportation for their return.

The seemingly random movement of individuals forms logical patterns at the scale of crowds. In the field of security policies, individual movements can be anticipated but 'human' behaviour remains a mystery. Peaceful marches and sports events can still go horribly wrong. During marches or sports events, the interaction of rival groups can lead to outbreaks. Therefore, the flow or movement of pedestrians or groups of pedestrians remains almost impossible to simulate due to sheer lack of knowledge. Although predictable it exhibits some peculiar characteristics, and therefore, the safety of the population is guaranteed on the basis of this flow. However, casual events can occur, causing panic or disorder, for example, the clash between two rival factions. Likewise, the population today is subjected to threats and violence, such as robberies planned by elements who wish to take advantage of the chaos of the crowds, such as in markets, street fairs, or political demonstrations. Recent events, including major terrorist attacks, have forced governments to make personal and asset security a priority in their policies.

In the context of a pandemic outbreak, currently a theme of discussion, monitoring systems can be useful for health and security services to detect, identify and monitor gatherings (thus enforcing quarantine rules and deterring gatherings that violate isolation or social distancing rules) and/or people walking around without a protective face mask.

For safety and security purposes, for example, instead of monitoring large groups, the act can be applied only to persons who may be suspected, by virtue of specific facts, either of having committed or contributed to an offense.

In this thesis, an approach to detect, segment, and track persons for application in surveillance systems is shown as well as in similar domains mainly in reasoning about people's intentions.

The following contributions can be highlighted:

- First, the usage of high-resolution RGB images, acquired through mobile cameras for pedestrian detection, is evaluated. Then, the state-of-the-art CNN-based object detection methods, namely Faster R-CNN Inceptionv2, and SSD MobileNet v2, for the detection of pedestrians on said mobile/RGB imagery are compared.
- A novel approach to perform the separation of human bodies from images with changing backgrounds is proposed. The method is based on image

segmentation, analysis of the possible pose, and a final refinement step based on probabilistic relaxation.
- Finally, a histogram-based tracking approach is proposed.

Part of the contents and results from this thesis has been published or submitted for publication. The following list provides some references to these documents.

- Amisse, C., Jijón-Palma, M. E., Centeno, J. A. S. (2019). *Fine-tuning deep learning models for pedestrian detection.* Bulletin of Geodetic Sciences.
- Amisse, C., Jijón-Palma, M. E., Centeno, J. A. S. (2019). *Pedestrian segmentation from complex background based on predefined pose fields and probabilistic relaxation*. Bulletin of Geodetic Sciences.

1.5 THESIS OUTLINE

In this manuscript, after the introduction, we introduce the literature review, state-of-the-art, the methods and experiments, results and conclusion of the research. The literature review shows an overview of some recent deep learning algorithms, utilized in human/object detection. The section also introduces transfer learning techniques and region-based detectors based on deep learning. Chapter 3 reviews relevant state-of-the-art techniques for pedestrian detection, foreground segmentation and pedestrian monitoring. In Chapter 4 is give an in-depth overview of the methodology and how the different steps work and can be utilized together to achieve the goal of pedestrian monitoring. The framework starts by using transfer learning method to fine-tune end-to-end deep learning models to detect pedestrians. Then a technique for foreground segmentation to separate human figure from foreground and extract features to feed the tracking algorithm is proposed and discussed. Here, is performed a comparison of three different pedestrian tracking methods based on the Euclidean Distance, Dynamic Time Warping and Correlation. In the next chapter, the proposed approaches are tested on several image sequences and the experimental results and analysis are provided. Finally, Chapter 6 summarises the findings of this thesis. It presents the conclusions based on the observations made in previous chapters and suggest future research directions. For the thesis purpose, the terms pedestrian, human, and person will be used as synonymous.

2 LITERATURE REVIEW

2.1 MACHINE LEARNING AND DEEP LEARNING

The concept of Machine learning was introduced in 1959 by Arthur Samuel as "the field of study that gives computers the ability to learn without being explicitly programmed" (Arthur L Samuel, 1959). In the last decades, the machine learning activities evolved significantly due mainly to the increase of computational power and Graphical Processing Units (GPU) acceleration, and the rise of large repositories of data, being almost ubiquitous nowadays. Machine learning systems, can be classified taking into account the amount and type of supervision they get during training into four concepts:

Supervised learning – The training data includes the ground truth, also called labels. Those labels are usually defined by humans, and the algorithm is able to predict the label for the unseen data. Common tasks in this field are classification and regression. In the case of object detection, training images that were previously annotated mark the locations and the classes of meaningful objects.

Unsupervised learning – In unsupervised learning, the training data is unlabelled, and the system tries to learn without guidance. The most relevant algorithms applied in this field are clustering, the visualization and dimensionality reduction, and the association rule learning. The major goal of unsupervised learning is to discover patterns in unlabelled data.

Semi-supervised learning – In semi-supervised learning, the training data is composed partially with labelled data. It is a combination of both labelled and unlabelled data to train the model.

Reinforcement learning – The learning system, denominated as agent, select and perform actions in an environment, getting rewards in return, or penalties as negative rewards, which allows the system to learn by itself, and able to identify the best strategy possible in order to get the most rewards. In this field, can be included genetic algorithms, swarm intelligence, among others.

Deep learning also known as deep structured learning or hierarchical learning is part of the Machine Learning methods. Deep Learning is defined as the study and modelling of learning processes and algorithms that give computers the ability to learn without being explicitly programmed. In the last few years, Deep Learning algorithms are gaining applications in remote sensing, as shown by recent publications such as Zhang et al. (2016), Zhu et al. (2017), and Li et al. (2018).

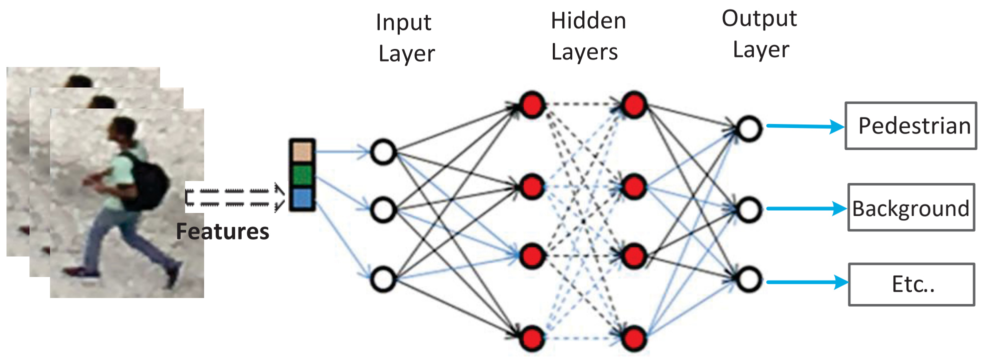
The main difference between machine learning and deep learning methods is found in the learning process. While in machine learning the system learns patterns based on samples of characteristics and features of the objects of interest presented by the analyst (or system programmer), in deep learning algorithms, learning is based on the samples presented to the system. The system is responsible for identifying and parameterizing the most relevant characteristics of the object classes presented. In the learning process, the descriptors are automatically extracted from the images by the system. Then, analyses and comparison of the differences and similarities between groups of objects through the examples is performed. It is also said that deep learning performs “end to end” learning. Meaning that the inputs of this system are raw data and a task to be performed, in the case of this thesis, the detection of pedestrians, and the system learns how to perform the task automatically. One of the main advantages of deep learning algorithms is that their performance improves by increasing the amount of data. The deep learning algorithms relies heavily on appropriate training. For this reason, large reference data sets (training samples) are required.

Deep learning algorithms are neural networks with many layers that learn the characteristics and features directly from the data using the backpropagation algorithm, generally without the need for manual extraction of object features. The most popular deep learning techniques include (Li et al., 2018): Multilayer Perceptron, Convolutional Neural Networks, Auto-encoders, and Restricted Boltzmann Machines. In the sequel, convolutional neural networks are described, as they were used for pedestrian detection in this thesis.

2.2 CONVOLUTIONAL NEURAL NETWORKS

A neural network is, basically, an arrangement of small logical processing units in which the input signals are weighted and added together to generate an output signal that is passed on to other units (network). The weight adjustment is a function of training dataset or data that have the required information. Figure 2.1 shows a simple neural network scheme. In this figure, a vector of parameters derived from the image, for example, the values of the digital counter in each band of a multispectral image, are presented to the network through an input layer. In these processing units (the mathematical neurons), the inputs are combined and the result passed on to each unit of the next layer, the hidden layers. The process is repeated in the different hidden layers and finally, a set of output values is generated, which may, in the case of classification, be the most likely class. Variations can be introduced by changing hidden layers of neurons per layer. Likewise, the input can be varied. For example, instead of digital counter values, spatial and spectral parameters or features of regions resulting from the segmentation process can be used.

FIGURE 2.1 - EXAMPLE OF THE STRUCTURE OF A NEURAL NETWORK.

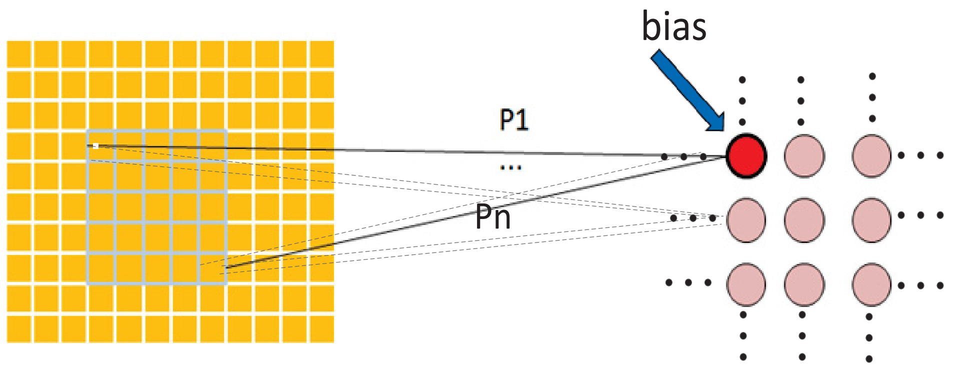


The neural network can also be applied for the analysis of regions of the image, through a sliding window. In this case, all pixels in the region would form the input vector. However, the architecture of a neural network does not take into account the spatial relationships between the elements of an image (pixels or segments). Convolutional Neural Networks (CNN) are class of Deep Learning algorithms, were designed to address this shortcoming. CNN uses three basic concepts: local receptive fields, shared weights and pool area. As Zhang et al. (2016) pointed out, the deep structure of CNN allows the model to learn, to detect highly abstract features or characteristics in the images and to be able to map and store such characteristics.

2.2.1 Local receptive fields

In CNN, the input is a region of the image, a moving (sliding) window. For this, a rectangular (or square) array of neurons is used to register the local variation of the image in the researched region. Schematically, this can be represented according to Figure 2.2. The digital value of each pixel in the image is then read and used as input in the two-dimensional array of neurons represented by the red array in Figure 2.2. Each input is weighted using a weight (p_1, p_2, ..., p_n), initially unknown and then adjusted during the iterative process, and “bias”, a constant value is added to the weighted sum of inputs.

FIGURE 2.2- LOCAL RECEPTIVE FIELDS.



Each hidden neuron uses the bias and a matrix of weights to compute an output. So, for the entire single hidden neuron matrix, the same weights and biases are used. Within a neuron, the input signals are weighted and combined according to equation 2.1 (Castelluccio et al., 2015):

$$s = \sigma\left(b + \sum_{l=1}^{N}\sum_{m=1}^{M} w_{lm}\, a_{j+l,i+m}\right) \qquad (2.1)$$

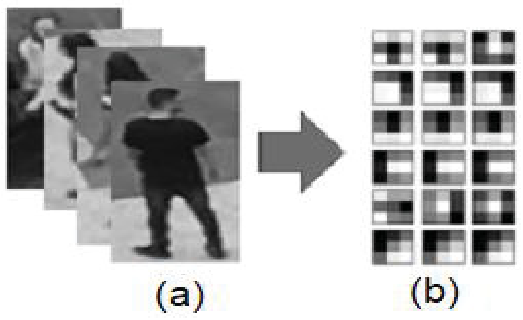
σ denotes the activation function, for example, the sigmoid function, *b* is the bias, $w_{l,m}$ the weight matrix and $a_{i,j}$ the input activation at particular position i,j (row, column) of the image and N, M are the size of the sliding window.

The summations in equation 2.1 resembles the formal description of a filtering process by convolution, where the filter is displaced along the image. Traditionally, the weights of the filter are previously selected to obtain the desired effect, like low-pass or high-pass filtering, but they can also be used to detect some spatial features, like edges in given directions (directional filtering). In equation 2.1, the result of the convolution is modified by adding, or subtracting a constant (bias) and then modulated by the activation function. The difference here is that the weights are not previously programmed, but they are adjusted by the system according to the samples that are presented.

As the process is repeated along the image, a neuron in the first hidden layer can detect the same image feature at different locations in the input image. Therefore, convolutional networks are quoted to be invariant to translation, a factor that becomes relevant in the pedestrian detection problem, because the position of the pedestrians in the image is “a priori” unknown.

Figure 2.3 is an illustrative example of some convolutional kernels that could be obtained using a convolutional neural network to detect pedestrians. On the left it is shown a set of examples that are the input to the network. Based on the examples, the system adjusts the weights of the kernels (Figure 2.3b) to detect the most useful kernels. The 18 small images represent the kernels that are adjusted using 18 hidden neurons. Some of them can be used to detect corners, like those displayed in the second line.

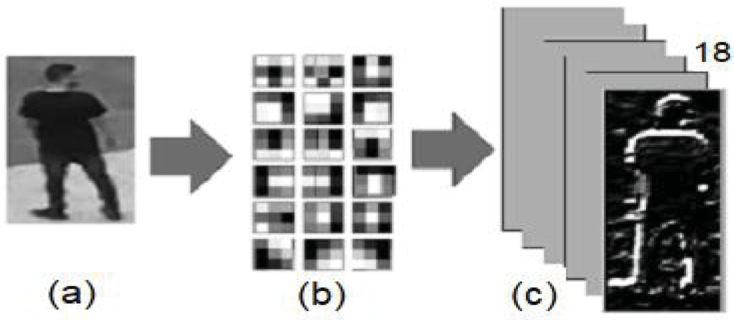
FIGURE 2. 3 - EXAMPLE OF LOCAL RECEPTIVE FIELD



2.2.2 Convolution layer

The use of a single filter allows detecting a special feature, a single characteristic. However, to detect a pedestrian in an image, several feature maps are needed. Therefore, a set of neurons is organized in a layer. A convolutional layer is composed of a set of different filters, each one detecting a different feature. Each neuron produces its own feature map, the result of the convolution. Figure 2.4 illustrates this procedure. When an image is presented, the filters are applied and each filter produces a feature map, like those displayed on the right side of Figure 2.4. As in the example, 18 neurons are applied, 18 feature (convolution) maps are obtained.

FIGURE 2.4 - EXAMPLE OF FEATURE MAPS



The convolution maps can be used as input of a new convolutional layer. Then, the neurons in the next layer use the basic feature maps of the first layer to compute more complex feature maps. So, the input feature map is convolved with a set of learnable convolution kernels to generate new feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights (also known as a filter bank) (LeCun et al., 2015). The convolutional layers are responsible for modeling the features from the images. The first layers usually obtain low-level features (like edges, lines, and corners), while the others get high-level features (like structures, objects, and shapes).

A complete convolutional layer consists of several different feature maps, as illustrated in Figure 2.4. Note that the weights were adjusted to detect some different patterns in the presented input set.

The user can specify the kernel size in the model architecture. The kernel is set to scan spatially the input image and at each location, it generates an output value. The kernel or filter is a feature detector that scans the image for specific features, that is, different types of shapes, edges, circular patterns, or specific color combinations. The kernel moves spatially over the input through the stride. These operations generate a map of features that compose the output of the convolutional layer expressed as (Xia et al., 2017):

$$x_i^k = \sum_{n=1}^{C} w_i^{n,k} \otimes x_{n-1}^n + b_i^k \tag{2.2}$$

where x_i^k is the *k*th output feature map after the convolution of the *k*th group convolution kernel using the input feature map; each group contains C convolution kernel; *k* = 1, 2, ... , K; *n* = 1, 2, ... , C is the channel index number of input feature maps; $w_i^{n,k}$ is the weight of the convolution kernel at the *i*th layer, the filter of *n*th channel in the *k*th group convolution kernel; ⊗ represents the convolutional operation; x_{n-1}^{n} is the input feature map of channel n, and b_i^k is the bias of the *k*th group filter in the *i* th layer.

2.2.3 Pooling layer

The result of a convolutional layer is a set of feature maps that have almost the same size as the original image. As they are used as input in the next convolutional layer, the amount of processing effort is increased drastically by the creation of multiple feature maps. To avoid this problem, the convolutional maps are resized before they are used as input in the next convolutional layer. This is performed by adding a "pooling layer" after the convolutional layer.

Pooling layers (also known as subsampling or downsampling) are used immediately after convolutional layers to reduce the dimension and quantity of trainable parameters of a CNN and to screen the main representative features from the activation layer output feature maps (Li et al., 2018). A pooling layer generates a downsampling operation which reduces the in-plane dimensionality of the feature maps to introduce a translation invariance to small shifts and distortions and decrease the number of subsequent learnable parameters. The hyperparameters in pooling operations are filter size, stride, and padding. The downsampling of feature maps is typically performed through pooling operation (Figure 2.5) being the L2 norm and the max-pooling functions the most common ones. The L2 norm pooling takes the sum of squares of the activation values (w_i) of the neuron region as:

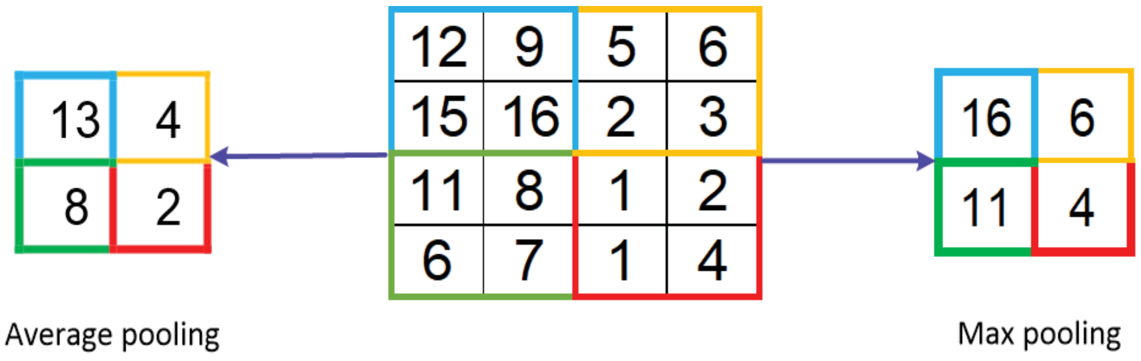
$$L2 = \lambda \sum_i w_i^2 \tag{2.3}$$

where λ is a smoothing factor. In the max-pooling function, the clustering unit adopts a maximum value as the representative value from all the elements in the corresponding region of the pooling window (LeCun et al, 2015), which is equivalent to consider the intensity of a given pattern. The mathematical expression of max-pooling is described as follows:

$$a_j = max\left(a_i^{mxn}\, u(n,n)\right) \tag{2.4}$$

$u(n,n)$ is a window function to the patch of the convolution layer applied to the input data to extract the maximum in the neighborhood, and a_j is the maximum in the neighborhood. Figure 2.5 shows an example of the average pooling and max pooling operations.

FIGURE 2.5 - SCHEMATIC REPRESENTATION OF THE CONVOLUTION AND POOLING LAYERS ON A CNN.



Another option for max-pooling is the average pooling operation that calculates the average value from all the elements as the representative value in the corresponding region of the pooling window. Comparatively, max-pooling is excellent at handling texture features, and average pooling is more sensitive to background information (Goodfellow et al., 2016). Regardless of which form of pooling operation is used, pooling layers aim to capture features, which ensures that the deep CNN can still learn effective features, even if a small amount of input data shift occurs.

So, the basic array of a convolutional neural network is the pair of convolutional and pooling layers. Nevertheless, several layers can be stacked, which results in a deep neural network, composed of several convolutional/pooling layers, which enables computing more complex features as the network becomes deeper.

2.2.4 Fully connected layer

In the last stage of a convolutional neural network, it is necessary to compute the output from the stacked convolutional layers. As explained above, the last layer of the stack is a pooling layer, which is used to compute the output in a final layer, the fully connected layer. The fully connected layers connect each neuron of the pooling layer to the output neurons. The fully connected layers interpret the feature representations generated from previous layers and perform the function of high-level reasoning. Therefore, it analyses the specific class that the produced feature maps correlate most strongly and compute the corresponding probabilities. To do this, it first processes its inputs x with a linear transformation by the weight w and the bias b vectors, and then, the pointwise non-linear activation is performed as follows:

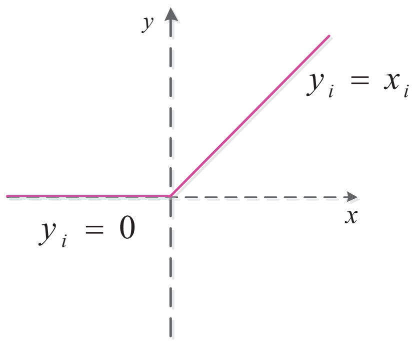
$$Out\ (x) = f\ (w * x + b) \tag{2.5}$$

where out(x) is the output of a fully connected layer and f (·) the activation function.

The activation function is necessary for the model to acquire the ability to capture variations and extract features in the data. The activation functions receive the input signal together with the "bias" and determine the output of the neuron. The most commonly used nonlinear activation functions are Sigmoid, Tanh, and ReLU (Rectified Linear Unit). Because the ReLU function has the form of linear, unsaturated, unilateral suppression and sparse activation, its use in CNN is more common than sigmoid and Tanh functions. The activation function ReLU (Nair e Hinton, 2010) (Figure 2.6) is given as:

$$y_i = \begin{cases} x_i & if \quad x_i \geq 0 \\ 0 & if \quad x_i < 0 \end{cases} \tag{2.6}$$

FIGURE 2. 6 - ReLU ACTIVATION FUNCTION



Unfortunately, the ReLU activation function has some drawbacks. It suppresses all tiny non-zero values to zero, also known as the dying ReLUs problem. In this situation, a common solution is to use alternative classes of ReLU activation functions (e.g., Leaky ReLU, parametric leaky ReLU (PReLU), or randomized leaky ReLU (RReLU), etc.).

2.3 TRAINING THE CNN

All layers, including the convolutional layers and pooling layers of the deep CNN model, use learning algorithms to adjust their free parameters (i.e., the biases and weights) to achieve the desired output. Backpropagation (LeCun, 1989 and LeCun et al., 1998) is the mechanism by which CNN is trained and learns. The backpropagation algorithm computes the gradient of an objective (i.e., a cost/loss/performance) function to adjust CNN parameters to minimize errors that affect its performance. The algorithm trains the CNN model layer by layer performing forward and backward computations stages. In the forward pass stage, the model computes an output based on the provided input and the available weights and biases. The system produces an output that is compared to the desired output and the error and a loss function related to the training dataset can be obtained. In the following stage, learnable parameters, namely kernels and weights, are updated according to the loss value through the backward pass, minimizing the error for each output neuron and the CNN as a whole. The weights w and bias b for any layer i update iteratively with the use of a loss function by the update formula:

$$w^i = w^i - \frac{\partial L}{\partial w^i} \tag{2.7}$$

$$b^i = b^i - \frac{\partial L}{\partial b^i} \tag{2.8}$$

After sufficient iterations cycles of training (forward and backward computations), when the loss cost becomes acceptably low, the training can be concluded. Too often, the training CNN process is prone to overfitting. This is a poor performance on a held-out test set after the CNN is trained on a small or even large training set. Numerous regularization methods have been proposed to prevent CNN from overfitting including dropout, DropConnect, stochastic pooling, among others. We refer to Kukačka et al. (2017) for a detailed discussion. A CNN network can be trained from scratch (complete training) or through transfer learning techniques.

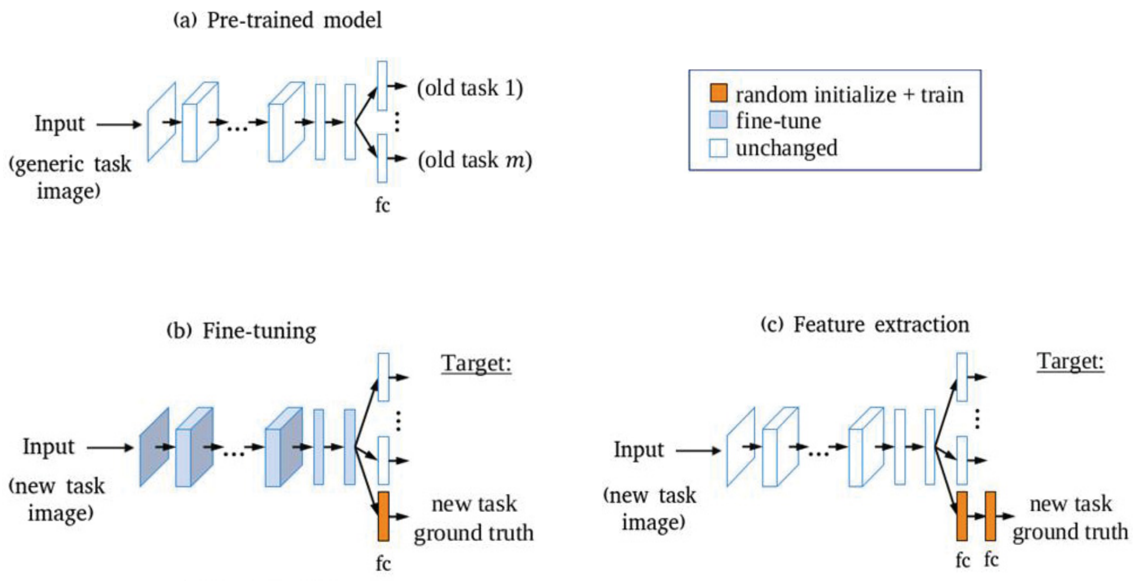
2.3.1 Training from Scratch

A common trait in the success of deep learning is the abundance of training data. However, in many scientific and technological applications, gathering a sufficient amount of data to support complete training often seems to be impractical. This seriously limits the use of complete training or training from scratch for solving problems in the medical, archaeological, astronomical domain, in particular for remote sensing problems. Also, training a deep CNN from scratch requires extensive computational and memory resources, without which the training process can be extremely time-consuming. Therefore, training a CNN from scratch is tedious and time-consuming, and in most cases unviable.

2.3.2 Transfer Learning

When one does not have sufficient data nor the computational resources needed to train CNN from scratch, transfer learning emerges as a promising solution, as it allows transferring knowledge between models in order to reduce the demand for enormous data. Transfer learning gives smaller datasets the possibility to use the knowledge gathered from generic datasets (e.g., ImageNet, COCO, Pascal VOC, etc), saves the time of collecting its knowledge, while the generalization of the algorithm is improved. Pan and Yang (2009) conducted a critical survey about transfer learning through the following questions: "what, which and when to transfer?", "What to transfer" refers to the specific part to be transferred across domains corresponding to the "how to transfer" issue. "When to transfer" suggests transfer learning to be done in a specific situation. Pre-trained models can be used in two ways to transfer learning.

FIGURE 2. 7 - SCHEMATIC REPRESENTATION OF TRANSFER LEARNING



SOURCE: Adapted from Li and Hoiem (2017).

a) Pre-trained model as a feature extractor

For this application, the last fully connected layer of the CNN network topology is removed and replaced with the one that meets the task of extracting features from the new data. Then, a new classifier (e.g., softmax, SVM, kNN, or a new CNN) is added on top of it, and retrained for the feature extraction, as shown in Figure 2.7a. Thus, the weight parameters of the previous layers (convolutional base) do not need to be updated in the retraining process and only the weights of the final layer (new classifier) are updated.

b) Fine- tuning a pre-trained model

Here, the strategy consists either by replacing and retraining the classifier on the top of the convolution base or fine-tuning the weights of the pre-trained model by continuing the backpropagation. As illustrated in Figure 2.7b, fine-tuning enable to fine-tune all the layers of the convolution base, or keep some of the starting layers as fixed (i.e., locked or froze) to avoid overfitting or underfitting and simply fine-tune the higher layers of the network model (Yosinski et al., 2014). This is possible thanks to that CNNs learn general features on the lower level layers (convolution base) and more abstract and dataset-specific features on the higher-level layers (Yosinski et al., 2014).

To fine-tune pre-trained models, it is required to label a new class that will be used to compute the losses. In this case, all layers of the new model will be initialized based on the pre-trained model, except for the last output layer that depends on the number of class labels of the target dataset. And it will, therefore, be randomly initialized. However, in some situations, it is very difficult to obtain the class labels for any target dataset.

The protocol to do transfer learning as feature extraction or fine-tuning depends on the size of the dataset available and the similarity of the source and target datasets. One should apply a feature extraction approach when the target dataset is relatively too small and similar to the source dataset. In such scenarios, the higher-level features learned from the source dataset should transfer well to the target dataset, and therefore lead to a better result. Alternatively, one should apply a fine-tuning approach when the target dataset is large and similar to the source dataset. Altering the already learned and adjusted weights might work well since the CNN model is unlikely to overfit the large target dataset.

2.4 R-CNN OBJECT DETECTOR

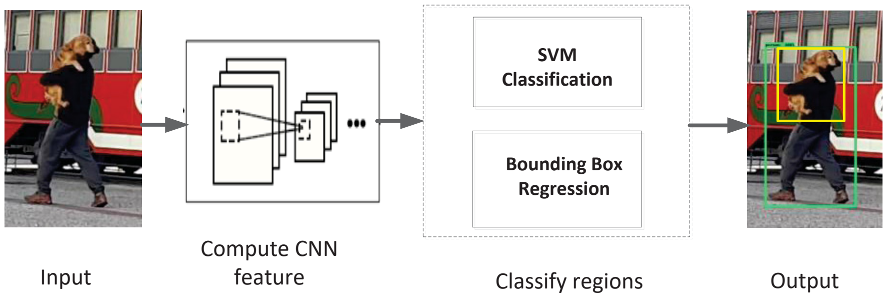
Before the advent of deep learning, object detectors consisted of two stages. Hand-crafted feature extraction (e.g., Haar Feature, HOG, LBP, ACF, etc.) and classification (e.g., SVM, Random Forest, among others). The main strengths of these methods are efficiency in computing, analysis of bad cases, and reasonable performance on limited training data. On the downside, adding more datasets to the model does not impact the performance in a similar way. Driven by the need for boosting accuracy and the use of recently more freely available large datasets, deep-learning region-based CNN detectors have emerged and achieved great success in object detection (Ren et al., 2015).

Region-based object detectors convert the object detection task into a bounding box classification and a regression problem. The state-of-the-art deep-learning-based detector can be broadly divided into two types: one-stage detectors method and two-stage detectors method.

One-stage detectors perform the detections with one pass of the CNN, generating classification and regression outputs directly. These include YOLO and SSD, which can achieve comparatively very low inference time with fairly good accuracy, but they do not handle well objects of small size. On the other hand, R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN are two-stage detectors, i.e., they first generate region proposals and then classify the proposed regions. R-CNN, Fast R-CNN, Faster R-CNN, and SSD are more related to the present work, so they are described briefly in the following.

R-CNN (Girshick et al., 2014): takes as input an image and outputs different regions of interest where the objects in the image could be located. Then, the vector with the different features extracted from each proposed region acts as the input for a set of fully connected layers that output a classification between the different classes and a confidence score. The confidence score helps to create a ranking with the proposals so the most confident ones are taken into account. The whole process can be summarized as follows: it extracts some region proposals from an input image, computes the features of those regions with a CNN, and classify these features with a support vector machine (SVM). The summary diagram of R-CNN is shown in Figure 2.8 in which three main parts can be distinguished: the extraction of region proposals, computation of CCN features, and classification of proposed regions.

FIGURE 2. 8 - ILLUSTRATION OF THE R-CNN ARCHITECTURE.

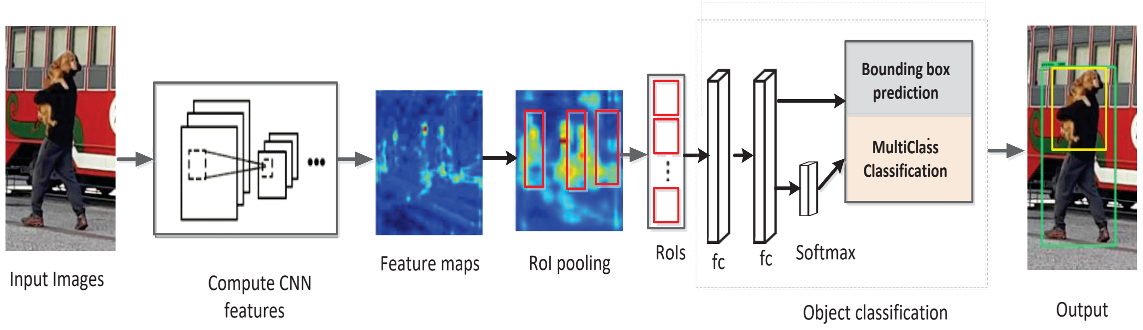


Fast R-CNN (Girshick, 2015): also comprises three modules, with the first one being identical to the one from R-CNN (see Figure 2.9). Unlike R-CNN, the features of the convolutional layers are extracted once, and then the regions proposed by the selective search are projected to the feature maps of the last layer. The Region of Interest (RoI) pooling layer passes these projected regions to the region-specific fully connected layers (fc) for classification and regression. The last module is applied once per region of interest, but the process is made much lighter by removing the need to call the full CNN on each proposal. Fast R-CNN optimizes classification and the bounding box regressors jointly by a multi-objective loss L defined as (Girshick, 2015):

$$L\,(p,u,t^u,v) = L_{cls}\,(p,u) + \lambda\,[u\ \geq 1]\,L_{reg}(t^u,v) \qquad [2.9]$$

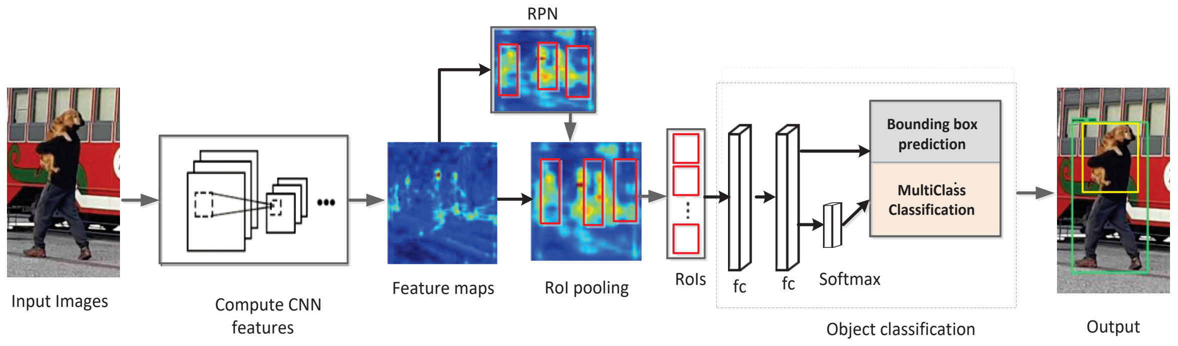
where u denotes the labels of the ground truth class, p the predicted class, v and t^u are respectively, the dimensions of real and predicted bounding boxes. The term $[u\ \geq 1]$ assigns 1 to positive classes and 0 to negative classes. L_{cls} and L_{reg} are respectively losses functions of the softmax layers and the bounding box regression.

FIGURE 2. 9 - ILLUSTRATION OF THE Fast R-CNN ARCHITECTURE.



Faster-RCNN (Ren et al., 2015): in this, a feature map is initially generated by a pre-trained CNN classifier (e.g: AlexNet, VGGNet, ResNet, etc.). Then the generated feature map is fed into a sub-network known as Region Proposal Network (RPN) that will propose candidate object bounding boxes. Finally, Fast R-CNN also accesses the feature map and extracts features from each candidate bounding box using an RoI pooling layer. This operation is based on max-pooling and aims to obtain a fixed-size feature map, regardless of the size of the candidate bounding box at its input. A softmax layer then predicts the class of the proposed regions as well as the offset values for their bounding boxes. An example of a Faster R-CNN and its main stages are illustrated in Figure 2.10.

FIGURE 2. 10 - ILLUSTRATION OF THE Faster R-CNN ARCHITECTURE. IN Faster R-CNN THE RPN SHARES LAYERS WITH A Fast R-CNN SYSTEM.



Different from Fast R-CNN, the Faster R-CNN loss function L is defined by the equation (Ren et al., 2015):

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \qquad (2.10)$$

here *i* is the index of the anchor in the mini-batch. The classification loss L_{cls} (p_i, p_i^*) is the log loss over two classes (object vs background). p_i is the output score from the classification branch for anchor *i*, and p_i* is the ground truth label (1 or 0).

The output is obtained through the normalization of N_{cls} and N_{reg} via weighting by a balance parameter λ.

The regression loss L_{re} (t_i, t_i*) is activated only if the anchor contains an object (i.e., the ground truth p_i* is 1). The term t_i is the output prediction of the regression layer and consists of four variables [t_x, t_y, tw, th]. The regression target t_i* and learned regression output t_i are calculated as (Girshick, et al., 2014):

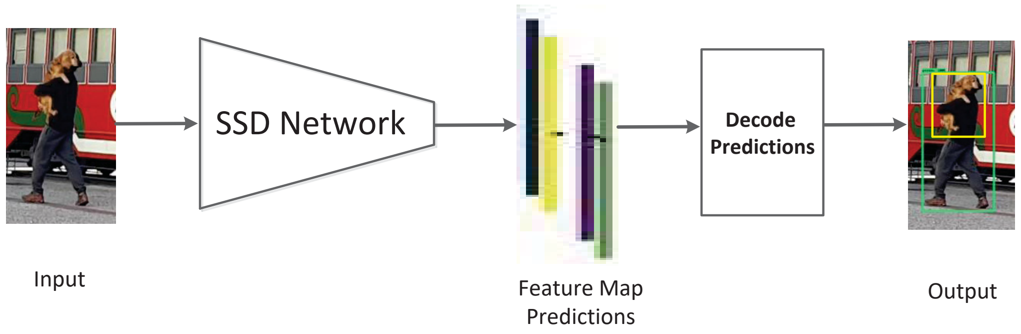
$$t_x = \frac{(x-x_a)}{w_a} \qquad t_y = \frac{(y-y_a)}{h_a} \qquad t_w = \log\left(\frac{w}{w_a}\right) \qquad t_h = \log\left(\frac{h}{h_a}\right) \qquad (2.11)$$

$$t_x^* = \frac{(x^*-x_a)}{w_a} \qquad t_y^* = \frac{(y^*-y_a)}{h_a} \qquad t_w^* = \log\left(\frac{w^*}{w_a}\right) \qquad t_h^* = \log\left(\frac{h^*}{h_a}\right)$$

here x, y, w, and h correspond to the *(x, y)* coordinates of the top-left coordinate and the height *h* and width *w* of the box. x*, x_a stands for the coordinates of the anchor box and its corresponding ground truth bounding box.

SSD (Liu et al., 2016): unlike Faster-RCNN, which has a stage for region proposal, this addresses object detection as a problem of direct regression from pixels to bounding box coordinates and class probabilities. SSD runs a CNN network on a given input image on a single forward pass and computes a feature map. On the computed feature map, SSD runs a convolutional kernel to predict the bounding boxes and classification probabilities. Then, it uses anchor boxes at various aspect ratios to learn the offset rather than learning the box. Finally, bounding boxes are predicted after various convolutional layers. Since each convolutional layer operates at a different scale, it can detect objects of different scales. Figure 2.11 illustrates the SSD diagram.

FIGURE 2. 11- ILLUSTRATION OF THE SSD ARCHITECTURE.



The SSD loss function is a combination of confidence loss L_{conf}, which is dependent on the confidence rate, and the localization L_{loc} that represents the network's performance on estimating the bounding boxes. Both losses are weighted and added as (Liu, 2016):

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \qquad (2.10)$$

here, N is a number of matched default boxes, L is the localization loss, α the weight term, c is the offset for the center, l is the predicted box, and g is the ground truth box parameters.

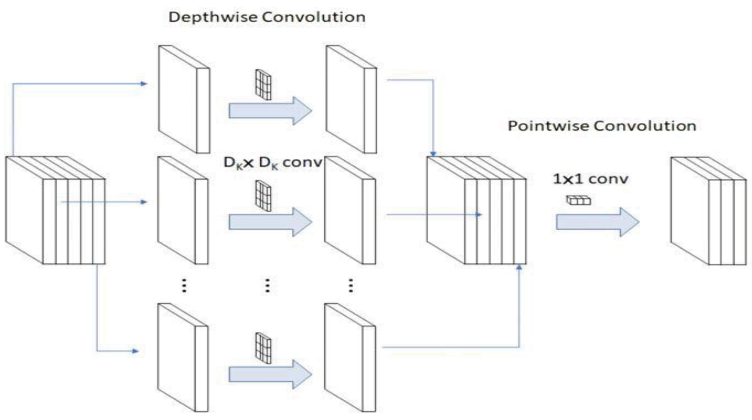
2.4.1 The feature extractor

In the detector architecture, two types of deep neural networks can be distinguished, feature extractor and the detection network. Inception, Resnet, MobileNet, VGG-Net, LeNet are examples of feature extractors networks. The two feature extractor networks used in the present work are described.

Inception: Inception v2 is an enhanced version of the original version Inception v1 (Szegedy et al., 2015), where the 5x5 convolution layer is replaced by two 3x3 layers. Then, the 3x3 convolutions are split into 1x3 and 3x1 and then made wider to remove the computational bottlenecks (Szegedy et al., 2015). There are other versions of inceptions: Inception v3 and v4 (Szegedy et al., 2016).

MobileNet: MobileNets are class of models built with Depth-wise Separable convolution, that consists in a Depthwise convolution followed by a Point-wise one (kernel 1_1) as illustrated in Figure 2.12. Instead of the usual (CONV, BATCH_NORM, RELU), it splits 3x3 convolutions up into a 3x3 depthwise convolution, followed by a 1x1 Pointwise CONV. The Figure 2.12 shows two different steps of the depthwise convolution and the pointwise convolution.

FIGURE 2.12 - THE DEPTHWISE SEPARABLE CONVOLUTION, COMPOSED BY A DEPTHWISE CONVOLUTION AND A POINTWISE ONE.



Source: from Tsang, 2019.

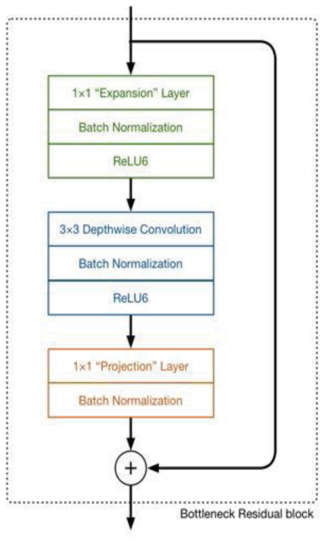
Pointwise convolutions: These are like regular convolutions, but with a 1×1 kernel. The purpose of pointwise convolutions is to combine the different channels of the input. Applied to an RGB image, they will compute a weighted sum of all channels.

Depthwise convolutions: These are like regular convolutions, but do not combine channels. The role of depthwise convolutions is to filter the content of the input (detect lines or patterns). Applied to an RGB image, they will compute a feature map for each channel.

When combined, these two types of convolutions perform similarly to regular convolutions. However, due to the small size of their kernels, they require fewer parameters and computational power, making this architecture suited for mobile devices. At present, MobileNet class of models include MobileNet v1 and MobileNet v2.

MobileNet v2 (Sandler et al., 2018), the architecture used in present work is the update of MobileNet v1. It introduces a new convolutional block called "Inverted Bottleneck Residual Block" (Figure 2.13). The two novelty of this block are: the bottleneck and the residual connection.

FIGURE 2.13 - DEPTHWISE SEPARABLE CONVOLUTION BLOCK.



Source: from Sandler et al., 2018.

The bottleneck block has three convolutional layers: Depthwise convolution layer, expansion layer, and a projection layer. A Depthwise convolution that filters the inputs is followed by a 1x1 Pointwise convolution layer. The 1x1 layer is called Projection layer, because it projects data with a high depth dimension (channels) into a tensor with a much lower one. This layer is also called a bottleneck layer because it reduces the amount of data that flows through the network. The first layer of the block is the new entry. It is again a 1x1 convolution and its purpose is to expand the number of channels of the input before it arrives to the Depthwise convolution. Hence, this layer called Expansion layer, produce an output depth dimension higher than the input ones. It is doing the opposite work of the Projection layer. The decision of how much expand the input is left to the Expansion factor. It is another hyperparameters to choose during experiments to mitigate the trade-off between speed and accuracy. The default expansion factor is 6. The expansion layer acts as a decompressor that unzip the data, then the Depthwise layer performs the filtering, and finally the projection layer compresses the data to make it small again. The trick that makes this all work is that the expansions and projections are done using convolutional layers with learnable parameters, and the model is able to learn how to best compress and decompress the data at each stage in the network.

The residual connection of the block has the function to help with the flow of gradients through the network. It makes a sum operation between the input and the output features maps of the block. The residual connection is used only when the number of channels in input in the block is the same of the number of channels in output.

The advantage of MobileNet v2 over MobileNet v1 is that the input fed to the tensors (connections between each convolution block) is of lower dimension. Low dimension input means tensors now require comparatively less memory and computational resources but, having low-dimensional tensors can be worse too. Filtering a lower dimensional tensor might not give all the useful information. However, the expansion layer takes care of that by expanding the input fed by tensors before the filtering step.

2.5 IMAGE SEGMENTATION MODELS

The image segmentation problem can be seen as an extension to the object detection problem since it is considered a pixel-wise classification task. The most known architectures to solve this problem include: DeepLab, Mask R-CNN, SegNet, RefiNet, PSPNet, UNET, among others (Minaee et al., 2020). In this work, Deeplab v3, Mask R-CNN, and Yolact++ are used to compare them with the proposed method for human segmentation.

Mask R-CNN (He et al., 2017): Mask R-CNN model detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. Mask R-CNN is essentially a Faster RCNN with 3 output branches: the first computes the bounding box coordinates, the second computes the associated classes, and the third computes the binary mask to segment the object. The Mask R-CNN loss function combines the losses of the bounding box coordinates, the predicted class, and the segmentation mask, and trains all of them jointly.

YOLACT++: You Only Look At CoefficienTs (YOLACT) (Bolya et al., 2019) is one of the fastest state-of-the-art instance segmentation methods based on RetinaNet (Lin et al., 2017), a one-stage object detector algorithm. YOLACT proposes to first extract a set of feature maps of different resolutions using a deep convolutional neural network and then further process the feature maps using two parallel branches: (i) the segmentation head branch (the protenet), which is used to generate a dictionary of prototype masks, and (ii), the object detection head branch (the prediction head), which predicts a set of coefficients per instance.

DeepLabv3: DeepLabv1 (Chen et al., 2014), DeepLabv2 (Chen et al., 2017), and DeepLabv3 (Chen et al., 2017) are among some of the most popular image segmentation methods. Deeplabv3 was proposed to capture contextual information at multiple scales by employing several parallel Atrous Spatial Pyramid Pooling (ASPPs). The model combines cascaded and parallel modules of dilated convolutions. The parallel convolution modules are grouped in the ASPP. A 1x1 convolution and batch normalisation are added in the ASPP. All the outputs are concatenated and processed by another 1x1 convolution to create the final output with logits for each pixel. In 2018, Chen et al. proposed DeepLabv3+ by combing the idea of both U-Net and Deeplabv3, as an extension of Deeplabv3 by adding a decoder module to recover the object boundaries.

3 LITERATURE REVIEW

3.1 PEDESTRIAN DETECTION

Pedestrian detection algorithms share similar computation pipelines. First, they extract higher-level spatial representations or features resorting from the raw pixel-level image content to the arbitrarily complex transformation for the application of pixel-by-pixel or window-by-window. Then, the features for any given spatial window are fed to a classifier that assesses whether such a region depicts a human. Next, a scale-space is used to detect pedestrians at different scales.

Viola et al. (2003) were among the first to build an efficient person detector. They used an integral image representation and a feature detection/learning algorithm based on AdaBoost. This was followed by the work of Dalal and Triggs (2005) in which they refine the process and they propose a Histogram of Gradients (HOG) as local image features to be fed to a linear Support Vector Machine aimed at identifying windows containing pedestrians. Felzenswalb et al. (2010) further improved the detection accuracy by combining the HOG with a Deformable Part Model (DPM). These were followed by improvements to the feature-based methods in order to improve feature-based detection. However, most of these methods used feature-based detection coupled with SVM regression methods to perform pedestrian detection.

There has also been considerable work along these lines (e.g., Mikolajczyk et al., 2004; Wu and Nevatia, 2005). Mikolajczyk et al. (2004) used a collection of supervised parts detectors to detect humans using co-occurrence features and AdaBoost classifier. Wu and Nevatia (2005) combine edgelet features and AdaBoost to learn a holistic model for the pedestrian as well as a collection of body parts. Efforts for boosting detection performance and speed include the work of Wojek et al. (2009) that combine HOG and Histogram of Flows, and Nam et al. (2014) that fuses Aggregated Channel Features (ACF) and Locally Decorrelated Channel Features, respectively.

A survey conducted by Dollar et al. (2009) and (2011) summarizes the different feature-based methods and compare their performance on the Caltech pedestrian database. Dollar et al. (2009) and Benenson et al. (2014) state that improving the detection rate requires to improve both feature detection and machine learning algorithms.

Modern pedestrian detectors are based on deep learning techniques. Chronologically, Sermanet et al. (2013) were the first to use CNN in an integrated structure to detect pedestrians. They proposed two layers convolutional model, that adopts convolutional sparse coding to pre-train CNN for pedestrian detection. This approach can be considered as the beginning of the integrated network structures such as R-CNN, Fast R-CNN, and Faster R-CNN. In fact, for Girshick et al. (2014), overfeat can be seen as a special case of the R-CNN network. Chen et al. (2014) proposed a pre-trained Deep CNN (DCNN) to learn features from the ACF detector. These features are then fed to an SVM classifier.

Ouyang and Wang, (2013) proposed a joint deep model that jointly learns four key components in pedestrian detection: feature extraction, deformation handling, occlusion handling, and classifier. Based on Fast R-CNN, Li et al. (2015) proposed a Scale-Aware Fast R-CNN (SAF R-CNN) which takes scales of pedestrians into account in pedestrian detection. Najibi et al. (2016) propose a similar approach through a Grid CNN (G-CNN) network whose differential in relation to Fast R-CNN is the reduction of bounding boxes to be calculated. When testing the application of Faster R-CNN (RPN + Fast R-CNN) in pedestrian detection, Zhang et al. (2016) observed that the RPN network outperforms Fast R-CNN. Therefore, they propose to remove the Fast R-CNN network to improve the detector's performance. When testing on a new pedestrian dataset called CityPersons, Zhang et al. (2017) analyzed the Faster R-CNN architecture and found that the network fails to handle small-scale objects (pedestrians).

Braun et al. (2019) present an overview of the main deep learning detection methods with a particular focus on pedestrian detection. The authors optimize and adapt Faster R-CNN, R-FCN, SSD, and YOLOv3 for the EuroCity Person dataset. They used Faster R-CNN, R-FCN, SSD with VGG-16 as the base classify network, and YOLOv3 with the DarkNet framework. They found that the variation of Faster R-CNN has the best performance on the EuroCity Person dataset. In Lan et al. (2018) it is presented a pedestrian detection based on a variation of the YOLO network (i.e., three layers were added to the original one), in order to join the shallow layer pedestrian features to the deep layer pedestrian features and connect the high, and the low-resolution pedestrian features. Recently, Baabou et al. (2019) reported the state of the art for pedestrian detectors based on deep learning. They presented a comparison and evaluation criteria of the traditional hand-crafted features methods and R-CNN detectors.

3.2 BACKGROUND MODELLING/FOREGROUND SEGMENTATION

Considering a generic situation, where the camera may be moving and the pedestrians move along a scene with varying elements, it is possible that the background around the image of a pedestrian changes from frame to frame. For example, when a pedestrian moves from a bright background into a shadowed area. In such cases, it would be expected that removing the background of the region of interest, and isolating the pedestrian figure, would be more useful for tracking purposes. Background modelling could be a crucial stage of target extraction and the following stages such as tracking needs would depend on the quality of the result produced.

Depending on levels of noise and complexity of the background, simple frame differencing (Piccardi, 2004), optical flow (Horn and Schunck, 1981), or background subtraction (Babacan and Pappas, 2007) can be used. These approaches are implemented generally through extracting interframe motion information, detecting optical flow change, or background modelling. They are simple to implement and results in faster execution times but they are incapable to handle dynamic backgrounds.

When considering complex background environment, one has to deal with a variety of different objects and motion types. More complex methods address these challenges using statistical models for each pixel. Widely used examples are the parametric statistical method which models each pixel with a mixture of Gaussian distributions (Stauffer and Grimson, 1999) and the non-parametric statistical method which estimates the probability density function at each pixel from many samples without any prior assumptions (Elgammal et al., 2000).

One of the first important results of the Gaussian model for background modeling was the work of Wren et al. (1997). They proposed modeling the background at each pixel location with a Gaussian distribution. In order to cope with pixels having a non-Gaussian distribution, Stauffer and Grimson (1999) suggested the Gaussian Mixture Model (GMM). The model is supposed to incorporate effects commonly found in outdoor scenes. Variations of Gaussian and GMM for background modeling have been studied. For example, Kim et al. (2007) have studied how to deal with strict constraints for better background modeling when using the Gaussian model. Zivkovic (2004) also suggested an adaptive GMM (AGMM) to efficiently update parameters in GMM.

Similarly, Lee (2005) used a new adaptive learning rate to boost the convergence rate without changing the stability of GMM. Shimada et al. (2006) have discussed how to deal with accuracy improvement and computational time reduction problems of GMM. However, the problem of pixel-based parametric approaches is that the dynamic textures cause large changes at an individual pixel level (Dalley et al., 2008). Additionally, dynamic scenes make parameter tuning problematic leading to undetected or "ghost" contained foregrounds.

The non-parametric approach, in contrast, can handle situations in which the background of the scene is cluttered and dynamic (Elgammal et al., 2000). Typical non-parametric algorithms are those based on the kernel density estimation (nKDE) technique which estimates the probability density function at each pixel from many samples without any prior assumptions. Although more flexible, it is difficult to choose the suitable bandwidth of KDE for implementation. Increased accuracy has been obtained thanks to the suitable choice of the bandwidth of KDE such as the adaptive Kernel Density Estimation (aKDE) (Mittal and Paragios, 2004), sequential Kernel Density Estimation (sKDE) (Han et al., 2008), and recursive Kernel Density Estimation (rKDE) (Brockwell, 2005). Likewise, the pixel-based adaptive segmenter (Hofman et al., 2012), which models the background by a history of recently observed pixel values, is also a non-parametric background modeling approach that introduces cybernetics to update threshold and background adaptively. Compared to parametric approach, a non-parametric approach is often viewed as more robust and efficiently deployed in the complex background scenes, however, the high computation cost limits its scope and practical application.

Some similar and earlier approaches exploit robust principal component analysis (RPCA) to construct background models (Candès et al., 2011). In Bouwmans et al. (2017) are reviewed different methods based on RPCA.

Other related approach includes visual background extractor (ViBe) method introduced by Barnich and Van Droogenbroeck (2010), which adopts a stochastic renewal strategy to set up and update background models, and then ensures that the information between adjacent pixels is transmitted. Bilodeau et al. (2013), propose a modified Local Binary Similarity Pattern descriptor to set up the background model in feature space. Learning historical representation of the background is another solution. For example, Kim et al. (2004) constructed a codebook to cluster background pixels and construct a background model.

Besides the approaches mentioned above, there is a line of approaches that suggest building a background model by training a neural network (Schofield et al., 1996, Jiménez et al., 2003, Culibrk et al., 2007) to classify each pixel in frame into background or foreground. Schofield et al. (1996) trained neural networks for background modeling and foreground detection by using Random Access Memory neural networks. Jiménez et al. (2003) classified each zone of a video frame into three classes of background with a Multilayer Perceptron Neural Network. In Culibrk et al. (2007), a feed-forward neural network based on an adaptive Bayesian model called Background Neural Network is used for background modeling.

Regarding background/foreground segmentation-based approaches, several strategies were proposed (e.g., pixel-based adaptive segmenter (PBAS), self-balanced sensitivity segmenter (SuBSENSE), pixel-based adaptive word consensus segmenter (PAWCS)). Besides these methods, the most relevant foreground segmentation approaches related to the proposed in the present thesis are those that employ superpixel models (Achanta et al., 2012). In particular, Yu et al. (2018), integrates the watershed algorithm and mean-shift clustering algorithm to obtain reliable initial foreground and background labels for simple linear iterative clustering superpixels. In Schick et al. (2012) the authors first convert a given pixel-based segmentation into a probabilistic superpixel representation. Based on these probabilistic superpixels, a Markov random field uses structural information and similarities to improve segmentation.

Recently, deep neural networks (DNN) have emerged as remarkable approaches to background modeling. DNN-based methods learn and extract deep convolutional features of an image and use them to construct a background model. For example, Shafiee et al. (2016) trained Neural Response Mixture to learn deep features used in the GMM. In another way, Xu et al. (2014) designed a background generation method based on two auto-encoder neural networks, whilst Zhang et al. (2015) trained Stacked Denoising Auto-Encoder to learn robust spatial features and modeled the background with density analysis. Later, Qu et al. (2016) exploit a context-encoder network for a motion-based background generation method by removing the moving foreground objects and learning the feature. Chaibou et al. (2020) have suggested learning superpixels from CNN for foreground segmentation. The survey of Minaee et al. (2020) enables a good and updated view of the most recent literature in image segmentation and discusses more than a hundred deep learning-based segmentation methods.

Generally, deep neural networks can obtain better results at the cost of high the computational load of the process.

3.3 PEDESTRIAN TRACKING

By looking at the story, Rohr (1994) was the first author who tried to track pedestrians using image sequences, with a focus on motion capture of an isolated human. Since then, numerous researchers have developed different approaches to track humans, each approach with its strengths and weaknesses. A comprehensive overview of earlier tracking methods with a particular focus on human tracking is covered in Yilmaz et al. (2006).

The simplest approach to track humans in image frames is to detect them first using background subtraction, and then establish correspondence from frame to frame to track (Wren et al., 1997). Wren et al. (1997) propose a system to track a human in an indoor environment, using color and shape to segment a human body from the background. Then the system detects and tracks the human's head. Haritaoglu et al. (2000) present a w4 model that operates on grayscale or infrared cameras to detect and track humans in an outdoor environment. The model adopts a combination of shape analysis and tracking to locate multiple humans and their body parts in the scene. In McKenna et al. (2000), it is proposed a sophisticated background subtraction method known as adaptive background subtraction to track multiple humans even during occlusion. The method combines color and gradient information to cope with shadows and unreliable color cues. In general, background subtraction is simple to compute, however, it solves the problem of human tracking in complex scenes in which there is unoccluded human and fixed camera.

An alternative approach to background subtraction is to separate the foreground/background using statistical models learned from observed frames (or extracted features) of the frame image sequence. Some of the more prominent algorithms that use statistical models include Mean-Shift, Kalman filtering, particle filter, and template Matching. For instance, Lu and Tan (2001) propose a system based on the color histogram to track humans using a modified mean shift (CAMShift). Beleznai et al. (2006) also employ the mean-shift procedure on different images for detecting and tracking humans.

Tracking systems such as those proposed by Zeng and Ma (2002), Jin et al. (2010), and Zhou et al. (2014) use particle filtering techniques to track people. Zeng and Ma

(2002) combine particle filtering with curve fitting to track heads. Jin et al. (2010) proposes to fuse the information of color-histogram and HOG to track humans in the complicated background through the Particle Filter algorithm. Zhou et al. (2014) introduce a human tracking approach that fuses two cues (color and spatio-temporal motion energy) within a particle filter-based framework.

Another robust algorithm that has been often used for tracking humans is the Kalman Filter. For instance, Zhao et al. (2001), Mirabi and Javadi (2012) explored the Kalman Filter algorithm to track humans in real-world scenarios. Both experiments were able to deal with challenging situations including cluttered background, noise, shadow, and poor contrast. Kalman Filter trackers are claimed to be computationally cost-effective in tracking objects. Pan and co-authors (2017) reviewed moving target tracking. The authors classified the various tracking algorithm based on the extracted features. In their paper, feature extraction methods were discussed and improvement in the tracking algorithm was suggested.

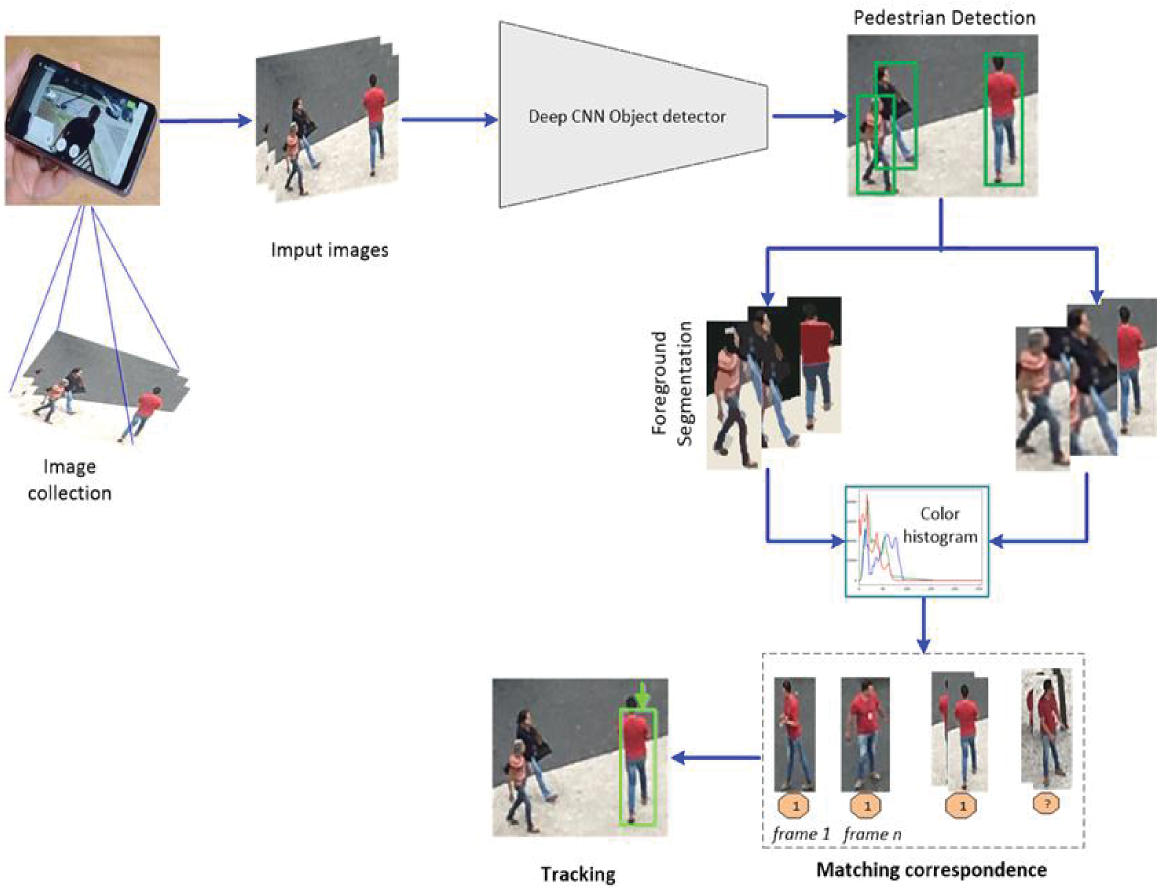
Recent algorithms use representations from CNNs for human tracking. Fan et al. (2010) propose a human tracking algorithm based on pre-trained CNN. An image frame extracted from surveillance videos is used as an input to predict the foreground heat map by single-pass forward propagation and learns a separate class-specific network to track. Jin et al. (2013) use CNN to obtain high dimensional feature vector, and then the feature vector is fed into a radial basis function network to produce a confidence map to the task of tracking problems. Similarly, Hong et al. (2015) came up with the combination of SVM (to learn discriminative appearance models from pre-trained CNN features) and sequential Bayesian filtering (to construct target-specific saliency map), such that they are united to achieve the promising online human tracker. In another work Song et al. (2018) use a one-stage Single shot multibox detector topology and MobileNet for pedestrian detection and tracking from the CCTV dataset. Luo et al. (2018) distinguish pedestrians in surveillance videos through Faster R-CNN and propose a deep Matching-Siamese network for tracking the previously distinguished pedestrian. For RGB-D datasets, Angelico and Wardani (2018) proposed combining the CNN method with Kalman Filter to detect and track humans. In Li et al. (2018) various deep learning-based trackers are discussed. They summarize and classify the existing deep learning trackers based on exploited deep learning networks, feature extraction function, and the training requirements. A comprehensive evaluation of up-to-date deep learning state-of-the-art was performed on available test datasets.

Later, Sun et al. (2020) give a comprehensive survey of recent advances of multiple pedestrian tracking based on a tracking-by-detection framework.

4 MATERIALS AND METHODS

The methodology implemented in this thesis can be divided into three stages (Figure 4.1). In the first stage, pedestrians are detected in each image frame using an object detection framework Faster R-CNN Inception v2 and SSD MobileNet v2. In the second stage, the detected pedestrian bounding boxes are segmented to extract features and compute histograms. So, two sets of images are available, with and without background suppression. These sets are used to track the pedestrian in the image series. In the last stage, an analysis of spatial correspondence between the elements of different images was performed based on the classification of the elements of consecutive image frames according to the histograms of the detected objects.

FIGURE 4.1- OVERVIEW DIAGRAM OF THE PROPOSED WORKFLOW COMPOSED BY THE DEEP CNN OBJECT DETECTOR; FOREGROUND SEGMENTATION AND FEATURE EXTRACTION; CORRESPONDENCES ANALYSE AND TRACKING.



4.1 IMPLEMENTATION OF PEDESTRIAN DETECTORS

There are mainly two types of state-of-the-art object detectors: one stage and two stage methods. The former models traditionally include R-CNN (Girshick et al., 2015), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2017), and R-FCN (Dai et al., 2016), which firstly generate a category-independent set of so-called region proposals (areas of the image that potentially contain an object) for further feature extraction and classification. The latter models based on end to end (also known as one-stage detectors) need setting the default box, training the network, and establishing the relationship of the prior box, default box, and ground truth box. The most common examples of end-to-end based models are YOLO (Redmon et al., 2016) and SSD (Liu et al., 2016). In the present work, the Faster R-CNN inception v2 and SSD MobileNet v2 detectors are tested and used to detect pedestrians in the transfer learning technique.

Although advanced methods, such as Mask R-CNN, YOLO and their versions, provide a “relative” increase in accuracy, their main contribution is the improvement of speed. For example, Mask R-CNN differ from Faster R-CNN in having the segmentation module (Zhao et al., 2019). YOLO, a one stage detector is recognized by achieving higher speed than SSD, but not without affecting the accuracy (Liu et al., 2016). In Huang et al. (2017) is presented a comprehensive review that compares the trade-off between accuracy and speed among different deep learning-based object detectors. They conclude that two-stage detectors achieve higher accuracy while one-stage detectors perform better in speed.

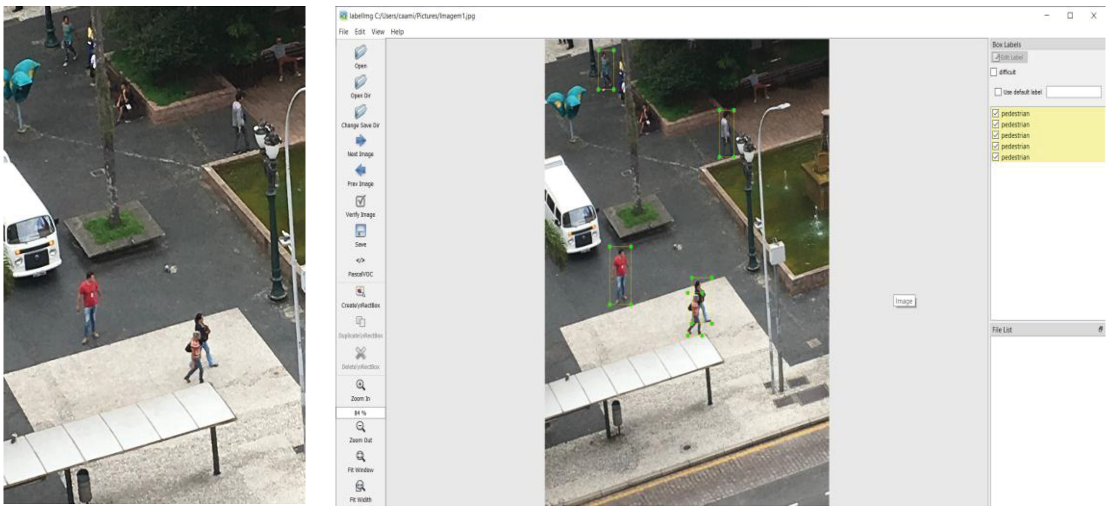
As the evaluation of the execution time is outside the scope of this thesis, these methods would not provide additional value to the experiments. Due to the need for high accuracy in the problem at hand Mask R-CNN and YOLO tests were not considered for pedestrian detection in this thesis. The focus of this step is to generate a bounding box where the pedestrian is located and therefore reduce the size of the region to be analyzed in further steps.

4.1.1 Dataset and training

The dataset created contains 700 RGB images, resized to 1920x1080 pixels. The sequence of images was captured in an outdoor environment in an urban scene using mobile cameras. To simulate a more general situation, the camera was not fixed, so that the exterior orientation of the sensor changed along the recording time. A raw image is Figured in 4.2a as a matter of example. The labelling was performed by using the labelImg tool (from https://github.com/tzutalin/labelImg). This tool allows creating bounding boxes around objects/pedestrian in order to extract the coordinates of those boxes in the image, as can be seen in Figure 4.2b.

Pedestrian labelled in the images includes those who are walking or standing. In the upper part of image in the Figure 4.2b, some pedestrian that are seating or occluded by tree were not labelled.

FIGURE 4.2 - EXAMPLE OF RAW IMAGE AND ANNOTATION USING THE LABELIMG TOOL

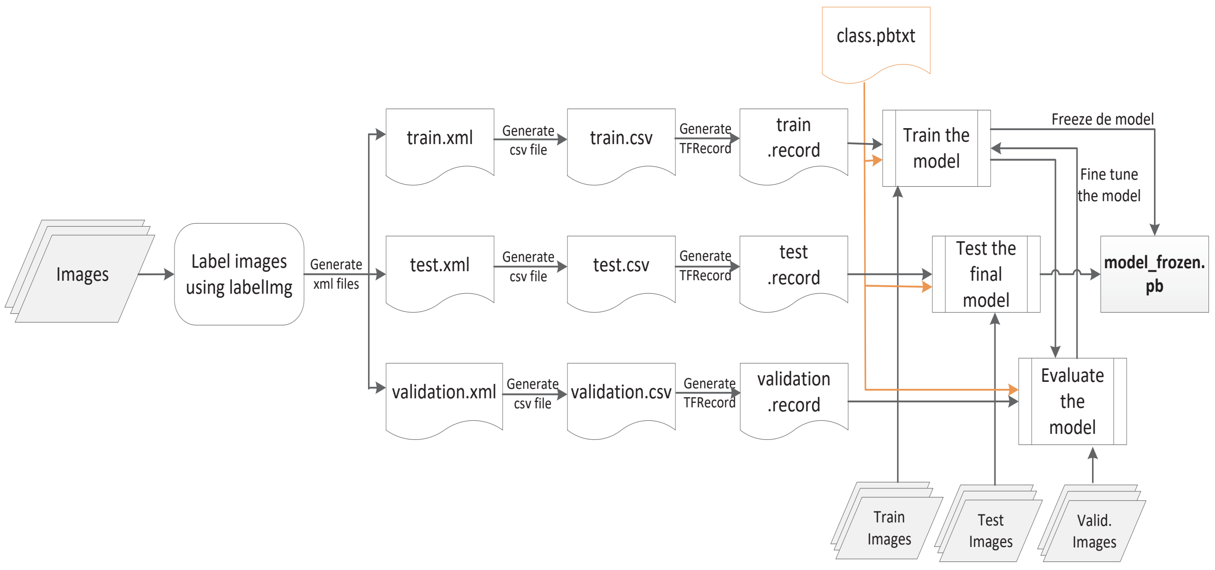


For training the detectors, the dataset was randomly partitioned into a training set (80%), a validation set (15%), and a test set (5%).

In the training step, transfer learning using the pre-trained models provided by the TensorFlow API (Huang et al., 2016) was applied. TensorFlow API, is a toolbox designed for simple, flexible, and easy use of CNN building blocks. It provides premade models (model zoo) combining some of the state-of-the-art detection algorithms with many of the main CNN backbone models. The provided models are trained on the MS COCO dataset with 90 classes. TensorFlow API also provides detailed tutorials to retrain models for new categories using transfer learning. An overview and usage of the TensorFlow Object Detection API is described in the URL: (https://github.com/tensorflow/models/blob/master/research/object_detection/README.md).

Figure 4.3 presents an overview of the workflow to train an object detection model in TensorFlow Object Detection API, an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. The main steps include: the extraction for each image a xml file with the coordinates of each bounding box and the corresponded label associated, in PASCAL VOC format. Because TensorFlow works with TFRecord files, a binary file format, where all the information stored in all the xml files corresponding to each partition will be rearranged and stored in one single file in the appropriate format. All the individual xml files (training, validation and test) obtained using labelImg should be converted to this format.
Creation of a. pbtxt file that will contain the id and name for each of the possible labels in the dataset. The TensorFlow API, do not consider the background as a class.

FIGURE 4.3 - FLOW DIAGRAM TO TRAIN AN OBJECT DETECTION MODEL IN TENSORFLOW OBJECT DETECTION API.



4.1.2. Experiment Implementation

The experiments were implemented on a computational server with the following hardware specifications: Intel (R) Core (TM) Processor i7-7500U CPU @ 2.70GHz 2.90GHz, installed memory (RAM) 8.00 GB 64-bit operating system type, x64-based processor and NVIDIA GeoForce 940MX graphics processing unit.

With regard to hyperparameter setting, many of the settings used in this research were inherited from the original Faster R-CNN and SSD MobileNet settings however, the following parameters were determined empirically.

a) The batch size: The batch size defines the number of samples that will be passed through the network at once. One epoch corresponds to one propagation of all training samples on the network, while the number of steps defines the number of passes one batch size pass through the network. The batch size parameter is mainly selected based on the RAM available on the machine. The batch size impacts directly the training. Large batch processes tend to converge to sharp minimizers of the training function, that leads to a decrease in the capacity of generalization of the model (Kesker et al., 2016).

The batch size is also highly correlated with the speed of the training: the larger the batch, the faster the training process. Another aspect to take into account by defining the batch size is the influence that is going to have on the learning rate parameter. The learning rate is applied once for every batch. In Smith et al. 2017 is verified that equivalent accuracy results can be achieved after the same number of training epochs, but with fewer parameter updates, which consequently shorts up the training time, by increasing the learning rate and scaling the batch size. Most algorithms use a batch size that falls in between, where the primary considerations are based on the memory available and the speed of the training (Goodfellow et al., 2016). The batch size value in the present thesis was set equal to 1.

b) Momentum: The Momentum optimizer (Polyak, 1964) was designed in order to accelerate learning. It accumulates an exponentially decaying moving average of past gradients and continues to move in their direction (Goodfellow et al., 2016). The momentum optimizer is defined by a momentum term, usually set as 0.9, that increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. Consequently, this leads to situations of faster convergence and diminished oscillations (Ruder, 2016). The default value of momentum (0.9) was used in experiments for Faster R-CNN Inception v2 and SSD MobileNet v2 models.

c) Learning Rate: The learning rate hyperparameter in CNN is one of the most important hyperparameters to set when training a CNN. It has a strong impact on both stability and efficiency of training times. If the learning rate is too large, the CNN training can become unstable and never converges, while if too small, training can take orders of magnitude longer than needed and never progresses. The process of finding a good initial learning rate can be difficult. One of the approaches to find a feasible learning rate to the problem at hand is to train the model for a few epochs using different learning rates and to compare the learning curves. The ideal learning rate will learn quickly and converge to a good solution. The decrease in the learning rate can be applied using different strategies, such as: constant learning rate, cosine decay learning rate, exponential decay learning rate, or manual step learning rate. In the present experiments, the learning rate was set equal to 0.0001.

d) Early stopping was adopted for retraining the models. CNN start overfitting when they iterate too many times over the same “small” set of training samples. Therefore, a straightforward solution to prevent this problem is to figure out the number of training steps a model needs. The number should be low enough to stop before the network starts overfitting, but still high enough for the network to learn all it can from this training set.

4.2 PEDESTRIAN SEGMENTATION BASED ON PREDEFINED POSE FIELDS AND PROBABILISTIC RELAXATION.

In the previous step, the detectors architectures output delimiter boxes, the class, and the confidence score for each detected pedestrian. The background of a detected pedestrian in the bounding box contains spurious information (e.g., other foreground objects, illumination changes, poles, the roof of the bus stop). The aim of this step is to separate the pedestrian figure from the background. This is performed in four stages: (i) Preliminary segmentation (superpixel); (ii) Second Level Segmentation by clustering; (iii) Foreground estimation and (iv) Refinement through relaxation.

(i) Preliminary segmentation (superpixel): For segmenting the input image, Simple Linear Iterative Clustering (SLIC) approach proposed by Achanta et al. (2012) is used. SLIC is appealing as it has shown to outperform other state-of-the-art superpixel methods in simplicity, segmentation performance and it does not require much computational power (Achanta et al., 2012).

Given the input image, the SLIC superpixels initiate with sampling the image into K regularly spaced cluster centers and move them to seed locations corresponding to the lowest gradient position. The size (S) of the initial clusters is found through the equation:

$$S = \sqrt{\frac{N}{K}} \quad (4.1)$$

where N is the number of all pixels in the image, and S is the regular grid interval between the superpixels.

Another initial operation in the SLIC process is the combination of a CIELab color vector and an xy spatial coordinates vector to form the five-dimensional Labxy feature vector. The five-dimensional feature vectors are used to display the location of the pixel in 5-dimensional space and to compute the cost distance, which contains the color space (d_c) and the physical space (d_s).

Equations (4.2) - (4.4) summarizes the SLIC algorithm. For each pixel i in the neighborhood of the centroid Cj, the spatial proximity distance (d_s) in coordinates space is computed as:

$$d_s = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \quad (4.2)$$

and the color proximity distance (d_c) in L*a*b color space is:

$$d_c = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \quad (4.3)$$

Then, the overall distance metric (D) is formulated as:

$$D = \sqrt{(\frac{d_c}{m})^2 + (\frac{d_s}{S})^2} \quad (4.4)$$

here, *m* controls the compactness of a superpixel to keep a good balance between color similarity and spatial proximity.

(ii) Second Level segmentation by clustering: The preliminary SLIC segmentation process produces small compact regions. Some parts of adjacent regions may be misclassified as may belong to the same image object and therefore have a similar color. To refine misclassified adjacent regions and achieve a reasonable estimation of the image objects, adjacent similar superpixels were grouped to build up a new larger uniform region.

Having obtained N superpixels, the neighboring relationship between the regions was stored in an adjacency matrix (A, NxN), with the following characteristics: it is a square binary matrix whose dimension is equal to the number of segments (N x N).

The neighborhood of two regions is represented by each cell, (1) if two regions are neighbors or (0) if not represents the neighborhood.

Then a second matrix is constructed to describe the color affinity of two neighboring regions (a and b), by computing the mean RGB values (m_a) of each segment, and the Euclidean Distance (ED) between the average values of the two neighboring regions (a and b) as:

$$ED(a,b) = \sum_{i=1}^{3}(m_{ai} - m_{bi})^{1/2} \tag{4.5}$$

In the next step, the ED is truncated to the range between 0 - 255 and a color similarity (S) measure derived (equation 4.6), which is normalized to the range 0-1.

$$S(a,b) = 1 - \frac{ED(a,b)}{255} \tag{4.6}$$

Finally, clustering is performed in an iterative manner. At each iteration, the best fusion is evaluated according to the color compatibility of the two neighboring segments:

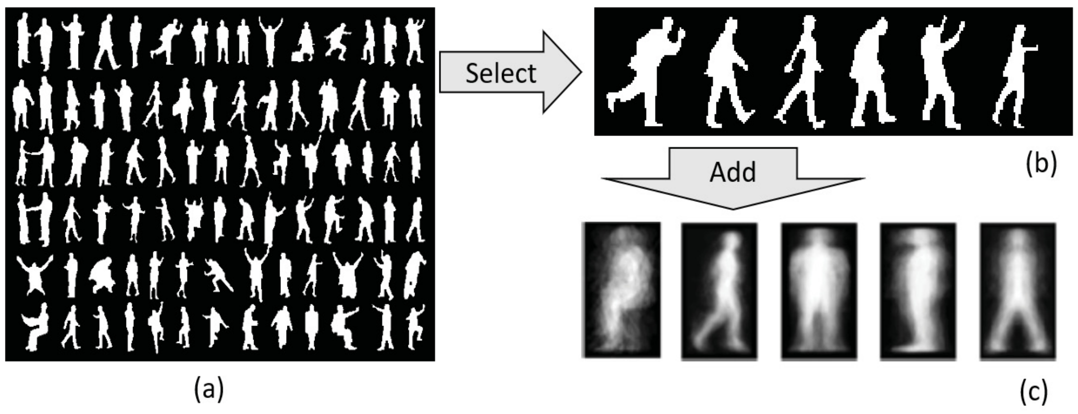
$$A(a,b) * S(a,b) > t_1 \tag{4.7}$$

where, t_1 is the minimum similarity defined by the user. Values between 0.85-9.95 are recommended. The process ends when there are no more possible fusions that satisfy the equation (4.7).

(iii) Foreground estimation: The result of the segmentation by clustering is a set of regions of different colors, but it is still confusing to distinguish which regions belong to the foreground and background. The goal now is to classify the segments in to two classes (foreground and background), although there is no previous information about the color of the classes available. It must also be considered that the background is varied and may be composed of different surfaces. So, pedestrian models with fuzzy boundaries were used to estimate the probability of belonging of a segment to the pedestrian (foreground).

To estimate the probability that a segment belongs to the pedestrian (foreground), the probability of a pixel belonging to the foreground was estimated from a series of available binary images. Several binary human silhouettes in different poses (Figures 4.4a) were downloaded from https://publicdomainvectors.org/ for this purpose. Given the articulated nature of the human body, people can adopt a wide range of poses (e.g.: walking, standing, sitting, lying, jumping, dancing, running, kneeling, squatting, or crouching) and can be captured in different viewpoints from single or multiple cameras. For the experiments, and to keep a reduced number of poses to speed up the process, five different poses: “walking1”; “stand side”; “front”; “walking2” and “sitting” were considered (Figure 4.4c). From the available set, 70 silhouettes for each class were selected. Figure 4.4b displays a small set of silhouettes used as samples of the class “walking”. The selected binary images were resized to 140x70 pixels and added to obtain a new image (as displayed in Figure 4.4c) that stores the frequency of the foreground in the image.

FIGURE 4.4 - COMPOSITION OF POSE IMAGES WITH FUZZY BORDERS. (a) ORIGINAL IMAGE DOWNLOADED FROM PUBLICDOMAINVECTORS.ORG; (b) BINARY IMAGE OF SELECTED POSE; (c) FIVE EXAMPLES OF THE COMBINATION OF MULTIPLE BINARY IMAGES WITH DIFFERENT POSES: “SITTING”; “WALKING1”; “FRONT”; “STAND SIDE”; AND “WALKING2”.



As a result, a fuzzy image aiming to map the density of active pixels is composed by adding various binary images. Then the derived image is normalized to the range 0-1. This process was repeated for five different possible poses described in Figure 4.4c. This field is the description of the probability of a pixel belonging to the foreground. In this thesis it will be referred as density field.

$$M(i,j) = \frac{1}{NB}\sum_{p=1}^{NB} BI_p(i,j) \tag{4.8}$$

The probability fields obtained applying Equation 4.8 allows computing the probability that a segment belongs to the foreground, under the hypothesis that if a segment belongs to the foreground, it will cover a region associated with higher density values. Thus, the segmented image is overlaid to the probability fields of each pose (Figure 4.4c) and a mean value (probability) is computed for each segment. Obviously, the probability values depend on the size of the segments. It is expected that fine over-segmentation produces small segments that would be better defined in terms of probability, but while misclassified segments would be more difficult to correct. The overlay produces different probability values, as a segment may overlay regions with low and high density. In order to identify the pose model that fits better the image, the mean probability of other segments is computed for each pose and the pose with a higher mean is selected. The selected pose is then used to classify the image and separate the foreground from the background. The segments are classified according to their mean. Thus, all segments with a probability below 0.5 are discarded and those with higher mean value are taken into account. As an initial situation, all segments with a mean value above 50% are considered to belong foreground, i.e., pedestrian.

(iv) Refinement through relaxation: Since the fuzzy model does not cover all possible poses, in this step, the aim is to update probability values using the information contained in the neighboring segments. It was hypothesised that if a segment that should belong to the foreground has a “relative” low mean value (e.g., below 50%), and it is not classified as that, its mean can be corrected considering the mean values of the surrounding segments. So, if a segment has low value but is surrounded by high value segments, then its neighborhood would contribute to increasing its probability. On the contrary, if it is surrounded by lower values its mean will decrease. This is done by updating iteratively the probabilities according to the formula:

$$\mathrm{P_i}^{\mathrm{t+1}}(\mathrm{cl}) = \frac{1}{\mathrm{L}}\left(\mathrm{P_i}^{\mathrm{t}} * \left[1 + \frac{1}{\mathrm{nv}}\sum_{\mathrm{j=1}}^{\mathrm{nv}}\left\{\mathrm{R(cl,i,j)} * \mathrm{P^t}_{\mathrm{j}}(\mathrm{cl}) * \mathrm{K(cl,j)}\right\}\right]\right) \qquad (4.\ 9)$$

where:

$P_i{}^t$: stands for the mean probability of the ith-segment at iteration t;

$R(cl, i, j)$ is the compatibility in terms of color between segments i and j;

cl denotes the class (foreground or background);

K is a binary value that is one when region j belongs to class cl;

L is a normalizing factor to keep the probabilities in the range between 0 and 1.

In the iterative process, the probability of each segment is updated considering the information provided by the neighbors. The process stops when the classification reaches a steady situation, i.e., when no segment is reclassified in a different class. Thus, the compatibility function, derived from the Euclidean distance, can vary between 1 (compatible) and -1 (totally incompatible). The color compatibility (CC) between two segments is described based on the color similarity function (S) derived from equation (4.6).

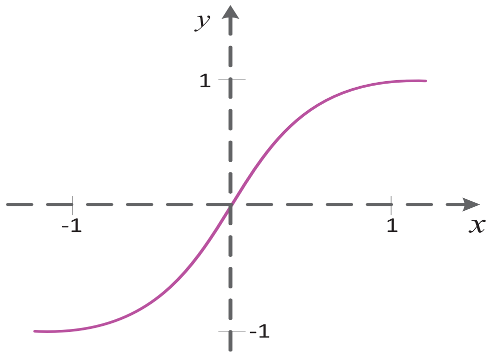
Initially, the similarity is set to range -1 and 1 and then, the sigmoid function is employed according to equation 4.11. A value of 10 iterations is used as it is getting the best results without giving away too much performance.

The color similarity function (S) proposed in equation 4.6, is used to describe the color compatibility (CC) between two segments. First, the similarity is scaled to the range -1 and 1 and then the sigmoid function is applied using equation 4.10. The use of the constant c = 5 is necessary to adapt the sigmoid function to the range (-1,1). The use of the sigmoid function is recommended because it has a higher slope for values close to zero and lower variation in the extremes, which allows changing the classification of dubious situations, as shown in Figure 4.5.

$$x(a,b) = 2c * (S(a.b) - 0.5) \tag{4.10}$$

$$CC(a,b) = \frac{c*1}{1+e^{-x(a,b)}} \tag{4.11}$$

FIGURE 4. 5 - SIGMOID FUNCTION ADAPTED TO THE RANGE BETWEEN -1AND 1.



4.3 TRACKING WITH HISTOGRAM CORRESPONDENCE

To associate a human detected in an image with one in the following image frame, the histograms of the regions of interest are compared. The hypothesis is that, if two regions contain the same pedestrian, even considering pose changes and some variations of the background, the histograms of the regions will be similar, as a considerable portion of the region is occupied by the pedestrian’s clothes. This hypothesis is valid when considering two consecutive frames but could fail when the interval between the frames increases. Therefore, different intervals will be analyzed in the experiments.

The differences between the image frames containing pedestrian in two images are highly dependent on the temporal difference between the two images. To follow a pedestrian in the series of images, one can compare each image with the following. However, it should be considered that the shorter the interval between images, the more calculations are needed to complete the task. At the same time, as the temporal difference between the images increases, the likelihood of finding the person in the following image may decrease as a function of pose and background variations as the person move through the scene. It should also be noted that a pedestrian can be covered by other objects such as poles and other people (occlusion), as well as leave the scene. For this reason, in this part of the research, series were compared with different intervals between images.

The imaging sensor stores 30 images per second. Analyzing a 20-second series, for example, would imply processing 20*30=600 images. For this reason, to verify the possibility of reducing processing load, it was verified the possibility of using one image every five, ten, and twenty frames.

4.3.1 Similarity

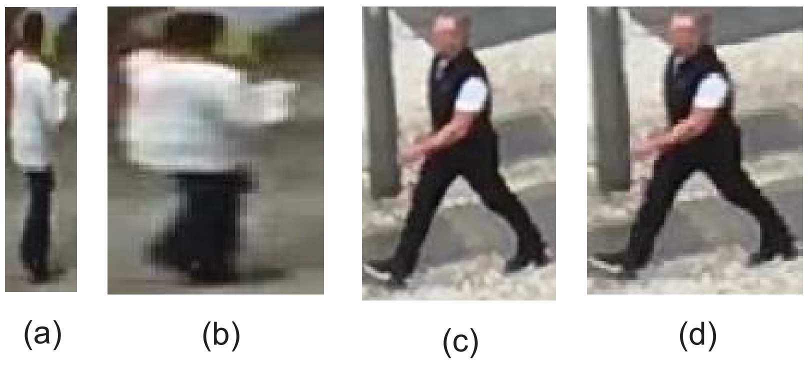
Given an image containing N1 regions in a region of interest containing pedestrians, the aim is to verify the presence of the pedestrian in one of the regions of the next frame. For this purpose, the following notation will be considered:

- ima_1: image in which the regions to be analyzed are located
- reg_i: a region of interest (containing a pedestrian) in the ima_1.
- ima_2: image in which the desire is to find the selected region in the ima_1. It can be the following image, or a later image, for example after 5,10, or 20 frames.
- reg_j = a selected region in the ima_2.

4.3.2 Geometric normalization

There is no guarantee that the size of the regions containing the same pedestrian in the two images is equal. The factors that contribute to this difference are the detection process, which can generate different regions depending on the contrast and pose of the pedestrian. So, for comparison purposes, all images were transformed to the same size: 120 rows per 90 columns. This transformation can introduce severe deformations into the images, depending on the pedestrian's pose. Figure 4.6 shows two examples of the result of this geometric transformation. In the first case, Figures 4.6a and 4.6b, the original image is relatively narrow, so the result introduces exaggeration in the horizontal direction. In the second case, as the person extended the legs while walking, the result is compression in the horizontal direction.

FIGURE 4. 6 - EXAMPLES OF THE GEOMETRIC NORMALIZATION.



4.3.3 Color reduction

Each image contains three bands (RGB). The comparison of two regions of interest implies the analysis of the three bands. Here, the images were transformed to grayscale to simplify the process. So, the mean intensity of each pixel was computed according to Equation 4.12.

$$I = 1/3\ (R + G + B) \tag{4.12}$$

4.3.4 Comparison

The region comparison was performed based on the similarity between the histograms of the two regions. Three options were considered and compared to measure the similarity between regions: the Euclidean distance, the distance measured by Dynamic Time Warping, and the correlation between image histograms.

The histogram of an image is the function that represents the frequency of each digital value in the image. As the images are stored using eight bits, the histogram ranges between [0,255].

Besides, the position of each region of interest is also stored and used in the comparison step. A general comparison would compare a given region to all the available regions in the next frame. Nevertheless, not all positions are possible, as the position of the pedestrian in the next frame is relatively close to the one in the first image when the interval is small. Therefore, the analysis was performed considering all possible combinations and considering only the regions within a pre-defined radius around the original position of the region of interest.

4.3.5 Euclidean Distance

After segmentation and feature extraction, we get a set of feature blobs in the image frame, and then establish the correspondence matching between the target feature. The matching is performed through a Euclidean-distance-based nearest neighbour approach defined as:

$$ED(A_i, B_j) = \sqrt{\sum_{t=1}^{T}(X_{i,t} - X_{j,t})^2} \tag{4.13}$$

where (Ai), (Bj) stands respectively, for a set of features in the consecutive image frames, and X is a vector of features.

The matching is valid if the distance between the two objects is lower than a distance threshold. Otherwise, he will be treated as an untraced pedestrian and removed from the model. The Euclidean distance between two histograms is computed according to Equation 4.14.

$$EucDist = \sqrt{\sum_{i=1}^{255}(H_1(i) - H_2(i))^2} \tag{4.14}$$

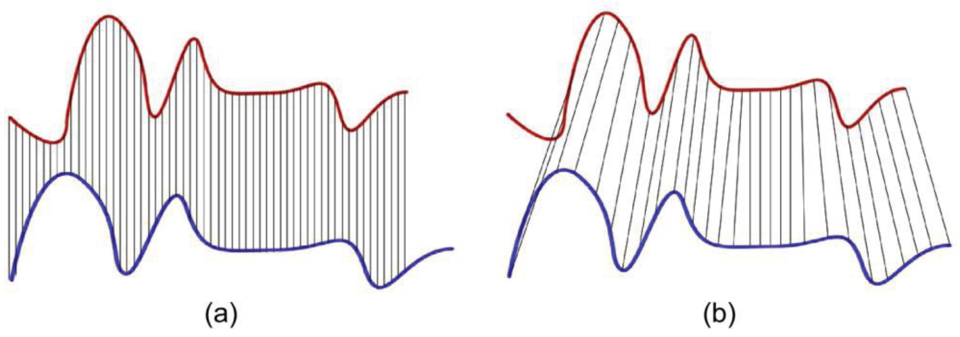
where H stands for the histogram and i for the digital value.

4.3.6 Dynamic Time Warping

The region correspondence was also evaluated based on a Dynamic Time Warping (DTW). DTW is useful to compare time series and can be applied to compare two histograms. DTW was first introduced by Vintsyuk (1968) in the field of speech recognition to match words at different speaking rates. Since then, it was widely applied in different fields such as handwriting recognition (Rath and Manmatha, 2003), matching series of medical images (Felipe et al, 2005), gesture recognition (Alon et al., 2009).

The original DTW algorithm was defined to match temporal distortions between two signals, finding a warping path between two time series: an input signal A = (a1, a2, …, an) and a certain sequence B = (b1, b2, …, bn) (as illustrated in Figure 4.7). In the present application, the time series A and B are image histograms, where each aj and bi are the frequency of the digital values in regions i and j. To align these two sequences, a $M_{m \times n}$ matrix is designed, where position (i, j) of the matrix contains the alignment cost between ai and bj. Then, an adjustment route C= (c1, c2, …, ck) is defined as a set of contiguous matrix elements, defining a mapping between A and B. The adjustment route is subjected to the following restrictions: (i) it is bounded between c (1,1) and c (n, m); (ii) no sequence of the route is to be skipped, and (ii) the sequence should not be reversed.

FIGURE 4. 7 - COMPARISON OF THE ED AND DTW DISTANCE MEASURES. (a) THE ED MEASURE COMPARES THE SAMPLES AT THE SAME TIME INSTANTS, WHEREAS (b) THE DTW MEASURE COMPARES SAMPLES WITH SIMILAR SHAPES TO MINIMIZE THE DISTANCE.



SOURCE: Retrieved from http://upload.wikimedia.org/wikipedia/commons/6/69/Euclidean vs DTW.jpg.

The best route (among several possible adjustment routes) is the one that minimizes the deformation cost, defined as:

$$DTW(A_i, B_j) = min\ (\tfrac{1}{L}\textstyle\sum_{t=1}^{T} C_t) \tag{4.15}$$

where: C_t is the t-th element of the adjustment route, L is the number of elements of the adjustment route. L is used to compensate for the fact that warping paths may have different lengths. The smaller the DTW dissimilarity between two sequences, the similar are the two compared sequences.

The cumulative cost at a certain position $M(i,j)$ is found as the composition of the distance $\delta(i,j)$ between the feature vectors of the sequences ai and bj with the minimum of the cumulative cost of the adjacent elements of the cost matrix up to that point as:

$$M(i,j) = \delta(a_i, b_j) + min \begin{cases} M(a_{i-1}, b_{j-1}) \\ M\ (a_i, b_{j-1}) \\ M\ (a_{i-1}, b_j) \end{cases} \qquad (4.16)$$

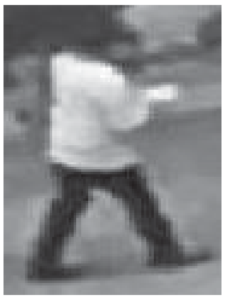
In our implementation, the matching is valid if the distance between the two objects is lower than a distance threshold. Otherwise, will be treated as an untraced pedestrian and removed from the model.

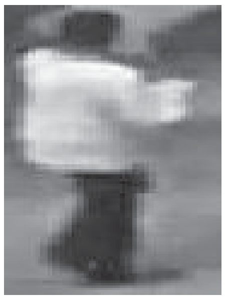
4.3.7 Correlation

The third similarity measure is based on the correlation of the histograms. As both histograms share the same size, the variation of the relative frequencies of the digital values can be compared through the correlation coefficient.

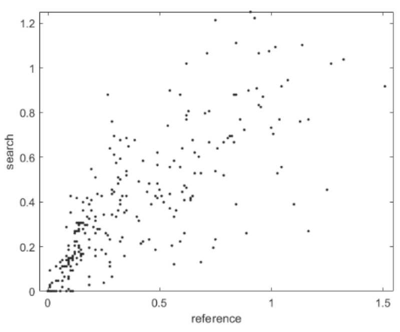
Figure 4.8 displays the comparison of one region (a) with two candidate regions of the next frame (b and c). The first candidate (b) is the right solution and the other (c) would be the wrong choice. Figures 4.8d and 4.8e display the plot of the histogram pairs (a vs b) in Figure 4.8d and (a vs c) in Figure 4.8e.

FIGURE 4. 8 - COMPARISON OF HISTOGRAMS.

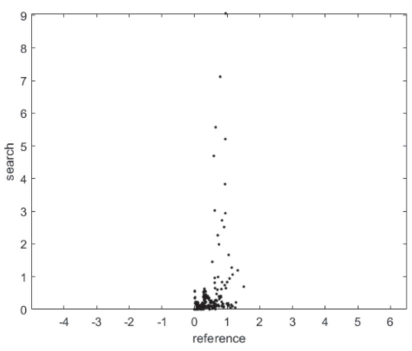

(a)


(b)

(c)



(d)



(e)

Although Figures 4.8a and 4.8b are very similar, their histograms are not identical and there are large variations in the frequency of digital values in Figure 4.8d. This is expected due to differences in lighting, contrast, and brightness, as well as variations in the frame, pedestrian pose, and background. Even with these variations, there is a correlation between histograms, with a correlation coefficient value of 0.77.

Considering the second pair, a comparison of histograms of Figures 4.8a and 4.8c, it is noted that there are large differences, which are reflected in Figure 4.8e. In this case, the variation of the values in image 4.8c is much higher than that of image 4.6a. The correlation coefficient, in this case, is 0.35.

To accept only cases of high correlation, only matches with a correlation coefficient greater than 0.8 were accepted as true.

4.4 EVALUATION OF DETECTION AND TRACKING

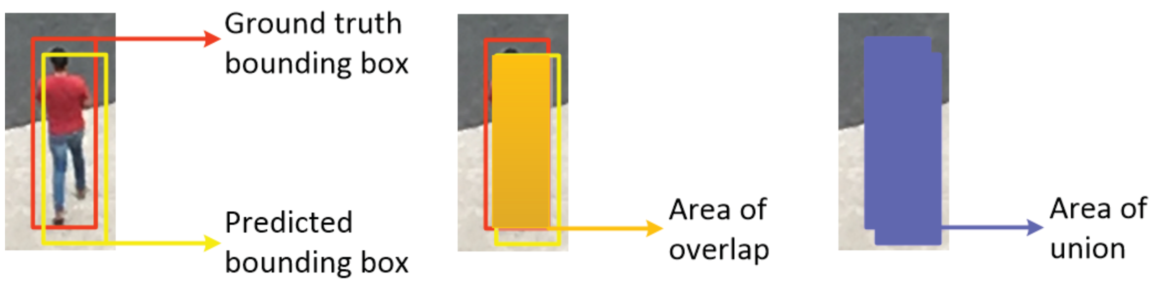
Three types of performance metrics were used in our procedure: metrics based on detection, on segmentation, and based on tracking.

- Detector metrics

To check how anchor regions overlapped with ground truth bounding boxes it is used the intersection of the union (IoU) method. The IoU is the ratio of the overlapped area between the ground truth and predicted area to the total area, or area of union, as shown in Figure 4.9. A threshold is set for a region containing an object as a person or as background. The detection d is considered as true positive whenever its IoU with the closest ground truth *g* is greater than 0.5 (Ren et al., 2015).

$$IoU(d,g) = \frac{|d \cap g|}{|d \cup g|} \begin{cases} \geq 0.5 = pedestrian \\ < 0.5 = background \end{cases} \tag{4.17}$$

FIGURE 4. 9 - THE IoU CONCEPT.



To measure in the detector if the pedestrian is present and where he is located within the image, we report the recall (r), precision (p), and F1-score (F) metrics (Goutte and Gaussier, 2005) computed as:

$$\mathrm{r} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{4.18}$$

$$\mathrm{p} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{4.19}$$

and

$$F1 - score = \frac{2*r*p}{r+p} \tag{4.20}$$

where:
TP is true positive, number of detections with IoU>0.5;
FN is false negative, number of detections with IoU <= 0.5 or detected more than once, and FP is false positive, number of objects not detected, or detected with an IoU <= 0.5.

- Segmentation metrics

Once the pedestrian was segmented, we evaluated the results by comparing pixel-by-pixel two binary images: the automatically segmented and the ground truth. To evaluate performance, the number of True Positives (TP), False Positives (FP), False Negatives (FN), Correct Classified (CC) pixels, and the total number of pixels labelled as pedestrians (TT = CC+FP) are counted, and then the following metrics are derived (McKeown et al., 1999):

$$Bf = \frac{FP}{TP} \tag{4.21}$$

$$Mf = \frac{FN}{TP} \tag{4.22}$$

These two quality parameters describe two types of error in the results, omission, and commission errors, but are relatively simple. Therefore, the detection success was also measured using recall, precision, and the intersection over union (IoU) (Minaee et al., 2020).

$$r = \frac{TP}{TP+FN} \tag{4.23}$$

$$p = \frac{TP}{TP+FP} \tag{4.24}$$

and

$$IoU = \frac{TP}{TP+FP+FN} \tag{4.25}$$

where Branching Factor (BF) is the measure of the ratio of incorrectly labeled pedestrian pixels. Miss Factor (Mf): it computes the rate of missed pedestrian pixels. The recall is a kind of user's accuracy, while precision describes the rate of correct pixels within the set of pixels labelled as a pedestrian. The IoU combines aspects of both measures to summarize algorithm performance.

- Tracker metrics

The performance of the identification of the right pair in a second image was measured by counting the number of correct pairs in comparison to the possible matches. Three cases may occur in this search:

1. TP: the j-region in the second image contains the pedestrian of the i-region of the first image (true positive).
2. FP: the j-region in the second image contains a different pedestrian than the i-region of the first image (false positive).
3. FN: the i-region of the first image is not detected in the second image, although it exists (False negative).

The result of the matching process was compared to a reference result, obtained by visual inspection of the images. This allowed computing metrics to describe the quality. For example, the quality of the result can be evaluated by counting the number of regions correctly identified (True Positive).

The performance is reported by accuracy, precision, mean average precision (mAP), F1-score, and recall. The recall (r), precision (p), and F1-score metrics are computed respectively according to equations 4.18-4.20. The accuracy can be expressed mathematically as:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.26}$$

The mean average precision metric is calculated as the mean of average precision of the object class.

A comparative study was performed using a series of 10 images, considering the use of background suppression and the three similarity measurements. The metrics were computed for each pair of images and also for all the comparisons. This allowed, first, to state when the algorithms failed, identifying the pairs with low performance, and also compare the global accuracy of each experiment, according to Equation 4.27.

$$Q = \frac{\sum_{i=1}^{9} TP(i)}{\sum_{i}^{9} TP(i) + \mathrm{FP(i)} + \mathrm{FN(i)}} \tag{4.27}$$

5 EXPERIMENTS AND RESULTS

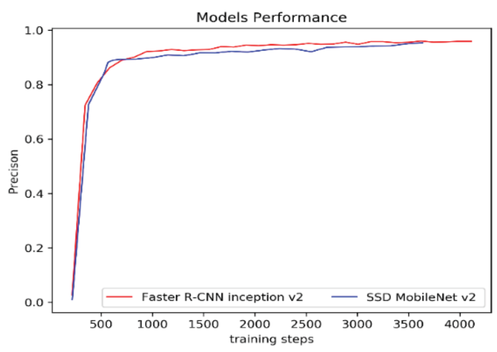
5.1 PEDESTRIAN DETECTION

There are three main work steps in this thesis, where in first Faster R-CNN and SSD topologies are implemented to train a pedestrian detector. The second step of methodology comprises a segmentation process for automatic feature extraction. Finally, tracking is performed using the correspondence method.

In the first step, Faster R-CNN and SSD models are trained to detect pedestrians in the image frames. A self-collected dataset was used for retraining the detectors, having 700 different images containing multiple pedestrians. The data is collected from different urban scenes recorded in the city of Curitiba, Brazil. Transfer learning technique with the pre-trained models provided on TensorFlow model zoo was leveraged to generate the results. In the experiments, the training steps was set to 4000. The training was stopped manually if the loss level did not change in the subsequent 10 epochs (Yao et al., 2007).

Figure 5.1 shows the evolution of precision in relation to the training steps. What emerges is that precision increase with the increase of correct rate of identification and the training steps. After a certain number of training steps, the precision does not increase significantly. That is when the precisions models reach a plateau value. Precision plateaus after 500, and 1000 training steps for SSD and Faster R-CNN models respectively. It is interesting to note that the SSD reaches the plateau earlier than Faster R-CNN. This means that, compared to the competing model, the SSD needs less processing time. This was also noted by Xu (2017).

FIGURE 5.1- PERFORMANCE EVALUATION OF THE MODELS.



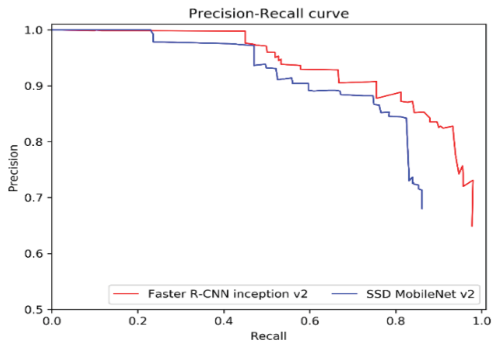
The performance of the detectors models that are directly learned from sample data is listed in Table 5.1. The precision, F1-score, and recall were measured. Upon evaluating the detectors, it can be seen that the SSD Mobilentev2 detector is outperformed by a considerable margin (see Table 5.1) by Faster R-CNN Inceptionv2.

TABLE 5.1- PERFORMANCE OF FINE-TUNED OBJECT DETECTION MODELS.

	Precision	Recall	F1-score
Faster R-CNN Inceptionv2	0.962	0.912	0.927
SSD Mobilentev2	0.891	0.891	0.816

The precision-recall curve is shown in Figure 5.2. The Faster R-CNN model shows a higher precision over the entire recall range, which is also indicated by comparing the average precision values of 0.962 (red) and 0.891 (blue).

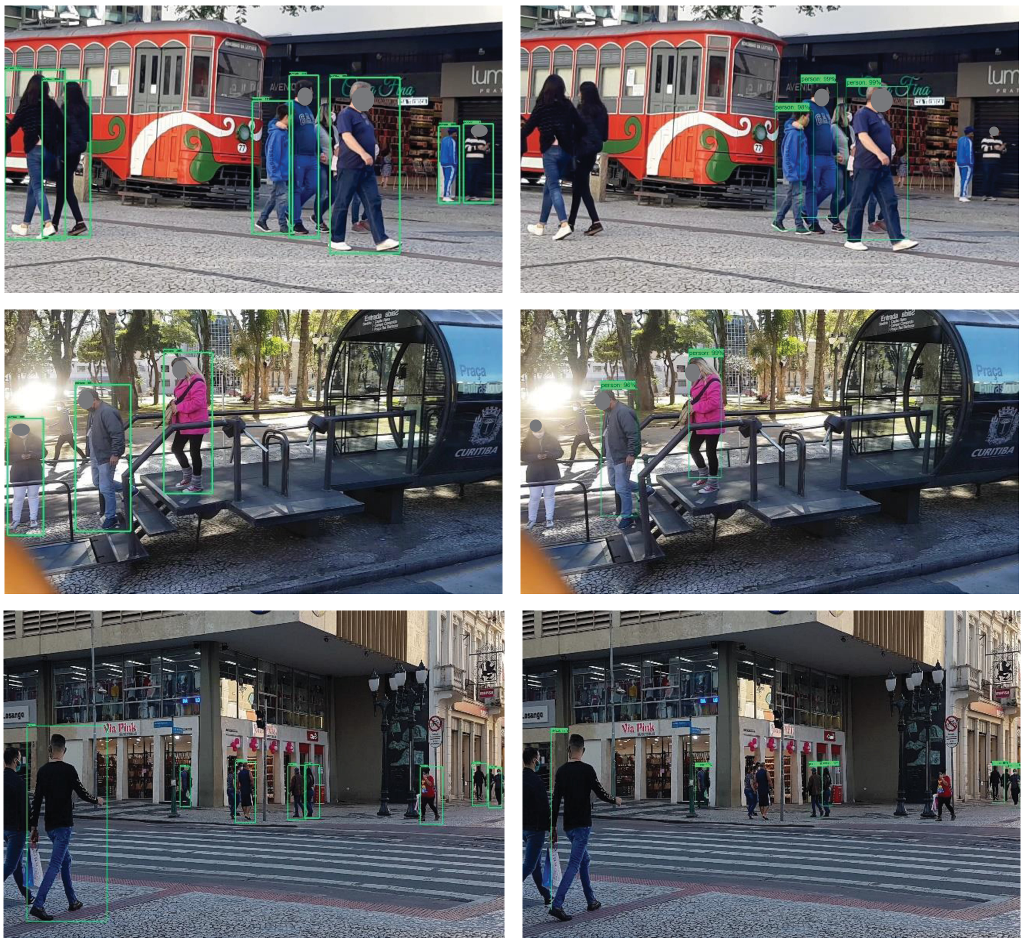
FIGURE 5.2- THE INTERPOLATED PRECISION-RECALL.



Figures 5.3 and 5.4 depict results from testing the detectors on different scenarios from RGB imagery acquired using mobile cameras. The detectors output delimiter boxes for each detected pedestrian (green bounding box in the sample frames), additionally the class label (person) and the confidence score (which is almost 80% in most of the cases). The tested models were able to accurately detect and classify a pedestrian in the image, in some cases, even if the contours of the pedestrian were obscured by other objects, for example, poles, trees, and the roof of the bus stop.

The images in Figure 5.3 represent some of the scenarios with the variation of pedestrian’s scales in the scene from test sample data. The columns on the left are the result from Faster-RCNN detector and the right from the SSD. Some of the worst results produced by the models are presented in Figure 5.3. In the first example, the pedestrians in the centre of the images are partially occluded (row 1), and part of pedestrian body is missing at the borders of the image (row 3). These were not counted as pedestrians. Apart from such difficulties, in general what can be seen from the results is that the SSD approach fails when the pedestrians appear on different scales.

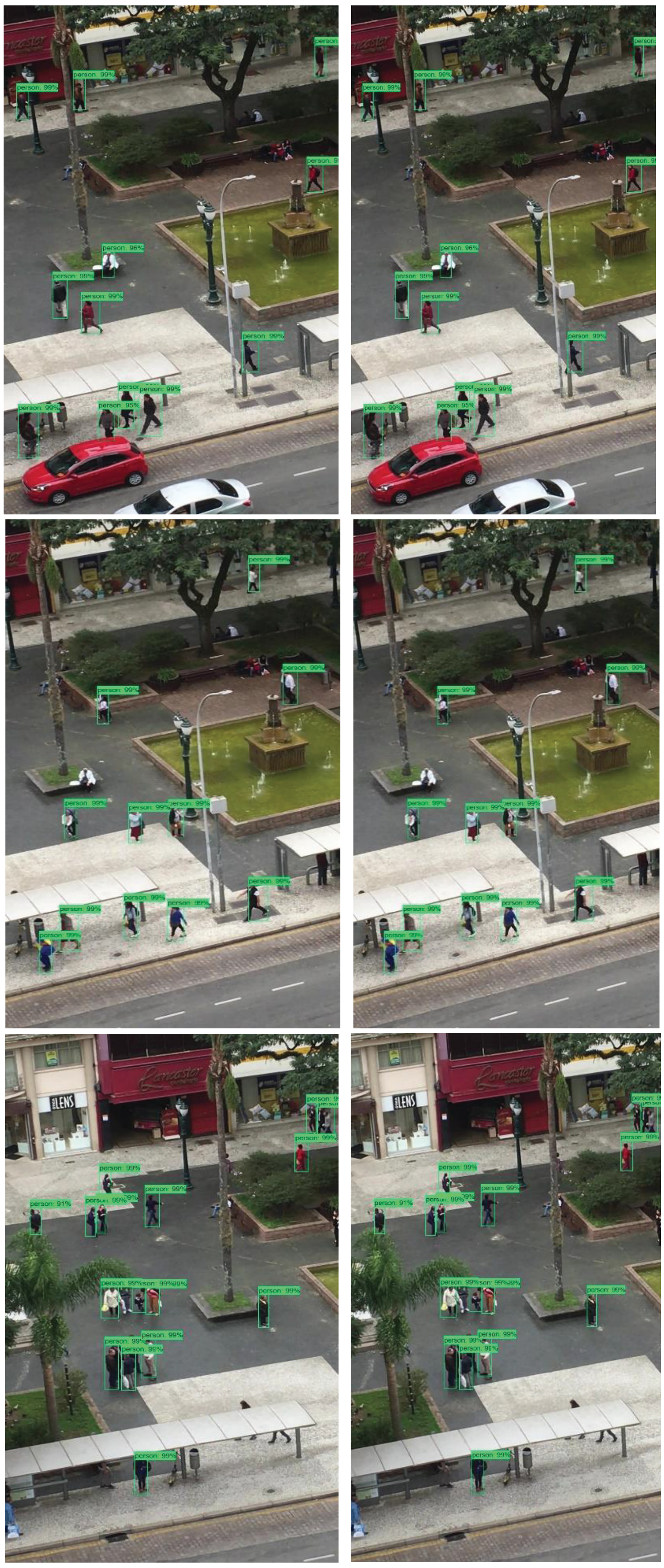
FIGURE 5.3 - DETECTORS OUTPUT ON DIFFERENT TEST IMAGE FRAMES.



(a) Faster R-CNN (b) SSD

Figure 5.4 demonstrates the robustness of the detectors even when pedestrians are partly occluded or stand too close to others. Some of the cases are pedestrians standing at the bus stop, or under trees. Others are standing in front of a similarly coloured background. The background where the pedestrians move varies from dark to light. In this case, it is difficult to detect all pedestrians present in the image, even by humans. Yet, the detectors were able to classify and detect pedestrians even if they were not fully shown in the image scene. For example, at the bottom and middle-left of the images (row 3), and in the lower-right corner of the images (row 2), only partial contours of the pedestrian were shown. Nevertheless, the detectors were able to accurately detect and classify almost all pedestrians present in the scene as a “person” class. In the scenes of Figure 5.4, the two detectors perform similarly in terms of performance, however, Faster R-CNN detects with relatively high confidence.

FIGURE 5.4 - DETECTORS OUTPUT ON DIFFERENT TEST IMAGE FRAMES.



(a) Faster R-CNN (b) SSD

The experimental results illustrate that although acceptable performance detection is achieved, some failure cases persist in both fine-tuned models (Figure 5.3 and 5.4). The ability of the fine-tuned models for object detection can be improved in several ways. For example, one can increase the training dataset, or improve the architecture model to enhance the performance and generalization ability of the model (Zhai et al., 2020).

Indeed, it is evident that the Fast R-CNN detects pedestrians with more accuracy, clearly with a greater processing time than the SSD. This is because Faster R-CNN comprises two sequential stages, the first one to propose regions, followed by the second one that classifies the proposed regions. The other reason is the huge difference in the number of parameters. For example, the fully connected layers on Faster R-CNN are 15 times bigger than the whole SSD network. As a direct consequence, the SSD has a lower processing time, which is a useful advantage when designing applications in embedded systems, mainly because the model handles object detection as a regression problem.

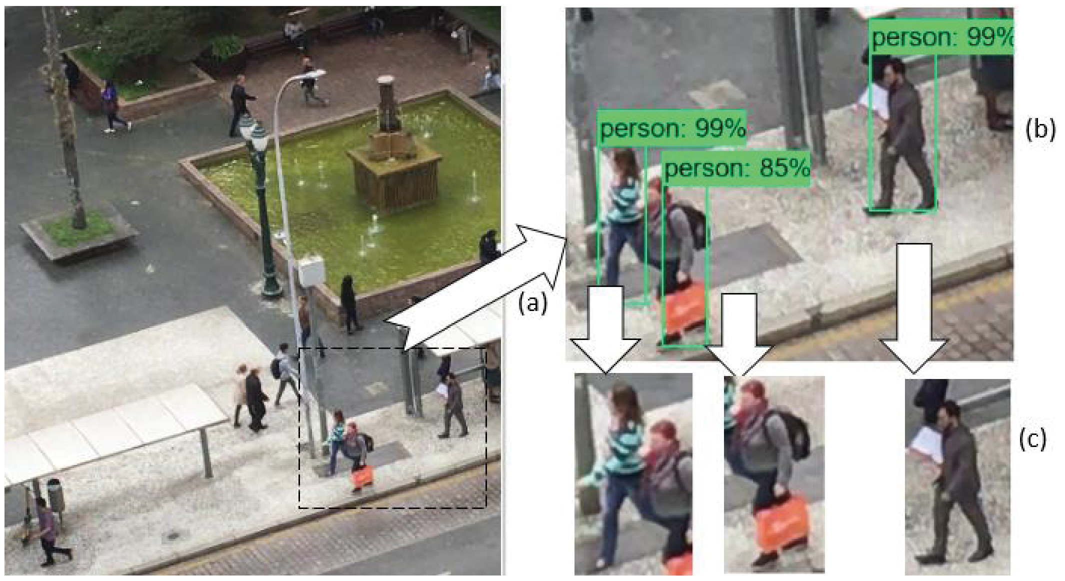
The importance of data set pre-processing, the computational costs, will have, together with the choice of hyperparameters, a major impact on how well the fine-turning will work. Although some parameters are used in both models, each has its specific ones, too. In both models, the following hyperparameters were found to be optimal in fine-tuning: Learning Rate (LR) equal to 0.0001, a Momentum (M) equal to 0.9, and a Batch Size (BS) of 1. LR influences the training time and precision. If one specifies a small learning rate, like for example 0.0001, the training will be longer but it could increase the overall precision. If a high value is chosen, like for example 0.95, then the training will be faster at the cost of a reduction in precision and increase of instability. The batch size value was defined as part of a strategy to adapt to the limitation of used graphic card memory. Additionally, to decrease the computational costs and reduce overfitting, an early stopping criterion (Yao et al., 2007), was implemented. By this criterion the training process is interrupted when validation accuracy does not improve for 10 subsequent epochs.

The purpose of the research in this phase was to study the applicability of deep-learning-based models for the detection of pedestrians in an outdoor environment to find the optimal model for applications in the next phase, which is tracking. So, it was found (in the present application) that Faster R-CNN was the best performing model, thus, its output was used as input for feature extraction for the pedestrian tracking phase.

5.2 PEDESTRIAN SEGMENTATION

In the following phase, the aim was to separate pedestrian figure from the background to extract features for tracking purposes. Recall that in the detection phase, the regions of interest on an image that contain a pedestrian were extracted. Having the region of interest, the pixel values describing the pedestrian body are separated from overall pixels. An example of this process is displayed in Figure 5.5.

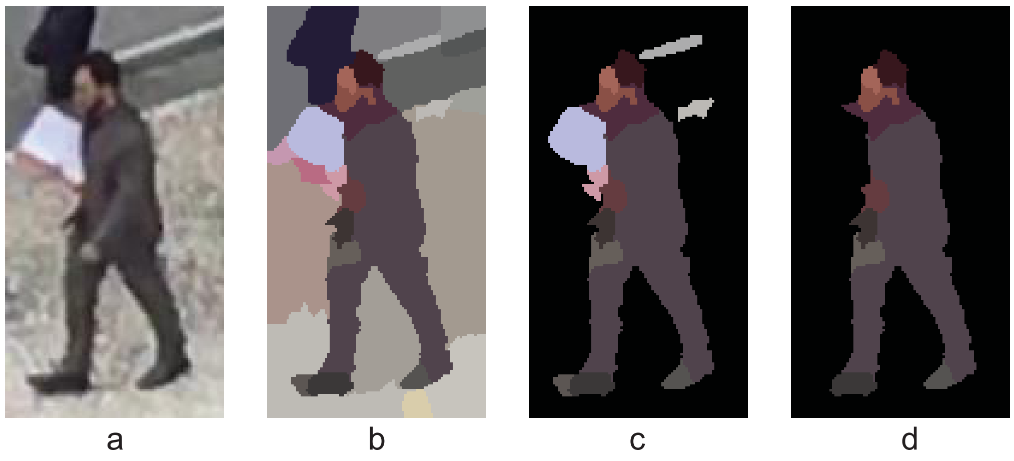
FIGURE 5.5 - EXAMPLE OF A NATURAL IMAGE USED IN THE EXPERIMENTS.



In Figure 5.6, the different steps of the isolated pedestrian from Figure 5.5c are shown. The first image is the original RGB input image. Note that the person seems to hold a white object, but it can be also understood as part of the background. In the second image, the result of the initial segmentation and clustering is displayed. The image has a small number of segments and one can note that the borders of the person are relatively "well" delineated in the segmentation. The third image (Figure 5.6c) is the result of the initial classification. It is noticeable that the classification based on the density fields is good, but some mistakes are visible, like the inclusion of two segments that are part of the background (street) in the upper right corner. The result also includes the object that the man holds in his right hand. This can be an error but also correct, depending on the manual reference segmentation. Figure 5.6d displays the result after improvements obtained in the relaxation step. Comparing Figure 5.6c and

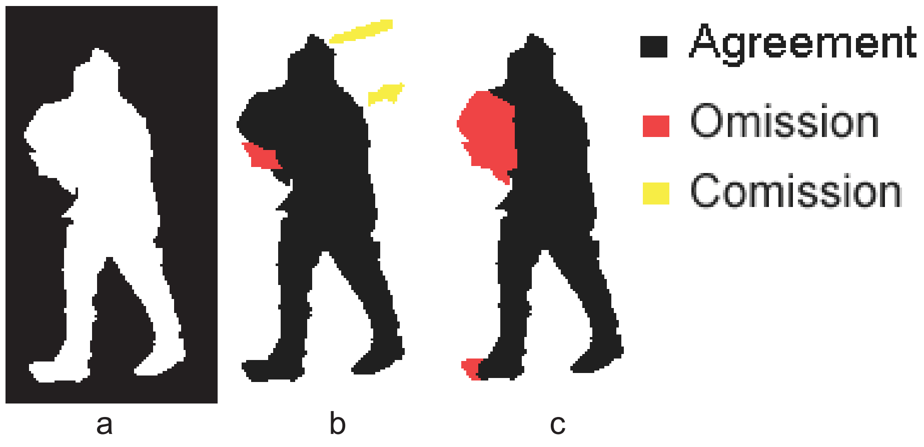
5.6d it is visible that errors like the light segments on the upper right corner and, a part of the foot of the pedestrian were removed. The white object the man carries is also removed. This can be attributed to the fact that the white object is more like the ground than to the person.

FIGURE 5.6 - EXAMPLE OF DIFFERENT STEPS: a) ORIGINAL (171X91) RGB IMAGE; b) SEGMENTED IMAGE; c) CLASSIFIED IMAGE; d) RESULT IMPROVED BY RELAXATION.



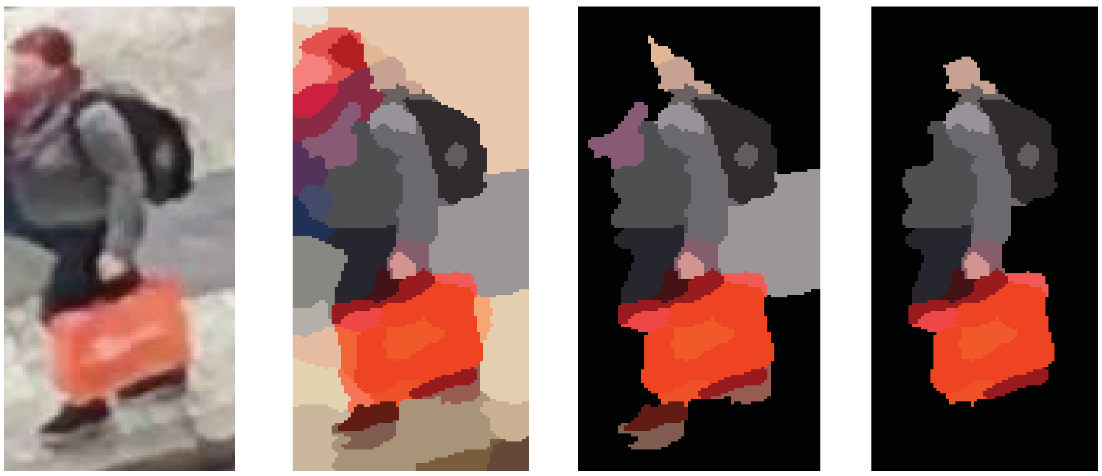
The Results can be compared to a manually obtained reference, as illustrated in Figure 5.7, where the reference image is displayed as a binary image. The commission errors are displayed in yellow and the omission errors in red. Figure 5.7b displays the comparison of the result of the first segmentation and the clustering. For the computation of the quality parameters, the total pixels omission (red) and commission (yellow) were counted. The black area displays the pixels that were correctly classified (agreement).

FIGURE 5.7- EXAMPLE OF COMPARISON OF THE RESULTS.



The second example is displayed in Figure 5.8. The woman in the Figure 5.8 also holds an object, an orange bag. It can be noted that the classification (based on the model of a side view of a person) did not perform well. Parts of the pedestrian as her head were lost. In the right portion of the segmented image, the ground is also included. After the reclassification based on relaxation, the result is partially improved due to the removal of the ground, nevertheless, other parts of the pedestrian like the feet are lost.

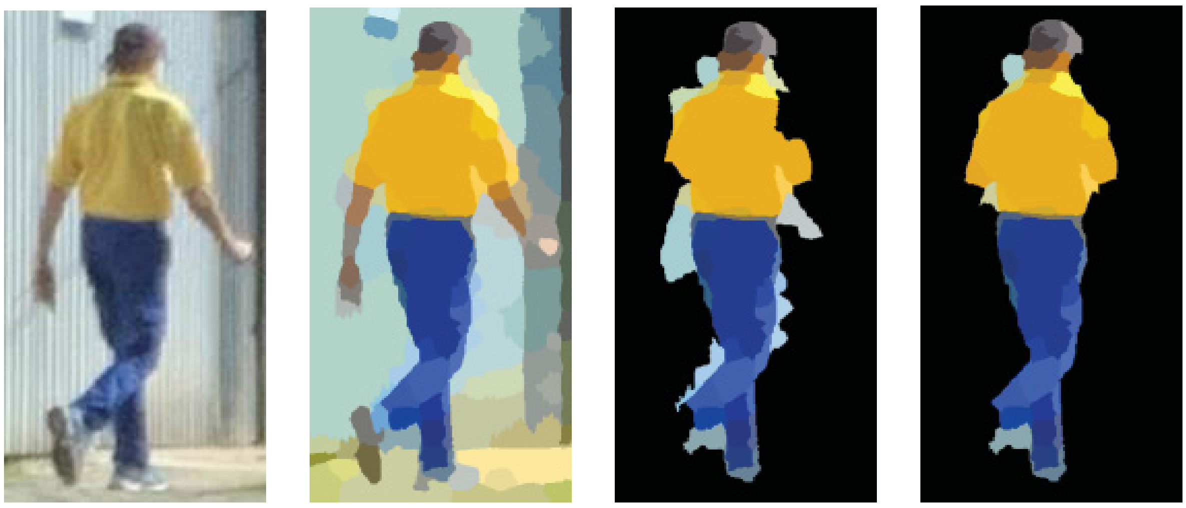
FIGURE 5.8 - EXAMPLE OF A WALKING WOMAN.



The third example is how relaxation can improve the classification. Figure 5.9 (column 3) displays the initial classification, which is not bad, but there are remaining some segments of the background. This happens because they are in the center of the image, where the density fields are high and consequently the mean value of the segments is also high. The relaxation step can remove such segments, as their compatibility with other background regions is higher than with the foreground regions.

A common fact seen here and in other images is that the arms or feet are not included. This is because the density fields cannot model the position of the arms and feet. The probability fields (Figure 4.2) do not include a good description of the arms due to their variable position. In the examples, it is noticed a tendency to confuse skin with the background. This happens due to the lack of saturation of the skin color. As the saturation is low, it is easily confused with gray tones, that are frequent in the background.

FIGURE 5.9 - EXAMPLE OF A WALKING MAN.



To evaluate the result with a reasonable set of images, five series of images were used. From each series, ten samples were collected so that fifty images were analyzed. For each image, the Branch Factor, Miss Factor, Recall, Precision, and IoU were computed. First, are presented results without the relaxation step. Table 5.2 summarizes the results of the success of the classification method considering the quality parameters of Branch factor, Miss factor, Recall, Precision, and IoU.

TABLE 5.2 - PERFORMANCE OF THE CLASSIFICATION MEASURED BY FIVE QUALITY PARAMETERS (%): BRANCH FACTOR (BF), MISS FACTOR (MF), RECALL, PRECISION AND IoU.

	Branch factor	Miss factor	Recall	Precision	IoU
Mean	28.91	36.80	82.04	79.90	67.48
Minimum	2.33	1.31	19.14	47.23	17.87
Maximum	111.73	>100	98.71	97.73	93.88

Table 5.3 displays the same statistics computed for the final image, after the probabilistic relaxation step. A comparison of these tables reveals that the results can be improved by the relaxation step. The mean values of the Miss and Branch factors are lower after the relaxation step. This indicates that errors are removed. The Branch factor without relaxation ranges from 29 to 111%, and between 15 and 59% after the relaxation, for example.

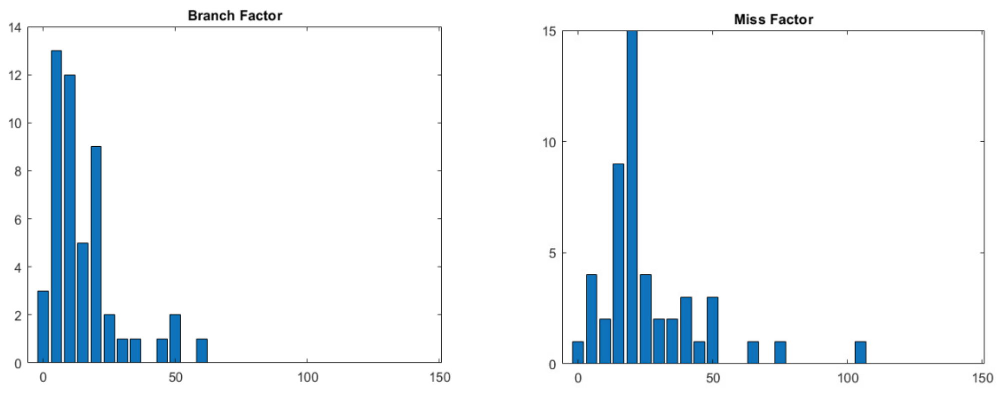
The same improvement is visible when comparing values of precision, recall, and IoU. The statistics after relaxation have smaller ranges and the mean values are higher. It is interesting to analyze the range of these parameters because it indicates that extreme low values are corrected. Based on the comparison made, it can be stated that relaxation introduces improvements in the results.

TABLE 5.3 - PERFORMANCE AFTER THE IMPROVEMENT BY RELAXATION, MEASURED BY FIVE QUALITY PARAMETERS (%): BRANCH FACTOR, MISS FACTOR, PRECISION, RECALL AND IoU.

	Branch factor	Miss factor	Recall	Precision	IoU
Mean	15.12	31.97	79.72	87.86	71.64
Minimum	0.00	2.09	23.50	62.57	23.22
Maximum	59.81	>100	97.95	100.00	87.36

It is also important to analyze the statistics displayed in table 5.3 with the support of the histograms of the quality parameters displayed in Figures 5.10 and 5.11.

FIGURE 5.10 - HISTOGRAMS OF THE BRANCH AND MISS FACTORS OF 50 SAMPLES.



The histograms of the Branch and Miss factors prove that the values concentrate on low values, although some high values are still present. The proposed method has limitations in solving all cases and, therefore, the errors are reflected as high values. Analyzing the histograms displayed in Figure 5.10, it is noticeable that the Miss factor is higher than the Branch factor, which reveals a tendency to miss segments in the result. In some cases, the Branch factor is almost zero, showing that the result is fully compatible with the reference image. Nevertheless, some bad results are also obtained, as it is indicated by the maximum Branch factor of 60%.

The statistics of recall and precision can also be analyzed with the support of the Histograms displayed in Figure 5.11. The recall describes the rate of correct pixels within the total set of labelled pixels. The computed values concentrate around 80% and can be considered high. There is a pair of bad results that caused the mean to be lower, showing that the method is not able to solve all the problems. It is worth to quote here that the density fields used are not able to model all possible poses and therefore not all the images can be correctly classified. The mean rate of correct pixels within the set of labelled pixels (Correctness), like the producer's accuracy concept is 82%, and some results have achieved values closer to 100%.

FIGURE 5.11 - HISTOGRAMS OF THE RECALL AND PRECISION INDEXES OF THE 50 SAMPLES.

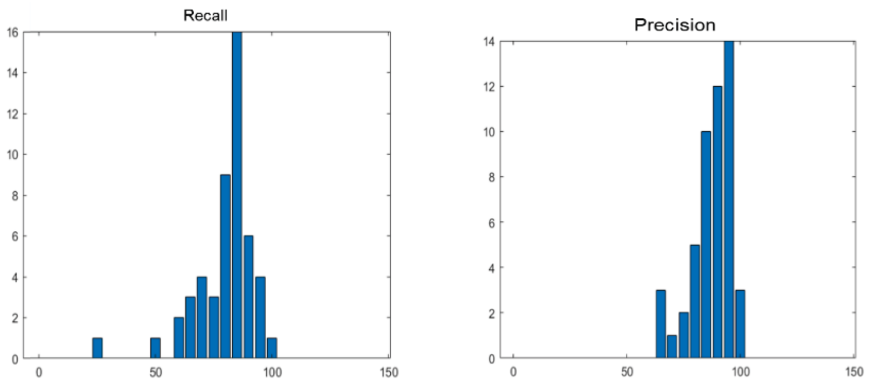
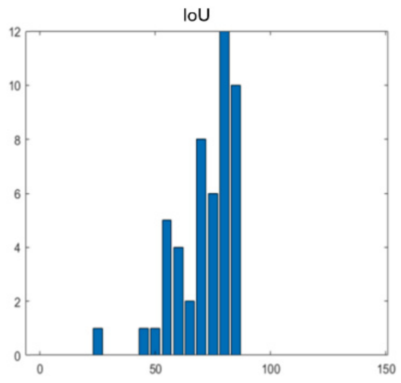


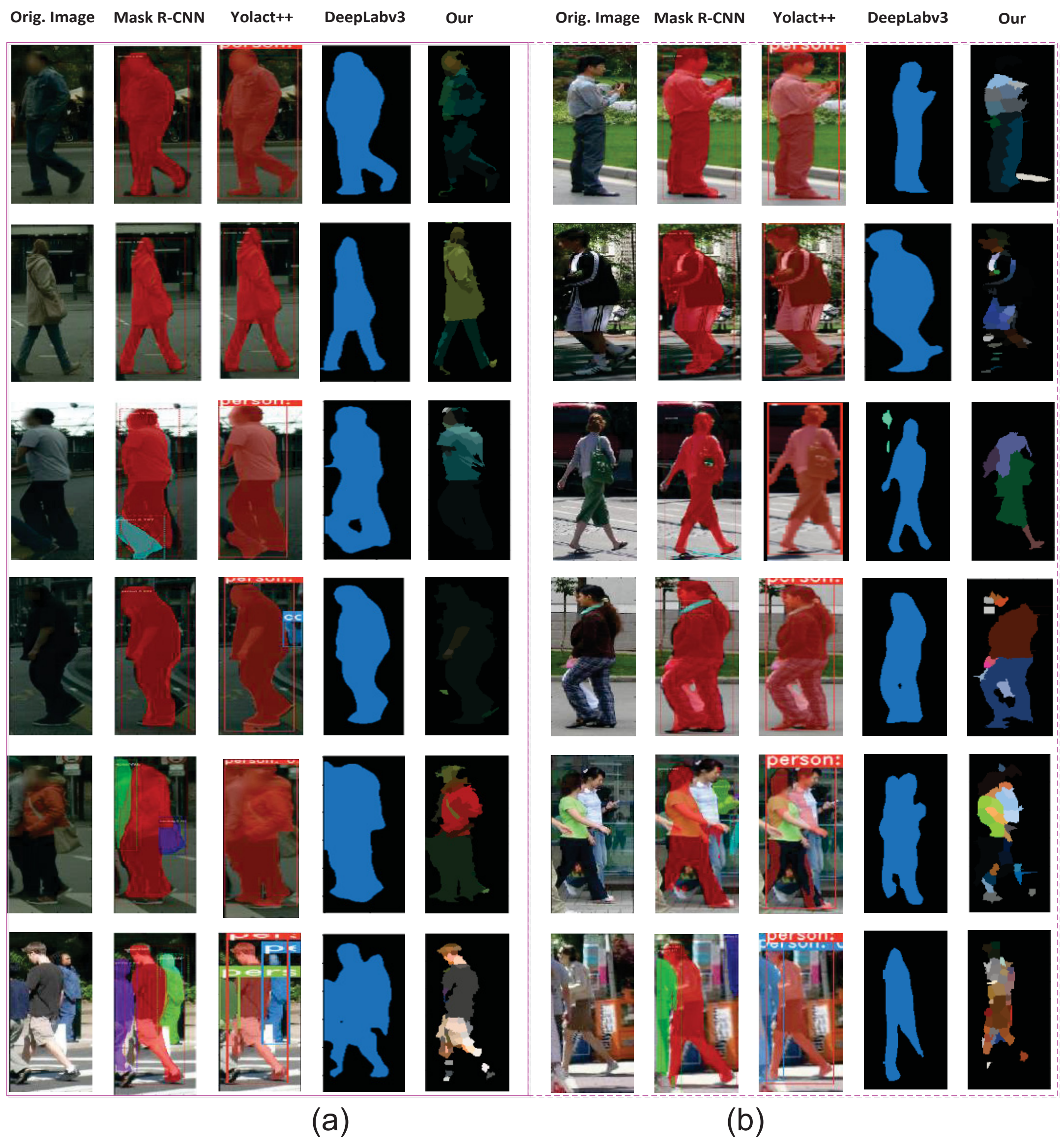
Figure 5.12 displays the variation of the IoU index. It is visible that the values are concentrated and close to 100, indicating good quality. The mean value is 71%, as some parts of the human figure are lost. Nevertheless, the quality is good and proves that the method can be used to separate pedestrians from the background even when the background changes from image to image.

FIGURE 5.12 - HISTOGRAMS OF THE IoU INDEX COMPUTED FROM 50 SAMPLES.



Apart from the experiments on high-resolution RGB images acquired through mobile cameras, we also tested the proposed method on two different datasets and made a comparison with some of the state-of-the-art methods. Cityscape dataset (Cordts et al., 2015), and Penn-Fudan dataset (Wang et al., 2007) are some acceptable benchmarks for human segmentation. Cityscapes is comprised of a large dataset of real images, containing high-quality pixel-level annotations for 5000 images collected in street scenes from 50 different cities. Penn-Fudan benchmark contains 170 color images with 345 boxes/shape-labelled pedestrians. We compare the proposed method against three deep learning-based segmentation models: Mask R-CNN (He et al., 2017), Yolact++ (Bolya et al., 2019), and DeepLabv3 (Chen et al., 2017). The results are obtained via finetuning their publicly available codes.

FIGURE 5.13 - COMPARISON OF SEGMENTATION RESULTS BETWEEN THE PROPOSED METHOD AND OTHER METHODS ON (a) CITYSCAPE AND (b) PENN-FUNDAN DATASETS. THE RED PIXELS/REGIONS IN MASK R-CNN AND YOLACT++ BELONG TO PEDESTRIANS.



Mask R-CNN, Yolact++, and DeepLabv3 models are fully trainable end to end. For comparison purposes, the models were fine-tuned under the same experimental environments on Cityscapes and Penn-Fudan datasets using pre-trained weights learnt on the COCO dataset. Mask R-CNN and DeepLav3 fine-tuned models uses resnet 101 as backbone while Yolact++ uses resnet 50 as a backbone.

Figure 5.13 displays the segmentation results on the two benchmarks. Figure 5.13a contains six images from the Cityscapes dataset while Figure 5.13b from the Penn-Fudan dataset. The first columns in each subfigure show the input data, and the following columns display respectively the results of segmentation performed with the Mask R-CNN, Yolact++, DeepLabv3, and the proposed method. The first four rows contain single human images, while the two last rows contain more than one or occluded human figures.

The results of segmentation on both benchmarks present the same characteristics: when a single human image is presented to the models, the human instance is segmented nearly perfect. In some cases, when two or than one human is present, or occluded the models do not segment properly all of them. In these examples, it could be caused due to the low exposure of the images and the mix between dark clothing and the environment. Additionally, the models sometimes fail to segment certain body parts (see the last two rows of Figure 5.13). This is a classical challenge in many segmentation algorithms including the proposed one. Overall, from the experimental results shown in Figure 5.13, we can see that our method performs comparatively well in a challenging background.

Table 5.4 shows the performance indexes of the proposed method and the standard deep learning algorithm on Cityscapes and Penn-Fudan datasets. Looking at the numbers in Table 5.4, one can see that DeepLabv3 has comparatively low metric values in both evaluated benchmarks, the lowest values of precision, recall, and IoU being respectively 79.4%, 70.8%, and 72.9% (on Penn-Fudan dataset). The proposed method achieved metric values that are close to those obtained with Yolact++. By analyzing IoU values, the proposed method far outperforms the Yolact++ (by 1.3% and 0.9%) in evaluated benchmarks. The same difference and ranking are verified when comparing the precision and recall indexes. These two methods can be considered equivalent.

TABLE 5.4 - PERFORMANCE COMPARISON OF THE PROPOSED METHOD AND OTHERS ON CITYSCAPES AND PENN-FUDAN DATASETS.

Dataset	Method	Precision	Recall	IoU
Cityscapes	Mask R-CNN	85.70	86.50	86.01
	Yolact++	89.69	89.15	90.30
	DeepLabv3	84.50	82.20	79.10
	Proposed method	91.60	90.30	91.60
Penn-Fudan	Mask R-CNN	81.60	82.80	78.70
	Yolact++	88.40	87.60	87.20
	DeepLabv3	79.40	70.80	72.90
	Proposed method	88.60	89.58	88.10

Overall, the results show that, without the need for the fine-tuning process, our approach achieves a similar performance or outperforms the one that requires finetuning. One reason is that when the evaluation scenarios are very different from the training data, fine-tuning based methods may suffer from confusion between instances and their performance may degrade substantially. Recall that fine-tuning is expensive as it requires a large amount of well-annotated data for training the model, which is potentially not practical in real-world applications. In contrast, the proposed method employs a pre-defined pose without any further fine-tuning. Despite that, it produces promising segmentation results on two benchmark datasets (see Table 5.4 and Figure 5.13). This demonstrates the potential of the proposed method since it is very different from some existing methods (such as Mask R-CNN, Yolact++, and DeepLabv3) which require careful fine-tuning of the pre-trained model for better results.

5.3 PEDESTRIAN TRACKING

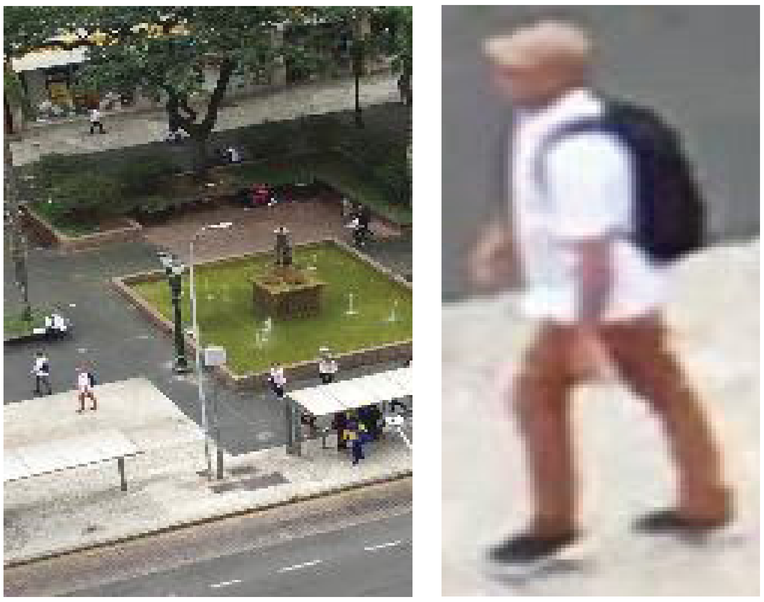
The tracking algorithms were evaluated under two conditions, with and without background suppression, and considering different frame intervals. So, the results are organized as follows: First, the frames were grabbed at different intervals. Every five, ten, and twenty frames, to evaluate the possibility of reducing the processing effort needed to track a pedestrian. Then, the series tracking methods were applied using regions of interest with and without the background suppression.

5.3.1 Varying interval

In this experiment, frames were grabbed every 5, 10, and 20 frames, which corresponds to intervals of 0.167, 0.333, and 0.667 seconds. Although such intervals seem to be small, the pose and position differences can be significant.

Figure 5.14 displays the first image of this series and an example of a region of interest.

FIGURE 5.14 - FIRST FRAME AND AN EXAMPLE OF A REGION OF INTEREST



5.3.2 Five frames without spatial constraint

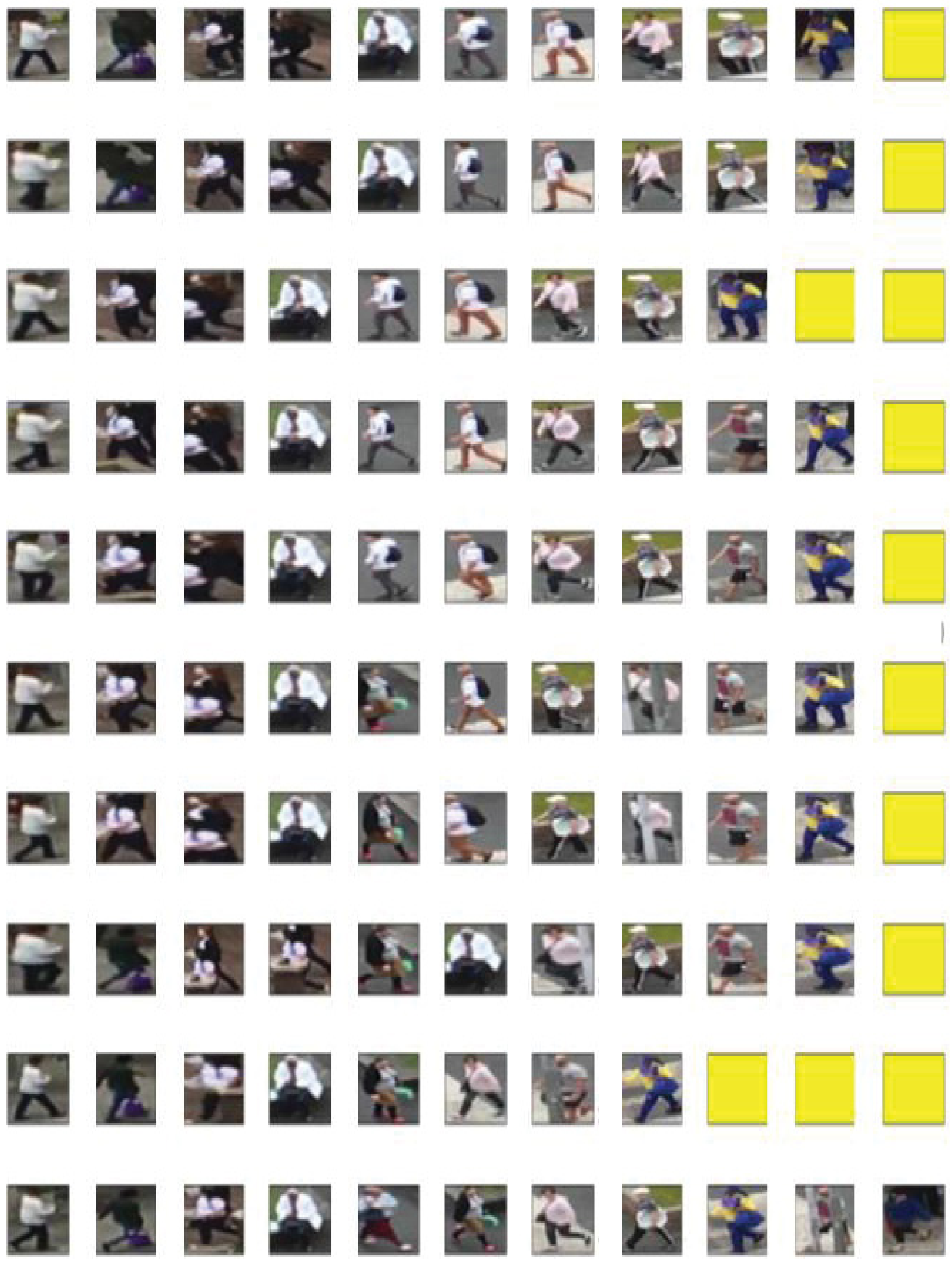
For the first experiment, one image was extracted every five frames of the video. For the estimation of quality, the first 9 pairs are analyzed. The number of regions per image varies, depending on the entry and exit of pedestrians in the scene, nor as to their occlusion, as shown in Table 5.5.

TABLE 5.5 - EXAMPLE OF NUMBER OF REGIONS PER IMAGE.

image	1	2	3	4	5	6	7	8	9	10
regions	10	10	9	10	10	10	10	10	8	11

The complete set of regions of interest is displayed in Figure 5.15.

FIGURE 5.15 - SET OF REGIONS OF INTEREST OF EACH IMAGE. ROW=FRAME, COLUMN=REGION.



The reference matches obtained by visual analysis of each pair of consecutive images are shown in Table 5.6. In this table, each row displays the pair of images and the column the number of the region. As the number of regions varies, some cells contain a null value. Each element of the table shows the correct pair in the following image (search). For example, when comparing the second and third images, it is noted that the first region in the second image corresponds to the first region of the second. The zero in the third column denotes that the second region of the second image does not exist in the third image.

TABLE 5.6 - GROUND TRUTH OF EXPERIMENT 1.

Region	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Image pairs										
1-2	1	2	3	4	5	6	7	8	9	10
2-3	1	0	2	3	4	5	6	7	8	9
3-4	1	2	3	4	5	6	7	8	9	0
4-5	1	2	3	4	5	6	7	8	9	10
5-6	1	2	3	4	0	6	8	7	9	10
6-7	1	2	3	4	5	6	7	8	9	10
7-8	1	0	3	6	5	0	8	7	9	10
8-9	1	2	0	0	5	4	6	0	7	8
9-10	1	2	3	4	0	7	10	9	0	0

In the first experiment, all possible matches were considered, without taking into account the spatial distance between the regions of interest.

(i) Euclidean Distance (ED)

Table 5.7 shows the result of the analysis of correspondence between regions of consecutive images based on the ED.

TABLE 5.7- MATCHES WITH THE ED.

Region	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Image pair										
1-2	1	2	3	4	5	6	7	8	9	10
2-3	1	0	2	3	4	5	6	7	8	9
3-4	1	2	3	4	5	6	7	8	10	0
4-5	1	2	3	4	5	6	7	8	9	10
5-6	1	2	0	4	6	8	8	7	8	10
6-7	1	2	3	4	5	9	7	8	8	10
7-8	1	4	1	6	5	10	8	10	9	10
8-9	1	2	1	3	1	4	8	1	7	8
9-10	1	2	3	4	1	7	10	9	0	0

The obtained matches were compared to the expected ones, which enabled computing the number of false positive, false negative, and true positive cases, as displayed in Table 5.8. Here, 1 stands for true positive, 2 for false positive, and 3 for false negative.

TABLE 5.8 - COMPARISON OF THE OBTAINED RESULTS WITH THE ED. TRUE POSITIVE (TP)=1; FALSE POSITIVE (FP)=2; FALSE NEGATIVE (FN)=3.

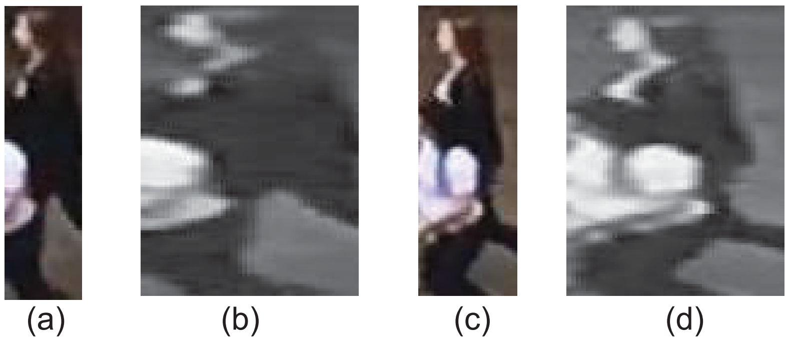
Region	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**					
Image pair											Tot	TP	FP	FN	Acc.
1-2	1	1	1	1	1	1	1	1	1	1	10	10	0	0	1.00
2-3	1	0	1	1	1	1	1	1	1	1	9	9	0	0	1.00
3-4	1	1	1	1	1	1	1	1	1	0	9	9	0	0	1.00
4-5	1	1	1	1	1	1	1	1	1	1	10	10	0	0	1.00
5-6	1	1	3	1	0	2	1	1	2	1	9	6	2	1	0.67
6-7	1	1	1	1	1	2	1	1	2	1	10	8	2	0	0.80
7-8	1	0	2	1	1	0	1	2	1	1	8	6	2	0	0.75
8-9	1	1	0	0	2	1	2	0	1	1	7	5	2	0	0.71
9-10	1	1	1	1	0	1	1	1	0	0	7	7	0	0	1.00
Sum											79	70	8	1	

The performance statistics are all high. The accuracy and precision are equal to 0.886 and 0.897, while the recall and F1-Score metrics reach respectively 0.986 and 0.939.

Although the number of regions varies in each image, it is noted that the use of the ED allows identifying correct pairs of regions. A maximum of two errors was counted, for example when comparing images 7 and 8, of a total of 79 regions, 70 were correctly identified, something around 89%.

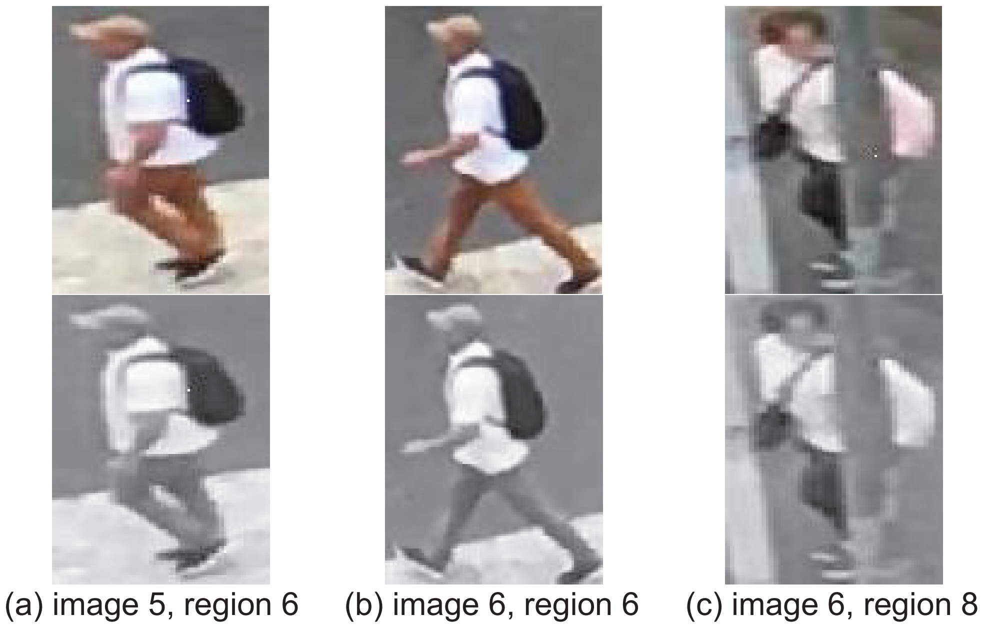
One factor that often causes errors is occlusion. This can be illustrated in the comparison between images 5 and 6. As shown in Figure 5.16, the detected person is partially hidden by a second person in front of them. As two pedestrians cross, one of them is occluded and the regions become more different. In Figure 5.16d, the image has many more white areas than the image of Figure 5.16b. This was an expected problem and greatly affects the ED. This situation caused the algorithm not to detect the region, producing a zero value in the fourth column (region 3 of pair 5-6).

FIGURE 5.16 - EXAMPLE OF THE EFFECT OF OCCLUSION.



Other errors stem from color loss. For example, in the same pair, region 6 of image 5 was identified as region 8 of the sixth image. Figure 5.17 shows these clippings. Note that the correct pair is the sixth region of image 6. However, this similarity is strongly supported by the similarity of the hue of the pixels, especially the pants. When comparing the gray level images, the situation changes because the person's body is composed of white pixels and dark gray areas. Such gray levels also occur in the gray-level image in the eighth region. Comparing the correct pair of regions (a and b), it is noted that they have some significant differences, mainly due to the background and shadows.

FIGURE 5.17 - EXAMPLE OF THE ERRORS DERIVED FROM COLOR SIMPLIFICATION.



As the comparison is performed based on the histograms it is worth looking at them.

FIGURE 5.18 - HISTOGRAMS OF THREE REGIONS.

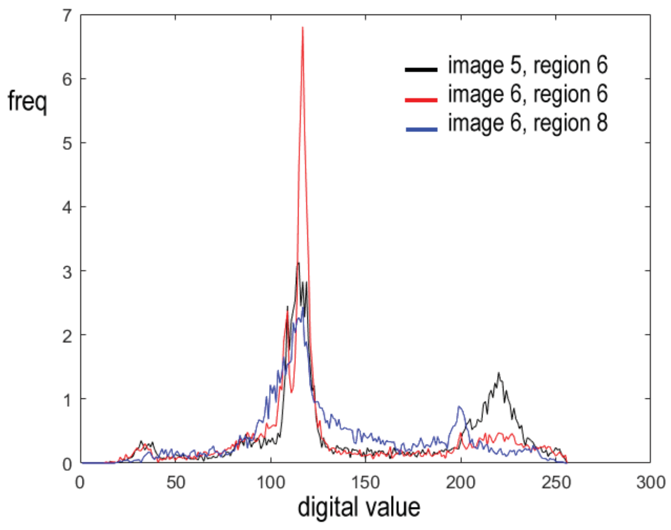


Figure 5.18 displays the histograms of the three regions displayed in the previous figure. When analyzing the region around the gray value 120, it is visible that the histogram of the ideal match (red) presents a high peak, causing larger distances, while the histogram of the eighth region is closer to the original histogram. The Euclidean distance between the correct match is: Euclidean Distance (5-6, 6-6) = 7.54, but the shortest distance is associated with the eighth region Euclidean Distance (5-6, 6-8) = 5.77.

(ii) Dynamic Time Warping (DTW)

Table 5.9 displays the results obtained using the dynamic time warping approach.

TABLE 5. 9 - MATCHES COMPUTED WITH THE DTW APPROACH.

Region	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Image pair										
1-2	1	4	3	4	5	6	7	8	9	10
2-3	1	5	2	3	4	5	6	7	8	9
3-4	1	2	3	4	5	6	7	8	10	0
4-5	1	2	3	4	5	6	7	8	9	10
5-6	10	2	0	4	6	8	8	7	9	10
6-7	1	2	2	4	1	9	7	8	8	10
7-8	4	1	1	6	5	9	8	9	9	10
8-9	2	2	1	3	5	4	8	1	7	8
9-10	1	2	3	4	6	7	8	9	0	0

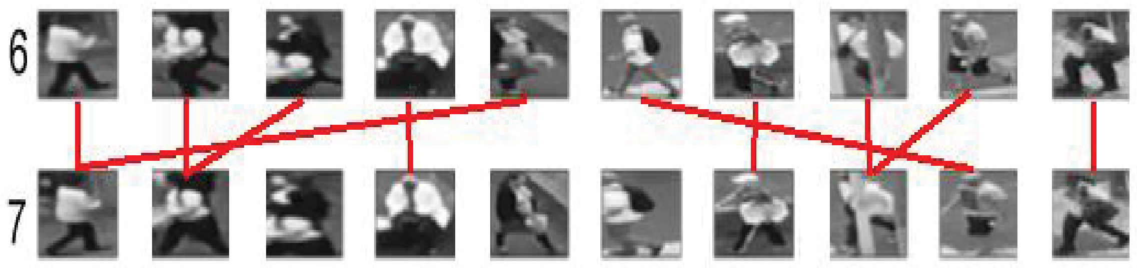
Again, the results obtained were compared to the reference matches, as displayed in Table 5.10. The overall performance of this method is lower than that obtained with the ED. Only 65 regions were correctly identified and the error rate was higher. Errors occur especially between images 5-6 and 6-7.

TABLE 5.10 - COMPARISON OF THE OBTAINED RESULTS WITH THE DTW.

Region	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**					
Image pair											Tot	TP	FP	FN	Acc.
1-2	2	1	1	1	1	1	1	1	1	1	10	9	1	0	0.90
2-3	1	0	1	1	1	1	1	1	1	1	9	9	0	0	1.00
3-4	2	1	1	1	1	1	1	1	1	0	9	8	1	0	0.89
4-5	1	1	1	1	1	1	1	1	1	1	10	10	0	0	1.00
5-6	1	1	2	1	0	2	2	1	1	1	9	6	3	0	0.67
6-7	1	1	2	1	1	2	1	1	2	1	10	7	3	0	0.70
7-8	2	0	2	1	1	0	1	2	1	1	8	5	3	0	0.63
8-9	2	1	0	0	1	1	2	0	1	1	7	5	2	0	0.71
9-10	1	1	1	1	0	1	2	1	0	0	7	6	1	0	0.86
Sum											79	65	13	1	0.822

The DTW algorithm failed to find matches that visually seem easy, as shown in Figure 5.19. For example, the pair of the third region of the sixth image was expected to be the third region of the seventh image, but the algorithm pointed to the second region based on the histograms. The overall accuracy was therefore lower. The recall is equal to 1 while the F1-score is 0.903. The accuracy and precision are equal to 0.823.

FIGURE 5.19 - EXAMPLE OF MATCHES WITH THE DTW APPROACH.



(iii) Correlation

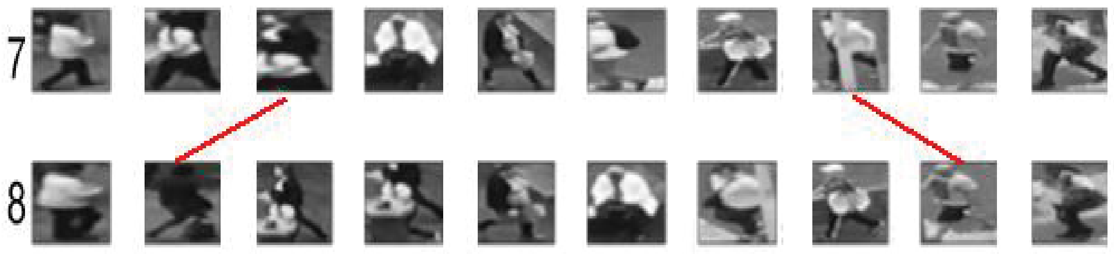
A similar analysis was performed using the correlation to compare image pairs. The results are summarized in Table 5.11.

TABLE 5.11- COMPARISON OF THE OBTAINED RESULTS WITH CORRELATION.

Pairs	Tot	TP	FP	FN	Acc.
1-2	10	9	1	0	0.90
2-3	9	9	0	0	1.00
3-4	9	9	0	0	1.00
4-5	10	10	0	0	1.00
5-6	9	8	1	0	0.89
6-7	10	9	1	0	0.90
7-8	8	6	2	0	0.75
8-9	7	6	1	0	0.86
9-10	7	7	0	0	1.00
sum	79	73	6	0	0.924

This method performed better than the previous two. In most cases, a maximum of one error was committed in each image. Only on pair 7-8, two errors were found. These errors are illustrated in Figure 5.20. For example, the third region of the seventh image was not correctly matched with the third region of the next image, but with the second one. The problem is caused by occlusion, which increased the percentage of light pixels. However, it should be considered that correlation-based analysis may find high correlation values even when the compared variables occupy different ranges or even have different dispersion. The correlation points only to the linear dependence between the variables. For this reason, if the variation is similar, even comparing a light region with a dark area, the correlation can point to high correspondence.

FIGURE 5.20 - EXAMPLES OF THE ERROS IN THE CORRELATION ANALYSIS.



5.3.3 Ten and twenty frames without spatial constraint

A similar process was repeated using one image every 10 frames and one image every 20 frames. The results will be summarized here to reduce redundancy in the text. The summary of the results using one image every 10 and 20 frames and the ED is shown in Table 5.12.

TABLE 5.12 - STATISTICS OBTAINED USING ONE IMAGE EVERY TEN OR TWENY FRAMES WITH THE ED.

	Every 10					Every 20				
Pairs	tot	TP	FP	FN	Acc	Tot	TP	FP	FN	Acc.
1-3	9	9	0	0	1.00	9	6	2	1	0.67
3-5	9	8	1	0	0.89	7	3	3	1	0.43
5-7	9	4	5	0	0.44	8	4	4	0	0.50
7-9	7	5	2	0	0.71	10	7	3	0	0.70
9-11	8	6	2	0	0.75	9	5	4	0	0.56
11-13	10	8	2	0	0.80	10	6	3	1	0.60
13-15	10	10	0	0	1.00	10	6	2	2	0.60
15-17	9	7	2	0	0.78	11	6	5	0	0.55
17-19	10	9	1	0	0.90	12	6	3	3	0.50
Sum	81	66	15	0		86	49	29	8	

The tracking performance changed rapidly for the experiment using one image every ten frames and when using one image every 20 frames. The precision value decreases from 0.815 to 0.628 and the recall (i.e., sensitivity) decreases from 1.000 to 0.860. The F1-Score decreases from 0.898 to 0.726. The accuracy dropped 0.248 (i.e., from 0.815 to 0.567).

To perform a comparison of the methods, the statistics that describe the quality of the results are summarized in Table 5.13.

TABLE 5.13 - SUMMARY OF THE ACCURACY STATISTICS WITHOUT SPATIAL CONSTRAINT.

	ED			DTW			Correlation		
frames	5	10	20	5	10	20	5	10	20
Accuracy	0.924	0.815	0.57	0.949	0.765	0.57	0.962	0.79	0.535

Recall	0.948	1	0.86	0.987	1	1	0.987	1	1
Precision	0.973	0.815	0.628	0.962	0.765	0.57	0.974	0.79	0.535
F1-score	0.961	0.898	0.726	0.974	0.867	0.726	0.981	0.883	0.697

The accuracies decrease as the distance between frames increases for all methods. The DTW method suffers more this effect when using one image every 10 frames but is better when using a 20 frames interval.

Among the methods, the correlation-based method produced the best results when using one image every five frames. This is the best situation, considering the frame rate. The worst results are obtained with the Euclidean distance. From the experiments, it was concluded that the next experiments will be performed using one image every five frames because higher intervals cause a 10% accuracy loss.

5.3.4 Inclusion of spatial constraint

In the previous experiment, all possible matches were considered but some combinations are impossible as the spatial shifts are not large as a result of the pedestrian or camera movement. So, in the next experiment, the frame rate was maintained constant (one image every 5 frames) but a spatial distance constraint was included. The candidate region was searched within a 120 pixels radius region around the position on the first image. The radius was chosen as the mean value of the region width.

TABLE 5.14 - SUMMARY OF THE ACCURACY STATISTICS WITH SPATIAL CONSTRAINT.

	ED	DTW	Correlation
Accuracy	0.924	0.949	0.962
Recall	0.948	0.987	0.987
Precision	0.973	0.962	0.974
F1-score	0.960	0.974	0.981

According to the accuracy values, the best results are obtained using the correlation, but the DTW method produces similar results. The statistics also show that the quality can be increased by introducing a spatial constraint. Nevertheless, this decision needs to consider the pedestrian movement speed and the camera shift if installed on a moving platform. For a fixed camera, this option can contribute to obtaining better results.

5.3.4 Tracking with background suppression

Finally, the regions of interest were segmented using the proposed background suppression method and the spatial constraint. The resulting quality statistics are presented in Table 5.15.

TABLE 5.15 - SUMMARY OF THE ACCURACY STATISTICS WITH BACKGROUND SUPPRESSION.

	ED	DTW	Correlation
Accuracy	0.810	0.937	0.899
Recall	0.853	0.987	0.947
Precision	0.941	0.949	0.947
F1-score	0.895	0.967	0.947

The proposed method does not cut completely the background, nevertheless, significantly reduces its occurrence in the analyzed regions. As consequence, the histograms are modified, and the results of the match analysis changed. The accuracies obtained using the correlation and the ED are lower, below 90%, while the one measured with the DTW stood above 90% and is like the one measured without background suppression. This means that background suppression did not improve the detection of matches with ED and correlation. As the images are obtained in short intervals, small background changes were expected, which supports the use of the background.

On the other hand, when the background is suppressed, the DTW is still efficient, which means that it could perform better even if the interval between images is larger.

All three algorithms proved that they can be used in tracking a human from image frames acquired using mobile cameras in an urban environment. As the results support, tracking with background suppression had improved performance tracking. These results sustain the working hypothesis of this thesis that the background suppression would improve the tracking process rather than simple template matching. Surprisingly, the correlation was more robust than ED and DTW on sets of images without background suppression. Correlation can be a worthy option in cases where the background RoI containing a pedestrian has not been suppressed. Although the tracking experiment was only run on self-collected data, we believed that a similar tendency would occur for other datasets as well.

CONCLUSIONS

In this thesis a method to detect and monitor the movement of pedestrians from high spatial resolution images, obtained in an outdoor environment, using deep learning-based methods and image processing techniques framework was developed.

For human detection, two techniques namely Faster R-CNN Inception v2 and SSD Mobilenet v2 were tested and evaluated to draw a comparison between them on the basis of accuracy, precision, recall, and mAP. The fine-tuned model of Faster R-CNN Inception v2 was selected as an optimal model for the proposed application.

The Faster R-CNN Inception v2 model can identify pedestrians under challenging conditions with high confidence, precision, and recall whereas the SSD Mobilenetv2 model performs better in speed which is useful for real-time pedestrian detection. Even so, Faster R-CNN models might be just fast if the number of proposed regions is limited.

From the experiment, the fine-tuning of the Faster R-CNN Inception v2, and SSD Mobilenet v2 detectors generate a coherent increase in the values of accuracy, precision, recall, and F1-score when retrained in small data set (700 images, taking around 4000 training steps), this demonstrates the potential of transfer learning technique in practical applications.

Changing parameters/hyperparameters can affect the performance of an object detection model significantly and should be evaluated and adjusted according to the problem in hand;

Using transfer learning can be an essential step in order to achieve viable results in the short term. Since the problem being addressed uses input images with no similarity to the existent datasets, gathering only the weights of the pre-trained model can lead to better results than freezing some of the layers, with the repercussion of a slower training process;

In the second phase of the framework, a method to perform the separation of human bodies from images with changing background was proposed and tested. The proposed method is based on image segmentation, the analysis of the possible pose, and a final refinement step based on probabilistic relaxation. Quantitative and qualitative experiments on different challenging image datasets demonstrate the superiority of the proposed approaches over some state-of-the-art methods.

The method was also tested with high-resolution RGB images acquired through mobile cameras and the results prove that the method can be used to identify the segments related to the pedestrian with relatively good performance. The tests reveal a mean IoU of around 71% with some cases reaching IoU values close to 100%. Nevertheless, the experiments also show that the method can fail. It occurs when the shape of the pedestrian cannot be modeled by the used density fields or when there is low contrast between the pedestrian and the background. The situation regarding the shape can be improved by including more templates with different poses, but it must also be considered difficult to model all possible poses and errors will persist.

The problem of low contrast between clothes or skin of the pedestrian and the background, is more difficult to solve and causes difficulties even to a human interpreter. The low contrast that affects the segmentation and clustering steps cannot be improved in the following steps. The low contrast also can affect the refinement by probabilistic relaxation because it is based on the RGB color information.

The method performed well and it can be used to segment pedestrians from images obtained from moving cameras as it does not depend on a fixed known background and can be applied to images with different sizes, which makes it usable in the analysis of images with varying depth.

Tracking is achieved by creating a correspondence between consecutive image frames. The proposed tracking approach is characterized by a high person matching rate, meaning that very few false positives and false negatives are obtained by the tracking technique. Additionally, was investigated how the performance of the trackers is affected by the frame-rate with and without the spatial constraint of the input image frames and background suppression. In general, for evaluated metrics, the trackers performed well. It is also noted that the tracker's performance drops when the interval between image frames is larger. In general, using background suppression boosted the performance slightly compared to using image frames without background suppression. Among the three tested tracking algorithms, when the background is suppressed, the DTW perform better even if the interval between images is larger.

Future work

Since there are countless configurations for CNN architectures that could be employed and will likely remain an area of significant future research. Although CNN fine-tuning and parameter configuration is important, for the purpose of this study only two architectures were tested.

Future work should test these architectures (or others) with a different number of classes and use fine-tuning with model topology adjustment by varying some model parameters such as IoU, graph, batch size, moment, weight decay, and anchors boxes.

It is recommended to improve the first segmentation step, by including more information about the possible shape of the human figures or, look for other alternatives to solve this problem. Other alternative color spaces like CIELab or CIELuv not explored in this thesis can be tested.

Instead of matching features with DTW, ED, or correlation and future work should consider training a network to aid matching correspondence.

The proposed approach to track pedestrians was based on histogram matching. The method is very promising for practical application in an outdoor environment. In a future application, one can use other segmentation methods such as DeepLab to extract silhouettes and perform correspondence. Along with the color histogram, other features like position, size, area, the perimeter can be incorporated in the proposed matching algorithms method.

In this thesis tracking algorithms are performed on small and quite specific self-collected data, an obvious future work is to apply the tests to larger datasets such as those in the MOTchallenge, PETS, so on. Another possible extension would be to test other tracking algorithms (e.g., SORT and Deep SORT) apart from those proposed in the present thesis. Testing other tracking algorithms would give further insights into how the use of the color-histogram feature affects tracking performance in the more general case.

In the future, other color spaces like HSV, CIELab, or CIELuv may be experimented with to compute histogram features and see if further improvements could be achieved.

The results obtained by the DTW method are very encouraging since accuracies of over 93% were obtained. Perhaps, this is the first time DTW is used for human tracking. For future work, experiments using extended dynamic time warping such as weighted distance time warping or probability-based dynamic time warping can be implemented and tested.

REFERENCES

ACHANTA, R., SHAJI, A., SMITH, K., LUCCHI, A., FUA, P., SÜSSTRUNK, S. 2012. **SLIC superpixels compared to state-of-the-art superpixel methods**. IEEE transactions on pattern analysis and machine intelligence, 34(11), 2274-2282.

ANGELICO, J., WARDANI, K. R. R. 2018. **Convolutional neural network using kalman filter for human detection and tracking on RGB-D video**. CommIT (Communication and Information Technology) Journal, 12(2), 105-110.

ALON, J., ATHITSOS, V., YUAN, Q., SCLAROFF, S. 2009. **A unified framework for gesture recognition and spatiotemporal gesture segmentation**. IEEE transactions on pattern analysis and machine intelligence, 31(9), 1685-1699.

BAABOU, S., FRADJ, A. B., FARAH, M. A., ABUBAKR, A. G., BREMOND, F., KACHOURI, A. 2019. **A comparative study and state-of-the-art evaluation for pedestrian detection**. In 2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA) (pp. 485-490). IEEE.

BABACAN, S. D., PAPPAS, T. N. 2007. **Spatiotemporal algorithm for background subtraction**. In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07 (Vol. 1, pp. I-1065). IEEE.

BARNICH, O., VAN DROOGENBROECK, M. 2010. **ViBe: A universal background subtraction algorithm for video sequences**. IEEE Transactions on Image processing, 20(6), 1709-1724.

BENENSON, R., OMRAN, M., HOSANG, J., SCHIELE, B. 2014. **Ten years of pedestrian detection, what have we learned?** In European Conference on Computer Vision (pp. 613-627). Springer, Cham.

BELEZNAI, C., FRÜHSTÜCK, B., BISCHOF, H. 2006**. Human Tracking by Fast Mean Shift Mode Seeking**. Journal of Multimedia, 1(1), pp.1-8.

BERNARDIN, K., R. STIEFELHAGEN 2008. **Evaluating multiple objects tracking performance: The CLEAR MOT metrics**. EURASIP Journal on Image and Video Processing 2008(1), 246309.

BILODEAU, G. A., JODOIN, J. P., SAUNIER, N. 2013. **Change detection in feature space using local binary similarity patterns**. In 2013 International Conference on Computer and Robot Vision (pp. 106-112). IEEE.

BIBBY, C., REID, I. 2008. **Robust real-time visual tracking using pixel-wise posteriors**. In European Conference on Computer Vision (pp. 831-844). Springer, Berlin, Heidelberg.

BIRCHFIELD, S. 1998. **Elliptical head tracking using intensity gradients and color histograms**. In Proceedings. 1998 IEEE Computer Society conference on computer vision and pattern recognition (Cat. No. 98CB36231) (pp. 232-237). IEEE.

BOUREZAK, R., BILODEAU, G. A. 2006. **Iterative division and correlograms for detection and tracking of moving objects**. In International Workshop on Intelligent Computing in Pattern Analysis and Synthesis (pp. 46-55). Springer, Berlin, Heidelberg.

BOUWMANS, T., SOBRAL, A., JAVED, S., JUNG, S. K., ZAHZAH, E. H. 2017. **Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset**. Computer Science Review, 23, 1-71.

BOLYA, D., ZHOU, C., XIAO, F. LEE, Y.J., 2019. **Yolact++: Better real-time instance segmentation**. arXiv preprint arXiv:1912.06218.

BRAUN, M., KREBS, S., FLOHR, F., GAVRILA, D. M. 2019. **EuroCity persons: A novel benchmark for person detection in traffic scenes**. IEEE transactions on pattern analysis and machine intelligence, 41(8), 1844-1861.

BROCKWELL, A. E. 2005. **Recursive kernel density estimation of the likelihood for generalized state-space models**. CMU Statistics Dept. Tech. Report# 816.

CASTELLUCCIO, M., POGGI, G., SANSONE, C., VERDOLIVA, L. 2015. **Land use classification in remote sensing images by convolutional neural networks**. arXiv preprint arXiv:1508.00092.

CAPELLADES, M. B., DOERMANN, D., DEMENTHON, D., CHELLAPPA, R. 2003. **An appearance based approach for human and object tracking**. In Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429) (Vol. 2, pp. II-85). IEEE.

CANDÈS, E. J., LI, X., MA, Y., WRIGHT, J. 2011. **Robust principal component analysis**. Journal of the ACM (JACM), 58(3), 1-37.

COMANICIU, D., RAMESH, V., MEER, P. 2000. **Real-time tracking of non-rigid objects using mean shift**. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) (Vol. 2, pp. 142-149). IEEE.

CORDTS, M., OMRAN, M., RAMOS, S., SCHARWÄCHTER, T., ENZWEILER, M., BENENSON, R., ... SCHIELE, B. 2015. **The cityscapes dataset**. In CVPR Workshop on the Future of Datasets in Vision (Vol. 2).

CHANDRAJIT, M., GIRISHA, R., VASUDEV, T., 2016. **Multiple objects tracking in surveillance video using color and hu moments**. Signal Image Process. An Int. J. 7, 15–27. doi:10.5121/sipij.2016.7302.

CHAIBOU, M. S., CONZE, P. H., KALTI, K., MAHJOUB, M. A., SOLAIMAN, B. 2020. **Learning contextual superpixel similarity for consistent image segmentation** Multimedia Tools and Applications, 79(3), 2601-2627.

CHEN, L. C., PAPANDREOU, G., SCHROFF, F., ADAM, H. 2017. **Rethinking atrous convolution for semantic image segmentation**. arXiv preprint arXiv:1706.05587.

CHEN, L. C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., YUILLE, A. L. 2014. **Semantic image segmentation with deep convolutional nets and fully connected crfs**. arXiv preprint arXiv:1412.7062.

CHEN, L.C., PAPANDREOU, G., KOKKINOS, I., MURPHY, K. AND YUILLE, A.L., 2017. **Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs**. IEEE transactions on pattern analysis and machine intelligence, 40(4), pp.834-848.

CHEN, L.C., PAPANDREOU, G., SCHROFF, F., ADAM, H., 2017. **Rethinking atrous convolution for semantic image segmentation**. arXiv preprint arXiv:1706.05587.

CHEN, L. C., ZHU, Y., PAPANDREOU, G., SCHROFF, F., ADAM, H. 2018. **Encoder-decoder with atrous separable convolution for semantic image segmentation**. In Proceedings of the European conference on computer vision (ECCV) (pp. 801-818).

CHEN, M., WEI, X., YANG, Q., LI, Q., WANG, G., YANG, M. H. 2017. **Spatiotemporal GMM for background subtraction with superpixel hierarchy**. IEEE transactions on pattern analysis and machine intelligence, 40 (6), 1518-1525.

CHEN, Y., YANG, X., ZHONG, B., PAN, S., CHEN, D., ZHANG, H. 2016. **CNNTracker: Online discriminative object tracking via deep convolutional neural network**. Applied Soft Computing, 38, 1088-1098.

CHEN, X., WEI, P., KE, W., YE, Q., JIAO, J. 2014. **Pedestrian detection with deep convolutional neural network**. In Asian Conference on Computer Vision (pp. 354-365). Springer, Cham.

CIAPARRONE, G., SÁNCHEZ, F. L., TABIK, S., TROIANO, L., TAGLIAFERRI, R., HERRERA, F. 2020. **Deep learning in video multi-object tracking: A survey**. Neurocomputing, 381, 61-88.

CULIBRK, D., MARQUES, O., SOCEK, D., KALVA, H., FURHT, B. 2007. **Neural network approach to background modeling for video object segmentation**. IEEE Transactions on Neural Networks, 18(6), 1614-1627.

DAI, J., LI, Y., HE, K., SUN, J. 2016. **R-fcn: Object detection via region-based fully convolutional networks**. In Advances in neural information processing systems (pp. 379-387).

DOLLAR, P., WOJEK, C., SCHIELE, B., PERONA, P. 2011. **Pedestrian detection: An evaluation of the state of the art**. IEEE transactions on pattern analysis and machine intelligence, 34(4), 743-761.

DOLLAR, P., WOJEK, C., SCHIELE, B., PERONA, P. 2009. **Pedestrian detection: A benchmark**. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 304-311). IEEE.

DALAL, N., TRIGGS, B. 2005. **Histograms of oriented gradients for human detection**. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.

DOLLÁR, P., APPEL, R., BELONGIE, S., PERONA, P. 2014. **Fast feature pyramids for object detection**. IEEE transactions on pattern analysis and machine intelligence, 36(8), 1532-1545.

DALLEY, G., MIGDAL, J., GRIMSON, W. E. L. 2008. **Background subtraction for temporally irregular dynamic textures**. In 2008 IEEE Workshop on Applications of Computer Vision (pp. 1-7). IEEE.

ELGAMMAL, A., HARWOOD, D., DAVIS, L. 2000. **Non-parametric model for background subtraction**. In European conference on computer vision (pp. 751-767). Springer, Berlin, Heidelberg.

FAN, J., XU, W., WU, Y., GONG, Y. 2010. **Human tracking using convolutional neural networks**. IEEE Transactions on Neural Networks, 21(10), 1610-1623.

FELIPE, J. C., TRAINA, A. J., TRAINA, C. 2005. **Global warp metric distance: Boosting content-based image retrieval through histograms**. In Seventh IEEE International Symposium on Multimedia (ISM'05) (pp. 8-pp). IEEE.

FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D., RAMANAN, D. 2010. **Object detection with discriminatively trained part-based models**. IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645.

GOODFELLOW, I., BENGIO, Y., COURVILLE, A. 2016. **Deep learning**. MIT press: Cambridge, MA, USA.

GOUTTE, C., GAUSSIER, E. 2005. **A probabilistic interpretation of precision, recall and F-score, with implication for evaluation**. In European conference on information retrieval (pp. 345-359). Springer, Berlin, Heidelberg.

GIRSHICK, R., DONAHUE, J., DARRELL, T., MALIK, J. 2014. **Rich feature hierarchies for accurate object detection and semantic segmentation**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

GIRSHICK, R. 2015. **Fast R-CNN**. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

HARITAOGLU, I., HARWOOD, D., DAVIS, L. S. 2000. **W/sup 4: real-time surveillance of people and their activities**. IEEE Transactions on pattern analysis and machine intelligence, 22(8), 809-830.

HAN, B., COMANICIU, D., ZHU, Y., DAVIS, L. S. 2008. **Sequential kernel density approximation and its application to real-time visual tracking**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (7), 1186-1197.

HE, K., GKIOXARI, G., DOLLÁR, P. GIRSHICK, R., 2017. **Mask r-cnn**. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

HOFMANN, M., TIEFENBACHER, P., RIGOLL, G. 2012. **Background segmentation with feedback: The pixel-based adaptive segmenter**. In 2012 IEEE computer society conference on computer vision and pattern recognition workshops (pp. 38-43). IEEE.

HORN, B. K., SCHUNCK, B. G. 1981. **Determining optical flow**. In Techniques and Applications of Image Understanding (Vol. 281, pp. 319-331). International Society for Optics and Photonics.

HONG, S., YOU, T., KWAK, S., HAN, B. 2015. **Online tracking by learning discriminative saliency map with convolutional neural network**. In International conference on machine learning (pp. 597-606).

HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A., FATHI, A., FISCHER, I., WOJNA, Z., SONG, Y., GUADARRAMA, S. AND MURPHY, K., 2017. **Speed/accuracy trade-offs for modern convolutional object detectors**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7310-7311).

HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A., FATHI, A., ...MURPHY, K. 2016. **Speed/accuracy trade-offs for modern convolutional object detectors**. arXiv preprint arXiv:1611.10012.

JIMÉNEZ, P. G., MALDONADO-BASCÓN, S., GIL-PITA, R., GÓMEZ-MORENO, H. 2003. **Background pixel classification for motion detection in video image sequences**. In International Work-Conference on Artificial Neural Networks (pp. 718-725). Springer, Berlin, Heidelberg.

JIN, J., DUNDAR, A., BATES, J., FARABET, C., CULURCIELLO, E. 2013. **Tracking with deep neural networks**. In 2013 47th Annual Conference on Information Sciences and Systems (CISS) (pp. 1-5). IEEE.

JIN, L., CHENG, J., HUANG, H. 2010. **Human tracking in the complicated background by particle filter using color-histogram and hog**. In 2010 International Symposium on Intelligent Signal Processing and Communication Systems (pp. 1-4). IEEE.

KIM, H., SAKAMOTO, R., KITAHARA, I., TORIYAMA, T., KOGURE, K. 2007. **Robust foreground extraction technique using Gaussian family model and multiple thresholds**. In Asian Conference on Computer Vision (pp. 758-768). Springer, Berlin, Heidelberg.

KIM, K., CHALIDABHONGSE, T. H., HARWOOD, D., DAVIS, L. 2004. **Background modeling and subtraction by codebook construction**. In 2004 International Conference on Image Processing, 2004. ICIP'04. (Vol. 5, pp. 3061-3064). IEEE.

KEMKER, R., SALVAGGIO, C., KANAN, C. 2018. **Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning**. ISPRS journal of photogrammetry and remote sensing, 145, 60-77.

KESKAR, N. S., MUDIGERE, D., NOCEDAL, J., SMELYANSKIY, M., TANG, P. T. P. 2016. **On large-batch training for deep learning: Generalization gap and sharp minima**. arXiv preprint arXiv:1609.04836.

KUKAČKA, J., GOLKOV, V., CREMERS, D. 2017**. Regularization for deep learning: A taxonomy**. arXiv preprint arXiv:1710.10686.

LAN, W., DANG, J., WANG, Y., WANG, S. 2018. **Pedestrian detection based on yolo network model**. In 2018 IEEE international conference on mechatronics and automation (ICMA) (pp. 1547-1551). IEEE.

LECUN, Y. 1989. **Generalization and network design strategies**. Connectionism in perspective, 19, 143-155.

LECUN, Y., BENGIO, Y., HINTON, G. 2015. **Deep learning**. Nature, 521 (7553), 436-444.

LECUN, Y., BOTTOU, L., BENGIO, Y., HAFFNER, P. 1998. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 86(11), 2278-2324.

LEE, D. S. 2005. **Effective Gaussian mixture learning for video background subtraction**. IEEE transactions on pattern analysis and machine intelligence, 27 (5), 827-832.

LI, J., LIANG, X., SHEN, S., XU, T., FENG, J., YAN, S. 2015. **Scale-aware Fast R-CNN for pedestrian detection**. arXiv preprint arXiv:1510.08160.

LI, P., WANG, D., WANG, L. LU, H., 2018. **Deep visual tracking: Review and experimental comparison**. Pattern Recognition, 76, pp.323-338.

LI, Z., HOIEM, D. 2017. **Learning without forgetting**. IEEE transactions on pattern analysis and machine intelligence, 40(12), 2935-2947.

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C. Y., BERG, A. C. 2016. **Ssd: Single shot multibox detector**. In European conference on computer vision (pp. 21-37). Springer, Cham.

LU, W., TAN, Y. P. 2001. **A color histogram based people tracking system**. In ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196) (Vol. 2, pp. 137-140). IEEE.

LUO, Y., YIN, D., WANG, A., WU, W. 2018. **Pedestrian tracking in surveillance video based on modified CNN**. Multimedia Tools and Applications, 77(18), 24041-24058.

MCKEOWN, D. M., COCHRAN, S. D., FORD, S. J., MCGLONE, J. C., SHUFELT, J. A., YOCUM, D. A. 1999. **Fusion of HYDICE hyperspectral data with panchromatic imagery for cartographic feature extraction**. IEEE Transactions on Geoscience and Remote Sensing, 37(3), 1261-1277.

MCKENNA, S. J., JABRI, S., DURIC, Z., ROSENFELD, A., WECHSLER, H. 2000. **Tracking groups of people**. Computer vision and image understanding, 80(1), 42-56.

MIKOLAJCZYK, K., SCHMID, C., ZISSERMAN, A. 2004. **Human detection based on a probabilistic assembly of robust part detectors**. In European Conference on Computer Vision (pp. 69-82). Springer, Berlin, Heidelberg.

MINAEE, S., BOYKOV, Y., PORIKLI, F., PLAZA, A., KEHTARNAVAZ, N., TERZOPOULOS, D. 2020. **Image segmentation using deep learning: A survey**. arXiv preprint arXiv:2001.05566.

MIRABI, M., JAVADI, S. 2012. **People tracking in outdoor environment using Kalman filter**. In 2012 Third International Conference on Intelligent Systems Modelling and Simulation (pp. 303-307). IEEE.

MILAN, A., LEAL-TAIXÉ, L., SCHINDLER, K. REID, I., 2015**. Joint tracking and segmentation of multiple targets**. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5397-5406).

MITTAL, A., PARAGIOS, N. 2004. **Motion-based background subtraction using adaptive kernel density estimation**. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. (Vol. 2, pp. II-II). IEEE.

NAIR, V., HINTON, G. E. 2010. **Rectified linear units improve restricted boltzmann machines**. In ICML.

NAJIBI, M., RASTEGARI, M., DAVIS, L. S. 2016. **G-cnn: an iterative grid based object detector.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2369-2377).

NAM, W., DOLLÁR, P., HAN, J. H., 2014. **Local decorrelation for improved pedestrian detection**. In: 28th Annual Conference on Neural Information Processing Systems (NIPS).

NUMMIARO, K., KOLLER-MEIER, E., SVOBODA, T., ROTH, D., VAN GOOL, L. 2003. **Color-based object tracking in multi-camera environments**. In Joint Pattern Recognition Symposium (pp. 591-599). Springer, Berlin, Heidelberg.

OUYANG, W., WANG, X. 2013. **Joint deep learning for pedestrian detection**. In Proceedings of the IEEE international conference on computer vision (pp. 2056-2063).

PAN, S. J., YANG, Q. 2009. **A survey on transfer learning**. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

PAN, Z., LIU, S., FU, W. 2017. **A review of visual moving target tracking**. Multimedia Tools and Applications, 76(16), 16989-17018.

PICCARDI, M. 2004. **Background subtraction techniques: a review**. In 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583) (Vol. 4, pp. 3099-3104). IEEE.

POLYAK, B.T., 1964. **Some methods of speeding up the convergence of iteration methods**. Ussr computational mathematics and mathematical physics, 4(5), pp.1-17.

QU, Z., YU, S., FU, M. 2016. **Motion background modeling based on context-encoder**. In 2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR) (pp. 1-5). IEEE.

RASMUSSEN, C., TOYAMA, K., HAGER, G. D. 1996. **Tracking objects by color alone**. DCS RR-1114, Yale University.

RATH, T. M., MANMATHA, R. 2003. **Word image matching using dynamic time warping**. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. (Vol. 2, pp. II-II). IEEE.

REDMON, J., DIVVALA, S., GIRSHICK, R., FARHADI, A. 2016. **You only look once: Unified, real-time object detection**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

REN, S., HE, K., GIRSHICK, R., SUN, J. 2017. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137.

REN, S., HE, K., GIRSHICK, R., SUN, J. 2015. **Faster R-CNN: Towards real-time object detection with region proposal networks**. In Advances in neural information processing systems (pp. 91-99).

ROHR, K. 1994. **Towards model-based recognition of human movements in image sequences**. CVGIP: Image understanding, 59(1), 94-115.

RUAN, Y., WEI, Z. 2016. **Discriminative descriptors for object tracking**. Journal of Visual Communication and Image Representation, 35, 146-154.

RUDER, S., 2016. **An overview of gradient descent optimization algorithms**. arXiv preprint arXiv:1609.04747.

SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A., CHEN, L. C. 2018. **Mobilenetv2: Inverted residuals and linear bottlenecks.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

SAYKOL, E., GÜDÜKBAY, U., ULUSOY, Ö. 2002. **A histogram-based approach for object-based query-by-shape-and-color in multimedia databases**. Available as a technical report (BU-CE-0201) at: http://www.cs.bilkent.edu.tr/tech-reports/2002/BU-CE-0201.ps.gz.

SCHOFIELD, A. J., MEHTA, P. A., STONHAM, T. J. 1996. **A system for counting people in video images using neural networks to identify the background scene**. Pattern Recognition, 29(8), 1421-1428.

SCHICK A., BÄUML M., STIEFELHAGEN R. 2012. **Improving foreground segmentations with probabilistic superpixel Markov random fields**. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW'12, IEEE, pp 27–31.

SERMANET, P., EIGEN D., ZHANG, X., MATHIEU, M., FERGUS, R., LECUN, Y. 2013. **Overfeat: Integrated recognition, localization and detection using convolutional networks**. arXiv preprint arXiv:1312.6229.

SMITH, S. L., KINDERMANS, P. J., YING, C., LE, Q. V. 2017. **Don't decay the learning rate, increase the batch size**. arXiv preprint arXiv:1711.00489.

SONG, H., CHOI, I. K., KO, M. S., BAE, J., KWAK, S., YOO, J. 2018. **Vulnerable pedestrian detection and tracking using deep learning**. In 2018 International Conference on Electronics, Information, and Communication (ICEIC) (pp. 1-2). IEEE.

SHAFIEE, M. J., SIVA, P., FIEGUTH, P., WONG, A. 2016. **Embedded motion detection via neural response mixture background modeling**. In 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (pp. 837-844). IEEE.

SHIMADA, A., ARITA, D., TANIGUCHI, R. I. 2006. **Dynamic control of adaptive mixture-of-gaussians background model**. In 2006 IEEE International Conference on Video and Signal Based Surveillance (pp. 5-5). IEEE.

STAUFFER, C., GRIMSON, W. E. L. 1999. **Adaptive background mixture models for real-time tracking. In Proceedings**. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149) (Vol. 2, pp. 246-252). IEEE.

SUN, Z., CHEN, J., LIANG, C., RUAN, W. MUKHERJEE, M., 2020. **A survey of multiple pedestrian tracking based on tracking-by-detection framework**. IEEE Transactions on Circuits and Systems for Video Technology.

SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J., WOJNA, Z. 2016. **Rethinking the inception architecture for computer vision**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D. ... RABINOVICH, A. 2015. **Going deeper with convolutions**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

TSANG, S. H. 2019. **Review: Xception-With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)**. Towards Data Science. Last accessed 12 July 2020.

VIOLA, P., JONES, M. J., SNOW, D. 2003. **Detecting pedestrians using patterns of motion and appearance**. In: Computer Vision, 2003. Proceedings. The 9th ICCV, Nice, France, pp. 734.741 vol.1.

VINTSYUK, T.K. 1968. **Speech discrimination by dynamic programming**. In: Cybernetics 4(1), pp. 52–57.

WANG, L., SHI, J., SONG, G., SHEN, I. F. 2007. **Object detection combining recognition and segmentation**. In Asian conference on computer vision (pp. 189-199). Springer, Berlin, Heidelberg.

WOJEK, C., WALK, S., SCHIELE, B. 2009. **Multi-cue onboard pedestrian detection**. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 794-801). IEEE.

WREN, C. R., AZARBAYEJANI, A., DARRELL, T., PENTLAND, A. P. 1997. **Pfinder: Real-time tracking of the human body**. IEEE Transactions on pattern analysis and machine intelligence, 19(7), 780-785.

WU, B., NEVATIA, R. 2005. **Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors**. In Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 (Vol. 1, pp. 90-97). IEEE.

XIA, M., LI, T., XU, L., LIU, L., DE SILVA, C. W. 2017. **Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks**. IEEE/ASME Transactions on Mechatronics, 23(1), 101-110.

XU, P., YE, M., LIU, Q., LI, X., PEI, L., DING, J. 2014. **Motion detection via a couple of auto-encoder networks**. In 2014 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). IEEE.

XU, J. 2017. **Deep learning for object detection: A comprehensive review.** Retrieved December 10, 2018.

YAO, Y., ROSASCO, L., CAPONNETTO, A. 2007. **On early stopping in gradient descent learning**. Constructive Approximation, 26(2), 289-315.

YANG, H., SHAO, L., ZHENG, F., WANG, L., SONG, Z. 2011. **Recent advances and trends in visual tracking: A review**. Neurocomputing, 74(18), 3823-3831.

YOSINSKI, J., CLUNE, J., BENGIO, Y., LIPSON, H. 2014. **How transferable are features in deep neural networks?** In Advances in neural information processing systems (pp. 3320-3328).

YILMAZ, A., JAVED, O., SHAH, M. 2006. **Object tracking: A survey**. Acm computing surveys (CSUR), 38(4), 13-es.

YU, W., HOU, Z., WANG, P., QIN, X., WANG, L., LI, H. 2018. **Weakly supervised foreground segmentation based on superpixel grouping**. IEEE Access, 6, 12269-12279.

ZENG, Z., MA, S. 2002. **Head tracking by active particle filtering**. In Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition (pp. 89-94). IEEE.

ZIVKOVIC, Z. 2004. **Improved adaptive gaussian mixture model for background subtraction**. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (Vol. 2, pp. 28-31). IEEE.

ZHAI, S., DONG, S., SHANG, D., WANG, S. 2020. **An improved Faster R-CNN pedestrian detection algorithm based on feature fusion and context analysis**. IEEE Access.

ZHAI, S., SHANG, D., WANG, S., DONG, S. 2020. **DF-SSD: An improved SSD object detection algorithm based on densenet and feature fusion**. IEEE Access, 8, 24344-24357.

ZHAO, T., NEVATIA, R., LV, F. 2001. **Segmentation and tracking of multiple humans in complex situations**. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 (Vol. 2, pp. II-II). IEEE.

ZHAO, Z. Q., ZHENG, P., XU, S. T., WU, X. 2019. **Object detection with deep learning: A review**. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.

ZHANG, Y., LI, X., ZHANG, Z., WU, F., ZHAO, L. 2015. **Deep learning driven blockwise moving object detection with binary scene modeling**. Neurocomputing, 168, 454-463.

ZHANG, L., LIN, L., LIANG, X., HE, K. 2016. **Is Faster R-CNN doing well for pedestrian detection?** In European conference on computer vision (pp. 443-457). Springer, Cham.

ZHANG, L., ZHANG, L., DU, B. 2016. **Deep learning for remote sensing data: A technical tutorial on the state of the art**. IEEE Geoscience and Remote Sensing Magazine, 4(2), 22-40.

ZHANG, S., BENENSON, R., SCHIELE, B. 2017. **Citypersons: A diverse dataset for pedestrian detection**. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3213-3221).

ZHOU, H., FEI, M., SADKA, A., ZHANG, Y., LI, X. 2014. **Adaptive fusion of particle filtering and spatio-temporal motion energy for human tracking**. Pattern Recognition, 47(11), 3552-3567.

ZHU, X. X., TUIA, D., MOU, L., XIA, G. S., ZHANG, L., XU, F., FRAUNDORFER, F. 2017. **Deep learning in remote sensing: A comprehensive review and list of resources**. IEEE Geoscience and Remote Sensing Magazine, 5(4), 8-36.