UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MAURÍCIO DA COSTA JUSTO IZÉ

Multiple Pedestrian Tracking using Geometric and Deep Features

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Cláudio Rosito Jung

Porto Alegre
December 2019

“You can’t connect the dots looking forward;
you can only connect them looking backwards.
So you have to trust that the dots will somehow
connect in your future.”

— Steve Jobs

AGRADECIMENTOS

A minha família, principalmente minha mãe, pelo apoio incondicional e por sempre incentivar minha educação. À Jessica pelo apoio e incentivo nos mais diversos momentos. Aos meus amigos, principalmente ao William e ao André, por todas as conversas compartilhadas. Ao meu orientador pelo suporte dado na realização deste trabalho. Às professoras Eliane Pozzebon e Luciana Bolan Frigo pela confiança em mim nos primeiros anos da universidade e incentivo ao intercâmbio. E a todos que direta ou indiretamente fizeram parte da minha formação.

ABSTRACT

Automated video analysis is a growing demand in the context where technology enables the creation of visual data at increasing rates. Detection and tracking of multiple pedestrians are specific needs of this demand as they are used in different types of applications such as video surveillance, automated vehicles, and retail analysis. This work implements a multiple pedestrian tracking application using the tracking-by-detection paradigm. We use a pre-trained model for object detection and pedestrian reidentification, a linear state prediction through recursive filtering, geometric and deep features as pedestrian dissimilarity, and an optimal algorithm for the assignment problem due to the probable presence of multiple pedestrians on the video frame. We tested four methods locally using the MOTChallenge dataset; when including static and moving camera sequences, the method using deep features and pedestrian reidentification achieves an overall 48.3% MOTA. Analyzing only the PETS09-S2L1 sequence, a static sequence that is closer to what is found in surveillance cameras, the method achieves 79.5% MOTA. Furthermore, we present the details of the implementation and results of the four developed methods.

Keywords: Pedestrian tracking. deep features. geometric features. filtering.

Rastreamento de Múltiplos Pedestres usando Características Geométricas e Profundas

RESUMO

A análise automatizada de vídeo é uma demanda crescente em diferentes áreas no contexto em que a tecnologia permite a criação de dados visuais em taxas crescentes. A detecção e o rastreamento de múltiplos pedestres são necessidades específicas dessa demanda, pois são usados em aplicativos como vigilância por vídeo, veículos automatizados, e geração de métricas para lojas. Este trabalho implementa um aplicativo de rastreamento de múltiplos pedestres usando a abordagem de rastreamento por detecção. Nós usamos um modelo pré-treinado para detecção de objetos e reidentificaçao de pedestres, predição linear de estado com filtragem recursiva, características geométricas e profundas como funções de dissimilaridade entre pedestres, e um algoritmo óptimo para o problema de associação devido à presença de vários pedestres num único quadro de vídeo. Nós testamos quatro métodos localmente usando o conjunto de dados do MOTChallenge. Ao incluir vídeos com câmeras estáticas e em movimento, o método usando características profundas com renascimento atinge 48,3% na métrica MOTA. Analisando apenas a sequência PETS09-S2L1, uma sequência estática mais próxima do que é encontrado nas câmeras de vigilância, o método alcança 79,5% na métrica MOTA. Além disso, apresentamos os detalhes da implementação e os resultados dos quatro métodos desenvolvidos.

Palavras-chave: Rastreamento de pedestres. características profundas. características geométricas. filtragem.

LIST OF ABBREVIATIONS AND ACRONYMS

MOTA	Multiple Pedestrian Tracking Accuracy
MOTP	Multiple Pedestrian Tracking Precision
SOT	Single Object Tracking
MOT	Multiple Object Tracking
NMS	Non-maximum Suppression
MTT	Multiple Target Tracking
MPT	Multiple Pedestrian Tracking
CNN	Convolutional Neural Network
SVM	Support Vector Machine
GPU	Graphics Processing Unit
KF	Kalman Filter
AP	Assignment Problem

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Image and video analysis have been receiving increasing interest since the technology has been leading us toward a world in which it is incredibly affordable to have the equipment to capture and store visual data. In this context, the multiple object tracking (MOT) is an important problem in computer vision community that is receiving attention in recent years due to both academic and commercial potential. It is a fundamental skill for several types of applications, such as video surveillance, human-computer interaction, and automated vehicles (FAN et al., 2016).

The problem is also one of the most challenging areas of research in computer vision (GUAN; XUE; ZHIYONG, 2016; LI et al., 2013). It has a long history spanning about 50 years (MALLICK et al., 2013) with around 30 multiple object tracking approaches being published in major conferences every year (KUMAR, 2014). Over the last years, deep learning models are boosting results in areas such as object detection and reidentification, allowing researchers to use such technologies to study novel approaches for the problem.

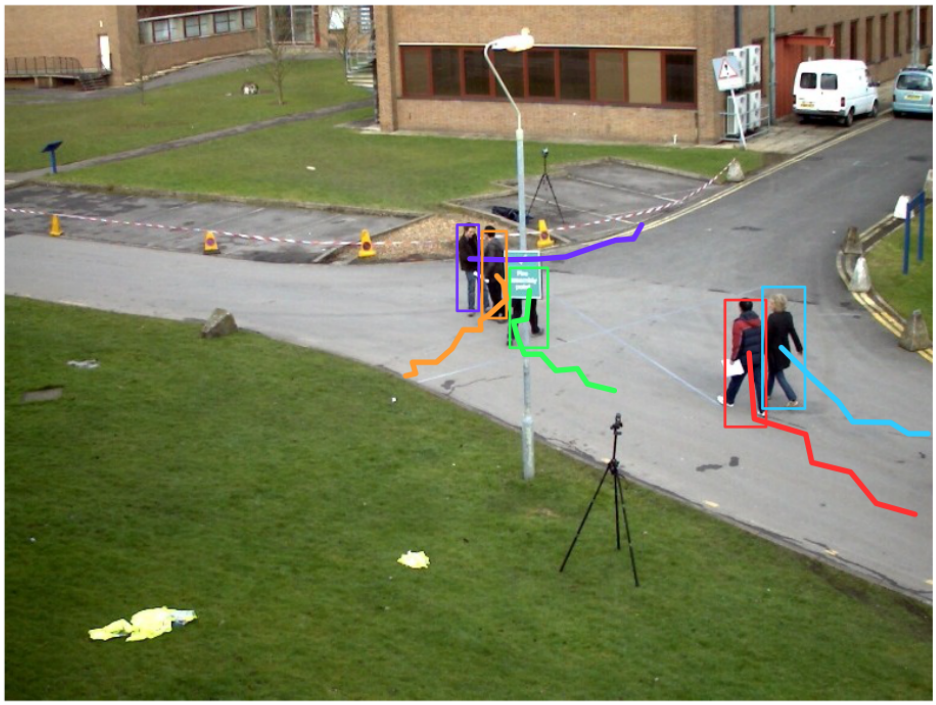
1.1 Problem definition

The MOT problem can be defined as the task of tracking multiple targets from a video stream. More formally, given a sequence of frames, the tracker needs to detect each target and propagate a unique identification for it through all the sequence. It also can be view as an extension of the object detection problem since the tracker needs to detect targets and also keep a connection between the same objects through the whole video sequence, even if the targets are moving or changing appearance through the scene.

This work uses pedestrians as object targets since they are the default class used in benchmarks of this subject, such as the MOTChallenge (LEAL-TAIXé et al., 2015). In this context, for each frame, the application needs to detect each *pedestrian* to check if it was seen before. If true, the detection must receive the same identification number as previously given; otherwise, a new unique identification number must be assigned to it. This is repeated for the whole stream of frames. In the end, the application outputs a list containing all pedestrians detected in the video with their unique identification number.

If we assign a color for each identification number, it is possible to plot the track lines produced by the tracker over the last frame of the video. The figure 1.1 shows an

Figure 1.1: An illustration of the track lines plotted over the frame



Source: Adapted from PETS09 dataset

illustration of the track lines produced by tracker that solves the MOT problem.

It is still an open problem in the literature since it presents several challenges even for state-of-art solutions, which still fall under complicated scenarios where there are many sources of uncertainty. The tracker system requires a combination of challenging computer vision tasks such as object detection, dissimilarity modeling, filtering techniques, and finally, pedestrian tracking (ANDRILUKA; ROTH; SCHIELE, 2008). Moreover, it is also a heavy computational problem that uses a large amount of data per frame, which makes it even more challenging for real-time applications.

Examples of complex scenarios that make tracking challenging are random target movements, noisy image sources, partial or total occlusion of objects. Scenarios with crowded pedestrians increase the number of false negative detections, which increases the difficulty for the application to track pedestrians with accuracy.

1.2 Work proposal

This work proposes the implementation of a multiple pedestrian tracking algorithm from a single uncalibrated camera using the tracking-by-detection paradigm where

two uncoupled components, the detector and the tracker, will be used to achieve the goal.

The detector is responsible for detecting all the pedestrians from a single frame, which is used to feed the tracker component that will assign a unique identification number for each detected pedestrian and propagate them through the whole video. In other words, as used in the literature, the tracker will do an online frame-by-frame assignment of detections without any information about future frames.

The use of some pre-trained models for object detection will be explored to detect pedestrians on each frame. For the tracker, the use of state prediction, geometric features, and a model trained for pedestrian reidentification will be studied to compute the dissimilarity between pedestrians and to explore methods to handle the reidentification of pedestrians.

The application should be able to online track one or more pedestrian from a single stream of video fed by a static monocular camera without any scenario information, background modeling, neither camera calibration. This work is not intended to deal with scenarios containing high density crowds. All the coordinates will be in bi-dimensional image coordinates where a point will be represented by (x, y) with the origin at the top-left corner of the image. The output consists of a text file with comma-separated values, which will contain the information of each detection - the frame number where it was detected, the boundary box position and size, and the assigned identification number unique for each pedestrian present in the whole sequence.

We want to check the overall results using four methodologies. The first method will use a state prediction combined with the Jaccard index as the dissimilarity function. The second will use the same technique, but followed by a pedestrian reidentification algorithm. The third method will use the cosine distance between deep feature vectors extracted from the pixels of the detected pedestrian as dissimilarity. Finally, the last method will use the cosine technique, but also followed by a pedestrian reidentification algorithm. All results will be compared using a dataset and an evaluation metrics commonly used in the literature for the MOT problem.

The following items describe the type of scenario expected by the implementation to handle the multiple pedestrian tracking.

- No scene information or ground plane calibration
- Single, static or moving, monocular camera videos
- Frame-by-frame online tracking

The techniques used to address the problem with the previous assumptions can be summarized by the following items.

- Tracking-by-detection paradigm
- Use of an optimal full-state estimator
- Use of an optimal linear assignment algorithm
- Use of a deeply learned model for object detection
- Use of a deeply learned pedestrian reidentification model

2 BACKGROUND & RELATED WORK

This chapter introduces the theoretical background used in the proposed methods of this monograph. Section 2.1 presents more formally the MOT problem and the tracking-by-detection paradigm in which this work is based. Section 2.2 presents about the detector component and techniques used to perform pedestrian detection. Finally, section 2.3 addresses the tracker component followed by techniques such as state prediction, optimal assignment, and pedestrian reidentification. Some related works will be presented with the exposed concepts, and any supplementary information can be found in the references.

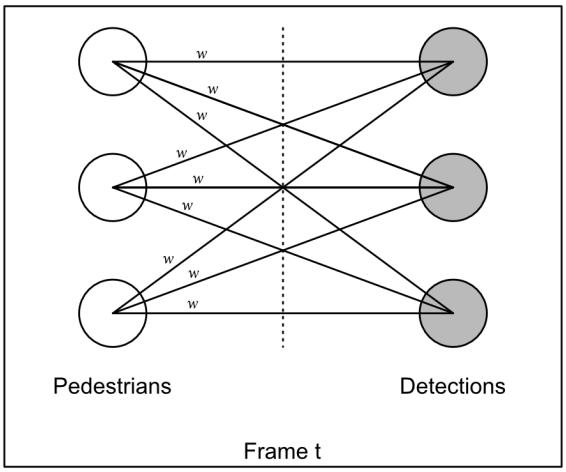
2.1 Multiple Object Tracking

The task of tracking multiple objects has the goal of finding trajectories of one or more targets in the scene from a video stream. It poses additional challenges when compared to single object tracking - where only the same and unique target is present in each frame - because more than one candidate is present to be associated, which produces a many-to-many relationship that needs to be solved by the tracker.

The problem can be view as a unidirectional weighed bi-partited graph where each frame contains M detections that need to be associated with N pedestrians, as illustrated in figure 2.1. Each possible association has a cost represented by the weight of the graph edge that models the dissimilarity between the two objects; thus, two different pedestrians should produce a high cost of association and vice versa. In each frame, the tracker needs to associate each detection with a pedestrian in order to minimize the global cost.

The subject can be grouped into two broad tracker categories: online or offline (KUMAR, 2014). An online tracker outputs a complete track of all objects of a sequence of frames from the beginning up to the current frame (BAE; YOON, 2014). In other words, it has no previous information about future frames, only data from the first frames up to the current frame being analyzed. The identification assignment is made on a frame-by-frame basis as the detections are being fed into the tracker. They often use some state estimator for prediction information about the next frame since there is no information about what is coming next. On the other hand, an offline tracker, also called a global or batch-based tracker, has access to all the frames before performing the assignments. Such trackers have more information available before making any decision, which addresses

Figure 2.1: An illustration of the problem as a bi-partited graph



Source: The author

them fewer challenges when compared to online trackers. This work will focus only on online trackers.

A common approach for an online tracker is the tracking-by-detection paradigm (CHU et al., 2019; WANG et al., 2019; ANDRILUKA; ROTH; SCHIELE, 2008). This methodology separates the system into two separated components: the detector and the tracker. This requires the continuous use of a detection algorithm in each frame to produce a list of detections to feed the next component that will associate a pedestrian numerical identification. These approaches have become increasingly popular, driven by the recent progress in object detection and also by being robust to handle typical challenges of the MOT problem such as changing background or occlusions of targets (Mekonnen; Lerasle, 2019; ANDRILUKA; ROTH; SCHIELE, 2008).

The *tracking without bells and whistles* (BERGMANN; MEINHARDT; LEAL-TAIXE, 2019) presents an approach based on bounding box regression where given a list of detections in a sequence of frames, the tracker uses a regressor which allows predicting of bounding box positions in future frames. However, the survey (LEON; GAVRILESCU, 2019) states that this approach presents some limitations since it requires that targets move slightly from frame to frame and also high frame rates to keep relatively stable.

Gaddigoudar et al. (2017) use particle-filter to solve non-linear problems such as state estimation in dynamic systems. Some of the tests included tracking pedestrians in crowded regions, which can have high occlusion, and the results presented a better performance when compared with the configuration using Kalman filter (KF), which is a linear state estimator. They argued that KF often terminates pedestrian tracks when

they get occluded in the scene. The solution was able to track fast-moving human targets effectively.

Bewley et al. (2016) proposed an online tracker for real-time applications using a similar approach to the first method of this work mentioned in section 1.2. They use the Kalman filter for state prediction and the Hungarian algorithm to find the globally optimal solution for the association. The approach achieved an accuracy comparable to state-of-the-art online trackers while being 20x faster.

The recent work of Chen et al. (2019) proposed a solution for the MOT problem using a novel spatial-temporal attentions and graph decomposition to assign pedestrians through the frames. The authors achieved a better performance than state-of-the-art works on the MOT16 and MOT17 dataset.

Bae and Yoon (2014) proposes a robust online multiple object tracking that aims to handle tracking even in occluded scenarios. The authors also propose a novel method for discriminating the appearances of objects using an incremental linear discriminant analysis, which led to keeping the correct tracking even in severe occlusions.

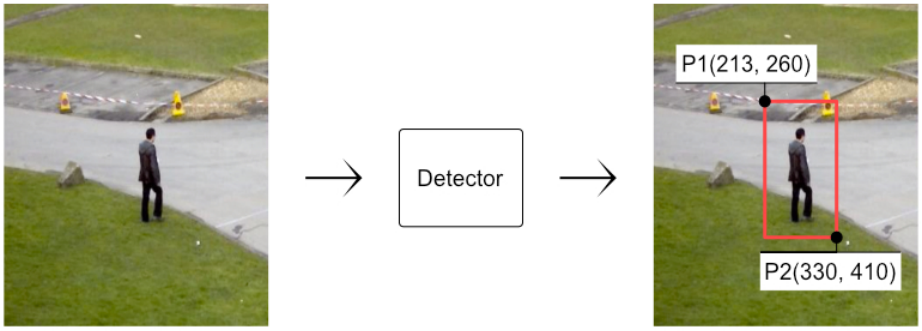
Breitenstein et al. (2011) uses particle filters to estimate the distribution of each target state and a greedy scheme for data association using cost values from classifiers trained per target in runtime. They were able to perform multiple object tracking from a single uncalibrated camera.

2.2 Detection

The goal of an object detector is to analysis a single image to localize and classify all the objects of interest. It outputs a list of bounding boxes containing the object localization and the confidence score about how sure it is about that detection being the targeted class label. The bounding box is a structure containing the position and size of the detection in the image coordinate system, often represented by the top-left and the bottom-right corners the box delimiting the detected object. Figure 2.2 illustrates this idea.

An image can be view as an array of pixels in which the detector needs to find the group of the pixels that represent the object. A basic technique for this task is to use a sliding window across the entire frame while searching for geometric features such as color, contour, or gradient until a threshold value is hit to a detection be made. More advanced techniques can include the use of integral channel features, optical flow, or

Figure 2.2: An illustration of a pedestrian detector input and output



Source: Adapted from PETS2009

background subtraction.

Lei and Huang (2018) use background subtraction to detect moving pedestrians by extracting background information using an adaptive background and extraction method combined with mathematical morphology operation and histogram segmentation. On the other hand, Gaddigoudar et al. (2017) propose a pedestrian tracking using particle filters and uses Bhattacharya distance between normalized histograms of two consecutive frame sequences to detect a pedestrian.

2.2.1 Deep Features

In the last years, machine learning techniques are boosting the results for object detection. Techniques that use convolutional neural network (CNN) are the dominant approach for almost all recognition and detection tasks. The good results of such techniques in pattern recognition tasks can be in part explained because they are kind invariant from position, orientation, or illumination of the targeted object.

They make use of deep features that are learned by the network itself in the training phase instead of being engineered by hand, allowing them to take advantage of computational power and data available (LECUN; BENGIO; HINTON, 2015). In other words, the model is structured in layers, then it is trained using a massive amount of data. This strategy forces the model to learn deep features by adjusting the weights of each layer until it can detect complex objects such as pedestrians.

The CNN structure is formed by different types of layers such as convolutional, pooling, or fully-connected layer and then connected from thousands to millions of weights. The data from one layer to another is a weighted sum of the inputs that go through a non-

linear activation function such as $ReLU$ (Rectified Linear Unit), $tanh$, or $sigmoid$. The stack of such layers can transform the input to increase both selectivity and invariance of the representation, allowing them to detect objects even if they present high variance of illumination, orientation, or position on the image.

Nowadays, popular well-performing CNN architectures used for object detection are Fast R-CNN (GIRSHICK, 2015), YOLO v3 (REDMON; FARHADI, 2018), and SSD (LIU et al., 2016). One of the first good results using CNN for object detection was introduced by OverFeat (SERMANET et al., 2013). Shortly after the publication, a method using regions was proposed in which around 2000 potential regions of interest were defined to apply an image classifier in each of them (GIRSHICK et al., 2013). Later, Fast-RCNN (GIRSHICK, 2015) and Faster-RCNN (REN et al., 2015) presented an optimization by using a neural network to propose objects of interest.

Novel approaches start to combine the use of regions of proposals and the classifier into one single network. This formed the basis for a single-shot object detector where the entire image is passed only once instead of a sliding window or selective region of interest, allowing much better results in speed. Examples of these types of detectors are the single-shot multibox detector (SSD), the you-only-look-once (YOLO), and the region-based fully convolutional network (R-FCN).

Redmon et al. (2015) proposed the YOLO model, which uses a single CNN to locate and classify the objects in one evaluation instead of the previous pipeline, which performed the classification on many regions of interest. It forwards the whole image through the neural network only once outputting the class probabilities and bounding boxes of each object detection. This approach presented a better speed when compared with other methods while keeping good accuracy. It uses a neural network with convolutional and pooling layers plus two fully connected layers at the end. The model splits the input image in a grid of $S \times S$ cells where each cell will be responsible for detecting one object which geometric center falls on it. The output of the system is a list of bounding-box and a confidence score. Each bounding box contains four variables that localize the object into the image by giving the position (x, y) and the size (w, h) of the detected object, also the confidence class score of the object inside the box which represent the probability of the detected object by a specific class label.

2.3 Tracking

The tracker component in a *tracking-by-detection* approach is responsible for assigning an identification per-frame basis for each detected object received from the previous detector component. To achieve this goal, it often makes use of techniques for state prediction, dissimilarity modeling, data association, and pedestrian reidentification.

2.3.1 Prediction

The use of a state predictor is valuable for an online tracker because they do not have any information about the next frame, so they need to make an estimation of variables of interest to increase the chances of making a correct association. In addition, filtering is also valuable for recursively generating optimal values, given a system where measurement and estimation uncertainties exist. The literature presents different ways to accomplish this, such as Kalman Filter (KF), Extended Kalman Filter, Particle Filter, and even recurrent neural networks. This work will focus on the use of KF as a filtering algorithm.

The KF is a linear optimal recursive state estimator that works at discrete time steps. It predicts and corrects initially defined state variables using linear processes to handle measurements containing statistical noise and estimation of uncertain variables by estimating a joint probability distribution over the variables for each time frame. In other words, it uses information from predicted states and noisy measurements that are combined to produce an ideal estimation of system variables in which the truthful values are unknown.

The KF has long been regarded as the optimal solution to many tracking and data prediction tasks. Although the pedestrian movement is not linear, the linear estimation realized by the filter can still be useful where the time difference between two consecutive frames in small enough to make the target movement looks linear. However, for low frame-rate cameras, a non-linear model movement should be considered, such as the Extended Kalman Filter or the Particle Filter.

The KF estimates the state vector $\vec{x} \in \Re^n$ of a system assuming the linear stochastic model presented in equation (2.1). The measurement $\vec{z} \in \Re^{\mathrm{n}}$ at time k is a linear combination between the state vector and the measurement noise v given in equation

(2.2).

$$\vec{x}_k = \mathbf{A}\vec{x}_{k-1} + \mathbf{B}\vec{u}_{k-1} + w_{k-1} \tag{2.1}$$

$$\vec{z}_k = \mathbf{H}_k\vec{x}_k + v_k \tag{2.2}$$

Matrix $\mathbf{A}$ represents the transition model, which modifies the state from the previous frame $\vec{x}_{k-1}$ to the state of the current frame $\vec{x}_k$; matrix $\mathbf{B}$ relates the control input model applied to the control vector $\vec{u}_k$; matrix $\mathbf{H}$ represents the observation model, which relates the state vector values with the measurement vector space. The variables w is the prediction noise and v is the measurement noise, both modeled as Gaussian distributions with zero mean, where $\mathbf{Q}$ is the process noise covariance and $\mathbf{R}$ is the measurement noise covariance, as shown by equations (2.3) and (2.4).

$$p(w) \sim N(0, \mathbf{Q}) \tag{2.3}$$

$$p(v) \sim N(0, \mathbf{R}) \tag{2.4}$$

The first practical step is to define the state variables to be estimated over time. For a pedestrian tracker that will receive the output directly from the object detector, the state of each pedestrian can be defined as containing at least four variables: x, y, v_x and v_y, which represent the horizontal and vertical position, and the horizontal and vertical velocity, respectively.

The prediction of the state $\hat{\mathbf{x}}_k$ from $\vec{x}_{k-1}$ and the covariance $\hat{\mathbf{P}}_k$ from $\mathbf{P}_{k-1}$ a priori are given by:

$$\vec{x}_k = \mathbf{A}\vec{x}_{k-1} + \mathbf{B}\vec{u}_{k-1} \tag{2.5}$$

$$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \tag{2.6}$$

Where matrix $\mathbf{A}$ and $\mathbf{B}$ are from equation (2.1) and $\mathbf{Q}$ from equation (2.3).

After a measurement is made, the posteriori variables estimates are given by:

$$K_k = \mathbf{P}_k\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}_k\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1} \tag{2.7}$$

$$\vec{x}_k = \vec{x}_k + K_k(\vec{z}_k - \mathbf{H}\vec{x}_k) \tag{2.8}$$

$$\mathbf{P}_k = (\mathbf{I} - K_k\mathbf{H})\mathbf{P}_k \tag{2.9}$$

Where equation (2.7) calculates the Kalman gain K. The equation (2.8) corrects the a priori predicted state. And the equation (2.9) updates the a priori state error covariance. The $\mathbf{I}$ is the identity matrix.

After each prediction and correction steps, the process is repeated to estimate the optimal state recursively. More deep information about the filter can be found at (SORENSON, 1970; MAYBECK et al., 1979; WELCH; BISHOP, 2006).

2.3.2 Data Association

The multiple object tracking has the problem in which there are many detections that need to be assigned to many pedestrian candidates already being tracked from the previous frame. Each pedestrian has a unique numeric identifier that must be assigned to every future detection that contains the same pedestrian. Jianguo, Peikun and Wei (2007) says that data association is one of the central problems in multiple object tracking.

A technique often used is to model a dissimilarity function between two pedestrians, followed by an optimal assignment algorithm to find the minimum global cost between candidates, also known as the assignment problem (AP). Typical techniques for AP in multiple targets association include the Hungarian algorithm, the Joint Probabilistic Data Association Filter (JPDAF), and the Multi Hypothesis Tracking (MHT). However, it is also possible to use a greedy algorithm to find a match between two detections.

Assignment problem: has been studied at the academy for less than 50 years (CHAOBO; QIANCHUAN, 2008). It can be view as a scheduling problem where n workers are assigned to finish m jobs in which every worker can only be assigned to one job. Each worker has a cost to finish each job; then the goal is to find an optimal assignment where the overall cost is minimal. As provide in (Li; Li; Qian, 2016), the mathematical formulation can be expressed by equation (2.10) subject to equations (2.11)

and (2.12).

$$min(\gamma) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}P_{ij}, \tag{2.10}$$

$$\sum_{i=1}^{n} P_{ij} = 1, j = \{1, 2, 3, ..., n\}, \tag{2.11}$$

$$\sum_{j=1}^{n} P_{ij} = 1, i = \{1, 2, 3, ..., n\}, \tag{2.12}$$

where C_{ij} is the cost of the assignment between worker i and job j and P_{ij} represents whether i is assigned to job j ($P_{ij} = 1$ if yes, otherwise $P_{ij} = 0$).

Kuhn–Munkres: is an optimal algorithm for the assignment problem, also known as the Hungarian algorithm (KUHN, 1955). It is a combinatorial optimization algorithm that solves the assignment problem in polynomial time $O(N^3)$. The algorithm takes as input a cost matrix C, where C_{ij} is the cost between two pedestrian states i and j. It solves in four steps in which the matrix C is manipulated to give the optimal matching.

Applied to the MOT problem, each entry in the cost matrix C is populated with the dissimilarity between two pedestrians in a way that a high dissimilarity represents a high cost, and vice versa. In the literature, it is possible to find several techniques for calculating dissimilarity between two pedestrians. Geometric techniques include the simple Euclidean distance between two central points, the Jaccard index, or template matching. Non-geometric methods use a trained model to generate deep features, in a similar way commented in section 2.2.1, of an image crop that can be compared using metrics such as the cosine distance, the Mahalanobis distance, or the Hamming distance. Some of these metrics are presented next.

Jaccard index: is defined by equation (2.13) which computes the intersection area over the union area JC of two bounding boxes A and B. It is a simple geometric approach that works well while the difference between two boxes is small enough to cause overlapping, which often happens in high frame videos where the pedestrian movement between two consecutive frames is small. However, it can be a problem when two bounding box does not overlap since it will return the cost value zero or when the scene presents high crowed pedestrians whose boxes overlap several others boxes.

$$JC(A, B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid} = \frac{\mid A \cap B \mid}{\mid A \mid + \mid B \mid - \mid A \cap B \mid} \tag{2.13}$$

Cosine similarity: is used to compute the similarity between two feature vectors (e.g. deep features extracted by a CNN model trained for pedestrian reidentification). The cosine similarity CS can be computed between two vectors $\vec{A}$ e $\vec{B}$ using equation (2.14) where A_i and B_i are components of the respectively vectors.

$$CS(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\left\|\vec{A}\right\| \left\|\vec{B}\right\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.14}$$

The total cost of matching a pair of pedestrians can be computed using more than one technique. For example, equation (2.15) shows the total cost W between two pedestrian A and B represented by a weighted combination of two techniques, the Jaccard index, and the cosine similarity. The parameters α and β weights the importance of each technique for the total cost between the pedestrians.

$$W(A, B) = \alpha \cdot JC + \beta \cdot CS \tag{2.15}$$

Pedestrian reidentification: is a technique used to try to reidentify a dead pedestrian. A pedestrian is considered dead when he is too many frames without receiving detection. Usually, when this happens, no attempt is made to associate a detection with it again. However, in some scenarios, it may happen that the pedestrian is only occluded by some background structure, but it takes so many frames that is enough for the tracker to kill it. To handle this, the tracker can compare the detections of the current frame with all pedestrians considered dead using the cosine distance from equation (2.14), if the similarity is less than a threshold, the pedestrian can be considered active again.

3 THE PROPOSED METHODS

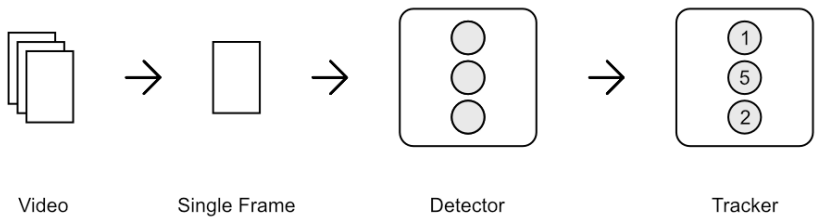
This chapter presents the details about the proposed method for online tracking of multiple pedestrians from a sequence of frames. The implementation was made using the C++11 language in a Linux environment. Two methods to compute the dissimilarity between pedestrian were implemented, and both were tested with and without pedestrian reidentification, summing up four methods tested to achieve the goal.

Section 3.1 briefly presents the organization of the application as a whole. Sections 3.2 and 3.3 go through implementation details about the detection and tracker components.

3.1 Overview

This work implemented an application to perform pedestrian tracking from a single camera using the tracking-by-detection paradigm. Hence, the application consists of two independent components: the detector and the tracker. The figure 3.1 illustrates the idea of this paradigm. The application makes use of the detector to create a list of pedestrians detected in each frame; then, it feeds the next component using the list to associate a numeric identification for each detection. Each identification number represents a unique pedestrian.

Figure 3.1: Simplified pipeline of the application



Source: The author

The detector component uses a pre-trained CNN model to identify the pedestrians present in the scene. The YOLO v3 (REDMON; FARHADI, 2018) model was used since it is the state-of-the-art in single-shot detection, being able to return the confidence scores on pre-determined regions relatively accurate and fast. Note that any other type of pedestrian detector can be used as long as the same input and output interface is maintained.

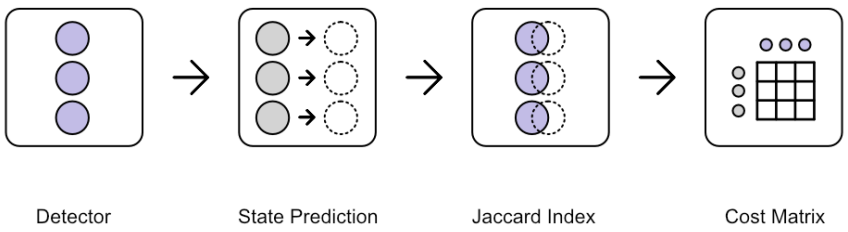
The network output layers contain a fixed number of regions with a confidence

score for each one of the objects of interest that the model was trained to detect. Only results with confidence above a threshold value are selected determined by the $detconf$ command-line argument of the application. Then, the non-maximum suppression (NMS) algorithm is used in an attempt to eliminate multiple detections for the same object when the Jaccard index is larger than a threshold value of given by $detnms$ argument. Finally, a linear search is performed to select only pedestrians from all possible objects that the model was trained to detect. The results are stored in a list type data structure.

The tracker component receives the detection list and attempts to give a unique numeric identifier to each detection asserting that it is unique for each pedestrian throughout the whole video sequence. A cost function is modeled to compute the dissimilarity between two pedestrians in which a high cost means that the detections are not similar. Thus, each of the detections sent by the detector is compared to each active pedestrian in the tracker producing a cost matrix C where each row of the matrix contains an active pedestrian and each column a current frame detection. All metrics used to fill the cost matrix are normalized into range $(0, 1)$. Two different technique were implemented to compute the cost between two detections.

(1) Geometric method: uses the Kalman filter as a state estimator to predict the bounding box position of each active pedestrian. Then the cost is computed using the Jaccard index between each predicted box and each detected pedestrian on the current frame. If a match occurs, the detection boundary box is used to fill the measurement vector $\vec{z}$ to update the filter. The filtering is realized by using the information about the updated filter state (a posteriori) as a result instead of the detected pedestrian box. Figure 3.2 illustrates the pipeline using this method.

Figure 3.2: Geometric method pipeline

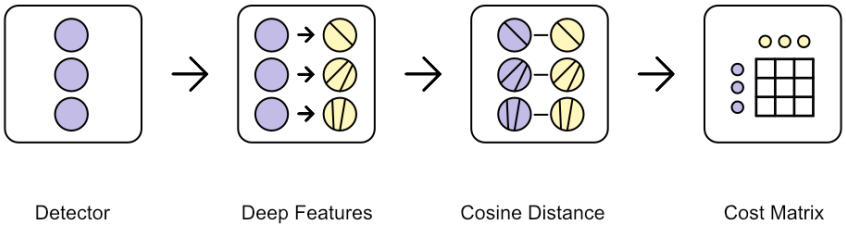


Source: The author

(2) Appearance method: uses the Intel OpenVINO *person-reidentification-retail-0031* (OpenVINO, 2019) model trained for pedestrian reidentification to create a deep feature vector from the frame pixels that are within the bounding box given by the detec-

tion. Then the cost is computed using the cosine distance between two vectors containing the deep features of pedestrians. No prediction is used, and the assigned detection box information is used as a result without any kind of filtering. Figure 3.3 illustrates the pipeline using this method.

Figure 3.3: Appearance method pipeline



Source: The author

Both methods use the Hungarian algorithm to solve the assignment problem in complexity $O(N^3)$ by finding the best matches between a detection and a pedestrian (D, P) in which the global cost is minimum given a cost matrix C. However, the association of a detection with a pedestrians is only accepted if the cost is less than the threshold value. The threshold value for the cosine distance is determined by the argument $thcos$, while the Jaccard index by $thiou$.

Track births and deaths are performed by analyzing the solution given by the association algorithm. If a detection from the current frame is not present in the assignment solution, a new pedestrian containing a new unique numerical identifier is created, formalizing a birth event. The death of a pedestrian occurs when he does not receive detection for a number of frames. Each pedestrian has an age property that is incremented at each frame in which it receives no detection, but zeroed when an association is made. If the age counter exceeds a threshold value given by $thage$, the pedestrian track is killed.

3.2 Detection

The pedestrian detector component receives a single frame as input and returns a list of detected pedestrians, which is encapsulated inside a detection class that contains the data needed for the tracking operation. The most essential is the position and the size of detection in image coordinates, a normalized $(0, 1)$ score about how sure the detector is about the classification of the object as a pedestrian, the frame number at which detection

occurred, and a slice of the frame containing the pixels delimited by the boundary box of detection. The list of information encapsulated by the class that can be seen in table 3.1.

We tested two loaders for the pre-trained model YOLO v3. The first was compiling the Darknet framework from source with CUDA support to take advantage of the computational power of the GPU. The second was using the CNN module of the OpenCV library, which only uses CPU instructions to forward an image through the network. Although only Darknet uses GPU support, both methods detect objects using approximately the same time. This can be explained because the CNN module loader is exceptionally optimized for CPU usage using specific instructions that were able to approximate the performance of the GPU installed on the testing machine, which is specified in section 4.2.

A different approach can also be used to grab detection by reading comma-separated lines from text files containing pedestrian detections provided by the dataset MOTChallenge (LEAL-TAIXé et al., 2015) used in this work to perform the evaluation. These text files contain current detections of different methods for each of the videos provided by the dataset. This allows for a diversified and standardized way to evaluate the tracker component by feeding it with different detector types and at the same time ensuring that all proposed benchmark solutions use the same detections.

When using a model loader, each frame is forwarded through the pre-trained network, which contains the right weights and nonlinear functions to transform the frame image until the output layer be filled with confidence scores for each target object of interest. The frame is an array of pixels containing three bytes per pixel (RGB colour-space). Before sending it to the network, the frame needs to be modified as the network expects it in a particular input standard. The model used in this work requires that each image should use the RGB color-space with normalized values (between zero and one at each channel), and have a size of 416 rows and 416 columns. After forwarded through the network, the resulting structure contains a matrix of size $10,647 \times 85$ (rows x columns) that is grabbed from the output layers of the CNN model. It can be decomposed as three matrix of size 507×85, 2028×85, and 8112×85. Each matrix contains different scales of the region of interest that may contain a pedestrian.

Each row of the matrix contains information about the size and position of the region of interest followed by 81 confidence scores, one per each object class in which the CNN was trained since it was used the COCO dataset. The matrix contains $10,647$ rows. In other words, the model analyses $10,647$ regions of interest grouped in three different

Table 3.1: Detection class structure

Variable	Type	Description
Id	Integer	Unique identification number of the detection per frame
x1	Float	The left most x-axis value of the boundary box
y1	Float	The top most y-axis value of the boundary box
x2	Float	The right most x-axis value of the boundary box
y2	Float	The bottom most y-axis value of the boundary box
Conf	Float	Confidence about the detected object being a pedestrian
Img	Image	A reference to the vector of pixels containing inside the boundary box of the detection
AssignId	Integer	The pedestrian identification assigned by the tracker

Source: The author

scales (small, medium, and big) and fills up 81 confidence scores for each object class. At the end of each network forward, a linear search is made to find the best candidates by checking which ones have the highest scores in the column containing the pedestrian class. Candidates with a low confidence score are discarded by a confidence threshold specified by $detconf$. After that, a non-maximum-suppression (NMS) algorithm is used to try to filter objects that may contain more than one boundary box over them. A Jaccard index threshold is given by $detnms$.

If the detections come from the dataset text file, the detector reads the text file, and it keeps in memory a data structure containing a list of detection objects. The text file is a comma separated values containing ten variables but only six are used: $frame$, x, y, w, h and $confidence$. The variable $frame$ contains the frame number to which the detection belongs. The two variables x and y report the position in image coordinates of the top-left point of the detected boundary box, while w and h report the width and height of this box. The last variable $confidence$ contains the detector's confidence score on the object contained in the boundary box being a pedestrian class.

Table 3.2: Pedestrian class structure

Variable	Type	Description
Id	Integer	The unique identification number of this pedestrian
DetList	Pointer	Reference to a list of detections objects assigned to this pedestrian
Age	Integer	How many frames since the last detection assigned to this pedestrian
IsDead	Boolean	Keep information if the pedestrian is active or not
Estimator	Pointer	The estate estimator object of this pedestrian

Source: The author

3.3 Tracking

In each frame, the tracker receives a list of detections of the current frame from the detector. Then it needs to give an identification number to each detection. If a frame detection is not associated with any pedestrian, it generates a new pedestrian. If a pedestrian receives no detection, the tracker must decide whether to kill him or not.

In the first frame containing pedestrian detections, no pedestrian is being tracked. Thus, the tracker creates a new pedestrian ID for each detection received without any filtering. When the tracker creates a new pedestrian object, an instance of the pedestrian class is allocated in heap memory. This object is initialized with a list containing only the detections that originated it. The KF is also initialized at this time since each pedestrian has its owns estate estimator.

The tracker maintains a pedestrian list that contains pedestrians that may be active or inactive. A pedestrian is considered active when the class variable $IsDead$ has the logical value false. Each pedestrian holds information about its state, such as the list of detections that was assigned to it. The last detection of the list holds information about the last boundary box of the pedestrian. Table 3.2 shows the pedestrian class structure.

3.3.1 Prediction

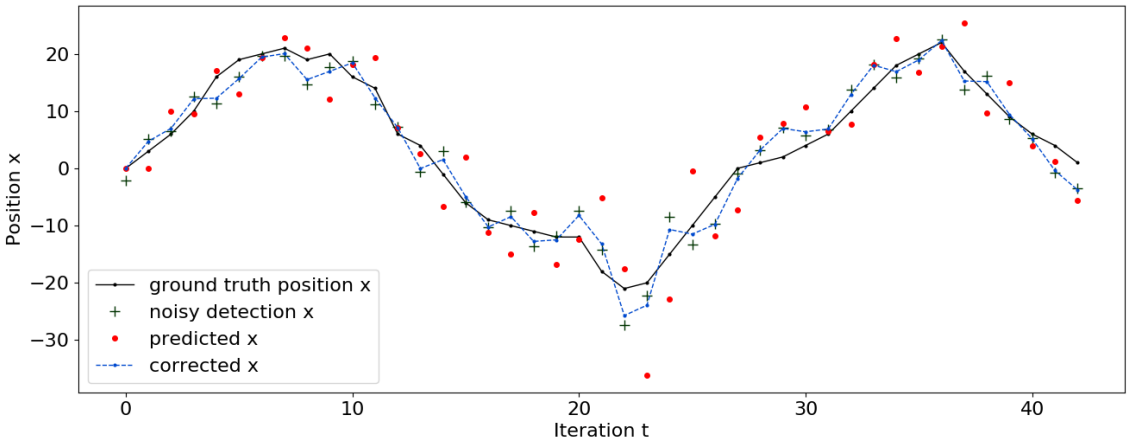
The KF algorithm is used to estimate a priori the state of the pedestrian at the current frame instead of using the last assigned detection information from some previous frame. This technique may increase the chances of a correct match when using the Jaccard index to compare the bounding boxes of two pedestrians. It is also used as a filter

since the tracker outputs data extracted from the current filter state vector $\vec{x}$, which combine information about prediction and measurement, instead of the raw data from the last assigned detection.

The state vector $\vec{x}$ and the covariance of the state error $\mathbf{P}$ are *predicted* each frame and *updated* when an assignment of a frame detection to a pedestrian object is made. The position, size, and position difference between the last two assigned bounding boxes $(x_1, y_1, x_2, y_2, v_x, v_y)$ is used to populate the measurement vector $\vec{z}$ which is then used by the algorithm to update its variables. The covariance of the state error $\mathbf{P}$ requires that the covariance matrix of the process $\mathbf{Q}$ and measurement noise $\mathbf{R}$ be defined.

Finding the right parameters to optimize the algorithm is challenging since there is no information about camera calibration. And even if we find it for one type of video, it may not work well for another type of camera perspective and target motion. In other words, it is difficult to find the right parameters that work well for all kind of camera position and camera movement available in real world or in the dataset used to test the application. Furthermore, each pedestrian can have its own random movement, but this tends to follow a small variance that can be modeled into the algorithm.

Figure 3.4: Kalman simulation



Source: The author

As a guide for choosing acceptable parameters for $\mathbf{Q}$ and $\mathbf{R}$, we implemented a Python script to simulate the filtering and prediction of the movement of a pedestrian in the x-axis from real values found by analyzing the PETS09 sequence, and the plot of the simulation is shown in Figure 3.4. Since our motion model for prediction is simplified, we chose to trust more in the detector information, and hence selected high values for process noise $\mathbf{Q}$ and lower for measurement noise matrix $\mathbf{R}$. The process noise matrix is an identity matrix with diagonal filled with value 12.0, while the measurement noise matrix is an identity matrix with diagonal filled with value 5.0. A small covariance value

of 0.1 was added to all elements to both matrices.

The vector state $\vec{x}$ was modeled using six variables: x_1, y_1, x_2, y_2, v_x, v_y. Variables x_1 and y_1 represent the top-left point of the bounding box, delimiting the last associated detection of that pedestrian; x_2 and y_2 relate to the bottom-right point. The last two variables v_x and v_y represent the position difference between the two last detected box assigned (i.e., the velocity).

The filter requires an initial guess of the state vector $\vec{x}$. The four positional variables are initialized with the information about the first boundary box, and the velocity information was initialized with zero values.

The prediction of the state $\vec{x}$ is made simple by modeling the classical mechanics model for constant velocity motion. The control matrix $\mathbf{B}$ and the control vector $\vec{u}$ was not used, and we defined the transition matrix $\mathbf{F}$ as a 6×6 as shown in equation (3.1). This matrix was built assuming the constant velocity motion model provided in equation (3.2).

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

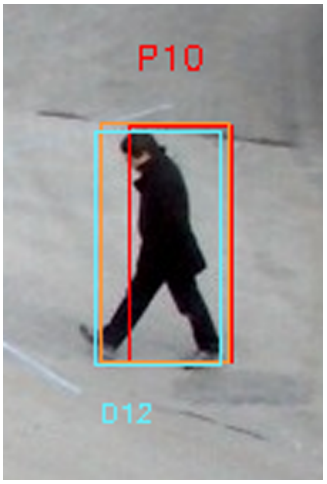
$$\begin{cases} \hat{x_1} = x_1 + v_x \\ \hat{y_1} = y_1 + v_y \\ \hat{x_2} = x_2 + v_x \\ \hat{y_2} = y_2 + v_y \\ \hat{v_x} = v_x \\ \hat{v_y} = v_y \end{cases} \tag{3.2}$$

The measurement relation matrix $\mathbf{H}$ is specific in order to mask which state values will be taken into account when calculating the error between the measured vector $\vec{z}$ and the predicted state $\hat{\vec{x}}$ where it will later be used to correct the prediction by a factor dictate by the Kalman gain value. Since our measurement matrix contains all the elements used inside the state vector, the relation matrix $\mathbf{H}$ is modeled as a 6×6 identity matrix to account for all the six variables of the state vector $\vec{x}$.

The state velocity variables v_x and v_y are updated whenever an association is made on the pedestrian, calculating a horizontal and vertical distance between the last two detections associated with the pedestrian.

Figure 3.5 shows an empirical example of this tracker predicting the new state of one pedestrian where it is possible to notice a satisfactory result in the prediction of the state. The red box shows the last assigned detection at frame t while the orange box shows

Figure 3.5: Prediction result



Source: Adapted from PETS09

the predicted state at frame $t+1$. The blue box shows the detected pedestrian at frame $t+1$, notice how it almost matches the predicted orange box in this frame example.

3.3.2 Data Association

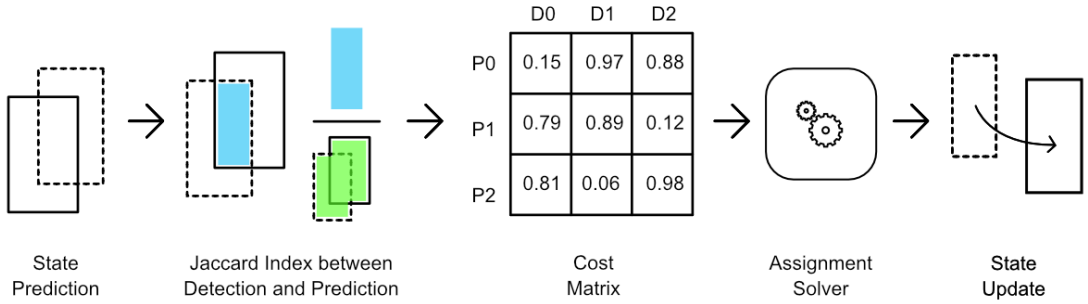
The goal of the association task is to find the best match between a pedestrian P and a frame detection D. At each frame, the detector feeds the tracker component with a list of detections. Each detection needs to be analyzed and compared with all currently active pedestrians to see if an association is possible.

We have implemented two methods to calculate dissimilarity between two pedestrians: a geometric and an appearance method. Both may or may not be followed by a function that attempts to reidentify pedestrians considered dead.

The geometric method: uses the KF algorithm to a priori estimate the bounding box (x_1, y_1, x_2, y_2) of each active pedestrian P in the tracker. A pedestrian is considered active if he is within four frames without receiving associations. Each predicted bounding box is compared to the detected bounding box coming from the previous detector component.

The comparison is performed using the Jaccard index, which returns a normalized value $(0, 1)$ containing a metric of how much intersection area there is between the two rectangles. Thus, a cost matrix C is populated by computing the metric among all candidates. Each row of the matrix represents a prediction of an active pedestrian, and each column represents a frame detection. Figure 3.6 illustrates this idea.

Figure 3.6: Geometric method pipeline



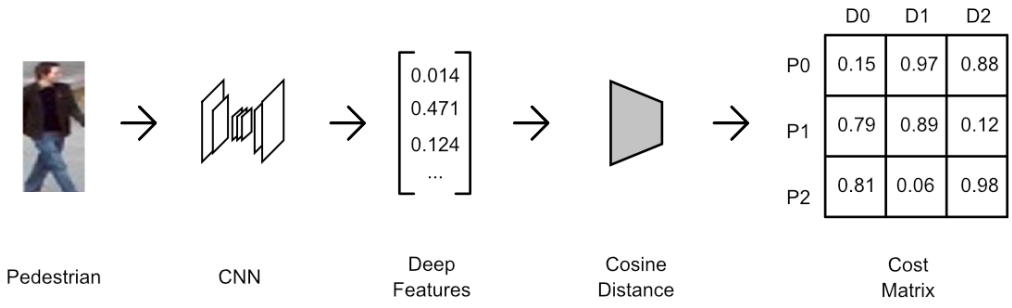
Source: The author

The appearance method: uses the pre-trained pedestrian reidentification model *person-reidentification-retail-0031* (OpenVINO, 2019) to produce a vector of deep features for each detection fed by the detector component. This model is based on the RMNet backbone that was developed for fast inference. At the Market-1501 benchmark (Zheng et al., 2015), used for pedestrian reidentification, it got an accuracy of 0.7791 and a precision of 0.6180.

The detection bounding box is used to delimit a section area in the frame in which the detection was made. The frame is cropped to create a separated sub-image, then it is resized to 96×48 (rows x columns), and forwarded throughout the model. The output layer contains 256 floating-point numbers, where each element represents a deep feature of the pedestrian image.

To compute the dissimilarity between active pedestrians and current frame detections, equation (2.14) is used to compute the cosine similarity between two deep feature vectors. The vector of the last detection associated with the pedestrian is compared to the vector of each detection sent by the previous component. This way, all applicants are compared until the cost matrix C is populated. This process is illustrated by figure 3.7.

Figure 3.7: Appearance method pipeline



Source: The author

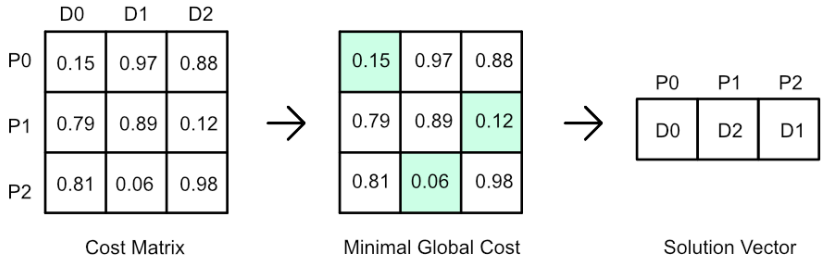
In both methods, a cost matrix C of size M x N is filled in each frame. The value M is the number of active pedestrians, and N is the number of detections fed by the

detector component. All elements of matrix C are normalized values in the range range $(0, 1)$.

The Hungarian algorithm is used to solve the data association problem optimally. The algorithm receives a cost matrix C and outputs the solution vector S. Each vector position represents a pedestrian used when the cost matrix was computed. The value of each position contains an integer representing the detection assigned to the pedestrian. By analyzing the solution vector, it is possible to extract each match (D, P) between a detection and a pedestrian asserting that the overall cost is the minimum.

This solution contains the best matches in order to minimize the overall cost; however it may not contain the best solution for the tracker. To try to reduce the number of wrong matches, only matches whose cost is less than a threshold value are accepted. The variables, passed as command line argument, $thcos$ and $thiou$ define the threshold values for the cosine distance and the Jaccard index, respectively. When a match has a higher cost than the set threshold value, it is removed from the solution. The figure 3.8 shows the assignment pipeline from the cost matrix to the solution vector.

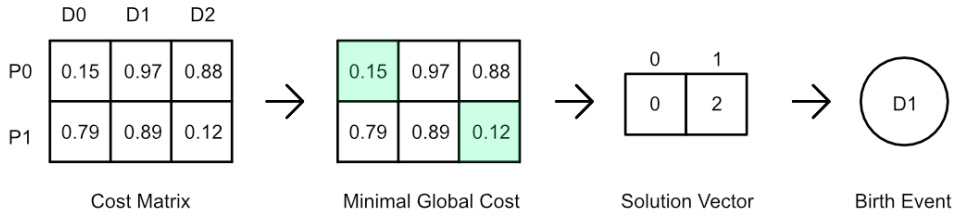
Figure 3.8: Assignment problem pipeline



Source: The author

Birth: is an event that occurs for every detection received by the detector that was not associated with any pedestrian. This occurs when there are more detections than pedestrians or when the detection is removed from the solution because it does not cost less than the configured threshold value. The figure 3.9 shows an example where the detection D_1 is not present in the solution vector causing the creation of a new pedestrian.

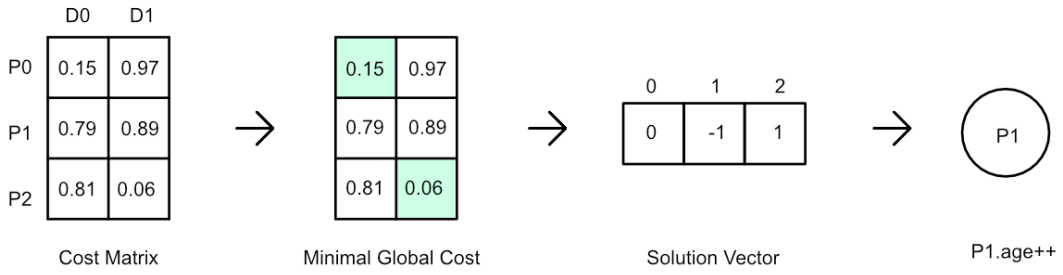
Figure 3.9: Birth event pipeline



Source: The author

Death: is an event that occurs when the pedestrian's age becomes greater than the threshold value. The age of the pedestrian is a variable that is incremented in every frame that it receives no detection. The threshold value used to cause a pedestrian to die is defined by $thage$. Thus, in each frame, every pedestrian has his age checked. If he is older than $thage$, he is killed by the tracker. The figure 3.10 shows an example where pedestrian P_1 is aged since it receives no detection in the current frame.

Figure 3.10: Death event pipeline



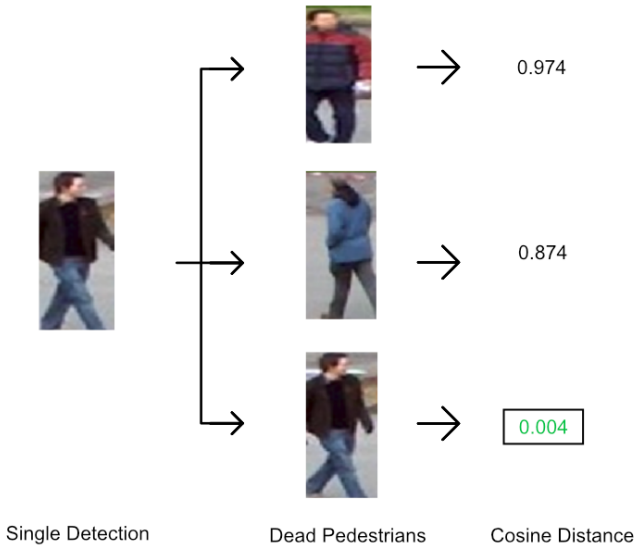
Source: The author

Pedestrian reidentification: is the last step of the tracker component, and it is used as an attempt to revive dead pedestrians. The same technique used in the *appearance method (2)*, i.e., the cosine distance between two deep feature vectors is used to attempt to reidentify a dead pedestrian. If the cosine distance between a dead pedestrian and a detection not assigned to any pedestrian is less than a threshold value specified by $threborn$ (reborn-threshold), an association between them is made. Then the pedestrian is considered live again for the next frames.

A time variable can be used to control the maximum time to try to reborn a dead pedestrian. Applications that may benefit from a long time are surveillance systems as having in memory the characteristics of an pedestrian to try to reidentify after a long time may be beneficial for future investigation. Other types of applications where pedestrians are unlikely to return after a certain time can benefit from the implementation of this

control variable. This work does not implement this feature, considering that the time is infinite, i.e., the application keeps trying to reidentify dead pedestrians while there are frames coming from the stream. The figure 3.11 illustrates an example where a single detection is compared to all dead pedestrian using the cosine distance between the deep features of each patch, the last comparison showing a distance less than the threshold, so that detection is assigned to the dead pedestrian, creating a reborn event.

Figure 3.11: Reidentification pipeline



Source: The author

4 RESULTS

4.1 MOT Evaluation

A popular benchmark for evaluating solutions to the MOT problem is the MOTChallenge (LEAL-TAIXé et al., 2015; MILAN et al., 2016). It provides a selection of public videos and standardized evaluation metrics, which includes the CLEARMOT (BERNARDIN KENI, 2008) and another set of metrics by Li, Huang and Nevatia (2009). The videos are selected from other public datasets such as the PETS (Ferryman; Shahrokni, 2009), which is a well know dataset targeted for video surveillance.

The MOTChallenge includes the two widely used metrics into the community: the Multiple Object Tracking Accuracy (MOTA) and the Multiple Object Tracking Precision (MOTP) (KASTURI et al., 2009; Mekonnen; Lerasle, 2019) as shown by equations (4.1) and (4.2), respectively.

$$MOTA = 1 - (F_p + F_n + Id_{sw}), \quad (4.1) \qquad MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t C_t}, \quad (4.2)$$

where $F_p = \frac{\sum_t FP_t}{\sum_t g_t}$ is the ratio of false positives over the number of ground truth objects from the whole dataset, $F_n = \frac{\sum_t FN_t}{\sum_t g_t}$ the ratio of false negatives, and $Id_{sw} = \frac{\sum_t Id_{sw,t}}{\sum_t g_t}$ the ratio of identification switches. The d_t^i is the Euclidean distance between the matched ground-truth location and the tracker target location. The C_t is the total number of matches made.

The $MOTA$ accounts for all errors, such as missed target or false positives. The $MOTP$ tries to account for the ability of the tracker to estimate precise positions, independent of its results from the detection step, keeping consistent trajectories. Detailed information is presented in (BERNARDIN; STIEFELHAGEN, 2008). This work will use the $MOTA$ and $MOTP$ to evaluate the implementation since those metrics are widely used in the literature. From Leal-Taixé et al. (2015), *"MOTA is perhaps the most widely used figure to evaluate a tracker's performance because it combines three important error metrics in one"*.

To generate the evaluation metrics, we used the python implementation *py-motmetrics* (HEINDL, 2019) compatible with the original MOTChallenge devkit scripts in Matlab.

We used the training group of videos because it was the only one we could access

with ground truth to perform the evaluation locally. We did not use any of the videos named as training to train any kind of model in our implementation. All videos were used for testing purposes only. To perform the evaluation using the videos from the test group, it is necessary to send the results in a text file to the benchmark server. However, to be able to do this, it is necessary to register on the site with an institutional email. Until the publication of this work, we did not obtain registration approval with the email used.

4.2 Environment

The evaluation was made using the Ubuntu Linux 18.04 environment installed on a computer using Intel i7 4710HQ CPU up to 2.50GHz, 16GB memory, and NVIDIA GeForce GTX 970M GPU which offers 1280 CUDA cores and 980MHz maximum clock rate. The major implementation of the tracker was compiled using C++11 standard, although some libraries used the C++98 standard and later linked to the main program. Additional libraries like Darknet, OpenCV, NVIDIA CUDA, Intel OpenVINO, and NVIDIA cuDNN were compiled from sources and installed into the environment or static linked to the main program.

The use of CUDA and cuDNN support was motivated since they increase the performance of machine learning models, which are known by being computational heavy, by using GPU resources which allow high parallelization and avoid the use of the main bus since the computation is made inside the cores and memory of the GPU. For example, the frame process time when using the Darknet loader was improved from 16 seconds per frame to an average of 30 milliseconds per frame after the compilation of the framework using CUDA support.

4.3 Results

Different threshold values informed via the command line was tested to try to improve the results using a shell script to partially automate the generation of the evaluation metrics, i.e., all threshold values used in this work was found by trial and error. The best threshold values was found by selection the highest metrics generated by the dataset evaluation script and also visually analyzing the results. Once the values were found, they were used to produce the results of the four methods tested. Tables 4.2, 4.3, 4.4 and 4.5

show the results obtained by each method when tested in each of the 11 MOTChallenge dataset videos when using the threshold values given by table 4.1. Table 4.6 displays statistical data on the MOTA metric for each method.

Table 4.1: Threshold values used to produce the results

Threshold	Value
-detconf	0.73
-detnms	0.33
-thcos	0.81
-thiou	0.60
-threborn	0.72
-thage	4

Source: The author

The first four tables have five columns each, where **Sequence** is the video being tested, **MOTA** the value computed from equation (4.1), **MOTP** from equation (4.2), $\mathbf{D_t}$ is the average time duration per frame of the detector component in milliseconds, and $\mathbf{T_t}$ is the average time duration per frame of the tracker component also in milliseconds.

The last table (4.6) contains the average (AVG), the standard deviation (SD), the maximum value (Max), and minimum value (Min) of the MOTA metric for each tested method among all videos. The blue background means the best score between all methods, and the red background the worst.

Table 4.7 presents the results of the top 10 trackers published on the MOTChallenge (LEAL-TAIXé et al., 2019) website in the benchmark 2D-MOT-2015 ordered by MOTA metric. They were tested on another group of videos (test videos) and not the videos used to evaluate our work (train videos), so we understand that it is not fair to make a comparison between the methods. Even so, we are publishing the results here as an example of what value of MOTA is achieved by these works.

Two frames are presented containing an empirical result of the track lines after running the application into the PETS2009 sequence. Figure 4.3 shows the state of the tracker after few frames analyzed. It is possible to notice that two pedestrians were born in the right corner and began to be tracked by the application. Figure 4.3 shows the state of the tracker after several frames of the PETS2009 sequence are processed. Each track line color represents a pedestrian identified by the tracker. The same color may be used more than one time for two different pedestrians since the number of pedestrians is high

in this sequence. In the footer is the cropped images of the detected pedestrians whose pixels are used to produce the deep features.

Table 4.2: Evaluation of appearance method using reidentification

Sequence	**MOTA**	**MOTP**	$\mathbf{D_t}$	$\mathbf{T_t}$
ETH-Bahnhof	21.7%	0.232	445	59
ADL-Rundle-8	40.7%	0.268	744	58
Venice-2	40.4%	0.247	389	61
KITTI-17	42.3%	0.272	722	66
TUD-Campus	56.8%	0.252	400	45
KITTI-13	14.8%	0.298	724	66
ADL-Rundle-6	52.2%	0.241	462	51
ETH-Pedcross2	56.0%	0.243	445	40
ETH-Sunnyday	66.4%	0.199	398	55
TUD-Stadtmitte	79.9%	0.229	380	44
PETS09-S2L1	79.5%	0.261	344	38
OVERALL	48.3%	0.245	495	53

Source: The author

Table 4.3: Evaluation of appearance method without reidentification

Sequence	**MOTA**	**MOTP**	$\mathbf{D_t}$	$\mathbf{T_t}$
ETH-Bahnhof	21.6%	0.232	334	39
ADL-Rundle-8	40.4%	0.267	342	40
Venice-2	40.3%	0.247	331	36
KITTI-17	41.4%	0.271	384	22
TUD-Campus	56.0%	0.252	340	28
KITTI-13	14.8%	0.297	331	06
ADL-Rundle-6	52.0%	0.241	332	33
ETH-Pedcross2	55.8%	0.243	340	28
ETH-Sunnyday	66.4%	0.199	341	32
TUD-Stadtmitte	79.5%	0.228	342	29
PETS09-S2L1	79.2%	0.261	339	28
OVERALL	48.1%	0.245	341	29

Source: The author

Table 4.4: Evaluation geometric method using reidentification

Sequence	**MOTA**	**MOTP**	$\mathbf{D_t}$	$\mathbf{T_t}$
ETH-Bahnhof	-26.0%	0.232	509	82
ADL-Rundle-8	25.9%	0.267	501	82
Venice-2	34.0%	0.246	665	69
KITTI-17	27.4%	0.271	501	35
TUD-Campus	47.9%	0.252	499	53
KITTI-13	-0.8%	0.295	495	14
ADL-Rundle-6	44.7%	0.240	496	65
ETH-Pedcross2	29.7%	0.242	497	52
ETH-Sunnyday	47.0%	0.199	504	65
TUD-Stadtmitte	78.3%	0.225	498	54
PETS09-S2L1	63.6%	0.261	497	54
OVERALL	29.7%	0.245	514	56

Source: The author

Table 4.5: Evaluation of geometric method without reidentification

Sequence	MOTA	MOTP	D_t	T_t
ETH-Bahnhof	15.2%	0.232	885	128
ADL-Rundle-8	38.1%	0.267	721	111
Venice-2	39.7%	0.247	517	77
KITTI-17	35.9%	0.271	530	36
TUD-Campus	52.9%	0.252	514	56
KITTI-13	-0.7%	0.295	517	14
ADL-Rundle-6	50.7%	0.242	511	70
ETH-Pedcross2	52.6%	0.243	548	66
ETH-Sunnyday	61.5%	0.199	552	79
TUD-Stadtmitte	79.1%	0.228	539	64
PETS09-S2L1	74.3%	0.261	496	58
OVERALL	44.8%	0.245	575	69

Source: The author

Table 4.6: Statistics of MOTA for each method

Method	AVG	SD	Max	Min
(1) Appearance using reidentification	50.06	20.97	79.9	14.8
(2) Appearance without reidentification	49.76	20.90	79.5	14.8
(3) Geometric using reidentification	33.79	28.72	78.3	-26.0
(4) Geometric without reidentification	45.39	23.62	79.1	-0.7

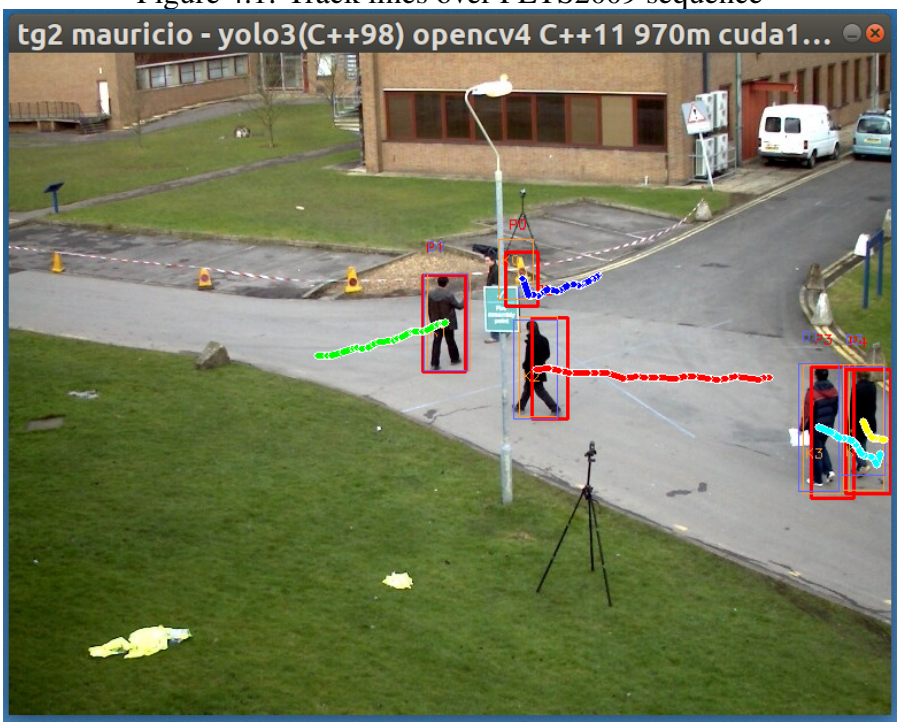
Source: The author

Table 4.7: Result from 2D MOT 2015 benchmark

Tracker	MOTA	SD +/-
MPNTrack15	48.3	12.0
Tracktor15	44.1	11.7
TLO	41.3	13.7
DeepMP	40.5	12.8
MHTREID15	40.0	16.2
CRFTrack_	40.0	14.5
TLO15	40.0	14.9
KCF	38.9	14.5
CRF-RNN15	38.9	15.1
AP_HWDPL_p	38.5	9.9

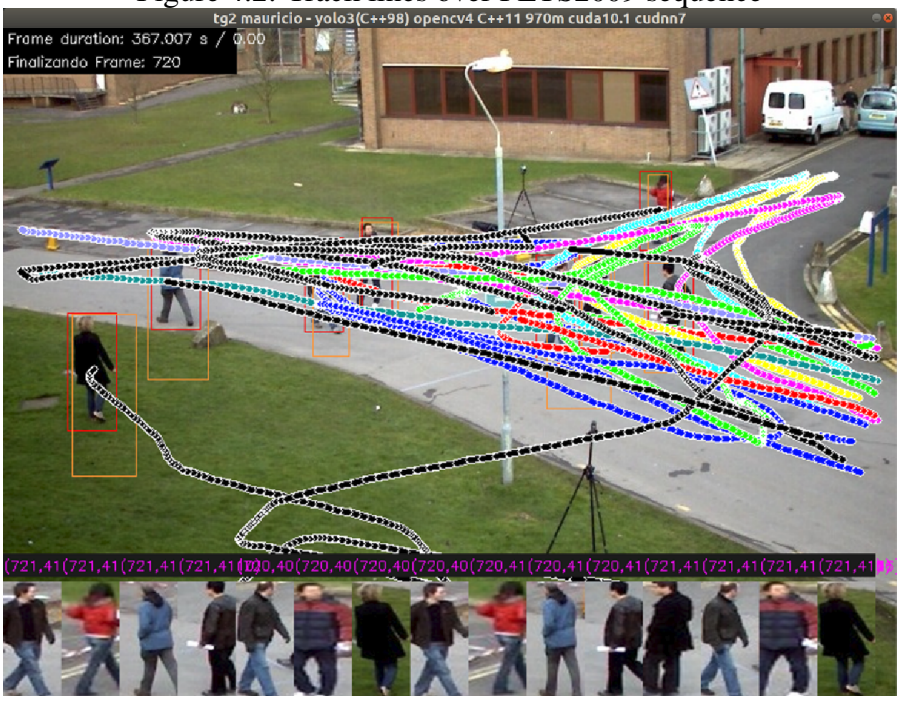
Source: Adapted from MOTChallenge

Figure 4.1: Track lines over PETS2009 sequence



Source: Adapted from PETS2009

Figure 4.2: Track lines over PETS2009 sequence



Source: Adapted from PETS2009

4.4 Discussion

The use of KF and the Jaccard index as geometric dissimilarity – methods (3) and (4) – is an acceptable approach when the pedestrian presents a small position difference between two consecutive frames and when the multiple pedestrians present are sparsely located in the scene. However, the accuracy drops dramatically when the detector fails to detect a pedestrian for a number of frames enough for having no more intersection area between the bounding boxes, which causes the Jaccard index to return a zero value. Moreover, this method also fails in scenarios with a dense number of pedestrians because several detected boundary boxes start to have intersection area in common which it is no longer sufficient to accurately distinguish which detection belongs to which pedestrian.

By using pedestrian reidentification, the results become much worse. This can be explained because reidentification begins to associate detections previously associated with other pedestrians; since the Jaccard index only uses geometric area information to associate a detection with a pedestrian, without looking at whether the other features are similar. So, when using deep features, the tracker begins to associate detections with pedestrians that actually have similar pedestrian deep features, increasing the number of identification switches, causing the MOTA metric to drop. As shown in Table 4.6, it

presents the worst metrics of all evaluated methods.

The method that uses only the cosine distance between deep feature vectors presented the two best MOTA metrics overall. Moreover, the use of pedestrian reidentification helped improve the metric. Thus, the method using deep features followed by a pedestrian reidentification achieved the best MOTA over the four tested methods. However, it still not enough to avoid identification switches because the same pedestrian can change appearance from one frame to another (in turn for example), so the cosine distance can produce a low similarity even if they are the same pedestrian causing the tracker to create a new pedestrian and lowering the MOTA metric. It also presented the lowest standard deviation, which indicates that this method got a not so sparse accuracy when different types of videos are being evaluated.

Method (1) showed a high MOTA metric of 79.5% in the sequence PETS09-S2L1, which has a scenario very close to what is found in surveillance videos. This video was used as a test during the implementation of this work, which corroborates the fact that it is possible to calibrate the tracker threshold variables to obtain a high metric for a video type, but an accurate result in a sequence does not imply that it will work well on other types of video. It is challenging to find a method and configuration that produces high accuracy for various types of scenarios.

All four methods presented very low accuracy in ETH-Bahnhof and KITTI-13 sequences. Both are sequences made by a moving camera at a low altitude. The method (3) had a negative MOTA for both sequence and method (4) only for the KITTI-13 sequence. A negative MOTA means that they had more errors than hits. Such types of videos feature very dynamic camera movements, pedestrian reflections over surfaces, and focus shifting. All of these videos present challenges that none of the methods were prepared to handle, featuring low MOTA metric.

5 CONCLUSIONS

In this work, we presented an application to track multiple pedestrians online from a single camera using the tracking-by-detection paradigm. For that purpose, we have met all the specific goals mentioned in the proposal (see section 1.2).

The evaluation of our work showed that the technique using the cosine distance between arrays of deep features followed by pedestrian reidentification had the best overall result among the 11 video types tested. The results presented different accuracy for different types of videos, ranging from a maximum MOTA metric of 79.9% to a minimum of 14.8% for the same method. In fact, the MOT problem presents a very diverse type of scenarios that are challenging to handle by a single method configuration.

The four methods presented difficulties when dealing with high levels of occlusion in scenes where the detector fails in feeding the tracker component. Even though the tracking-by-detection paradigm allows us to use two components that work separately, the result of the first component strongly affects the result of the second.

5.1 Future work

The implementation of this tracker makes use of components that can be improved in isolation by exploring new techniques, thereby verifying how they improve the overall results of the tracker.

- New techniques to perform pedestrian detections on each frame can be analyzed, such as the use of two-stage object detectors, which are slower, but may present better accuracy.
- The use of nonlinear motion model filtering can be used to try to improve the prediction and filtering.
- The use of new models to produce deep features of each pedestrian can be explored as well as types of metrics for computing distance between two deep feature vectors.
- Study filtering models that integrate linear and angular velocity to increase uncertainty when the target is changing direction.
- New template matching techniques to get better accuracy when modeling the dissimilarity between pedestrians.
- The study of the use of stereo cameras to make use of the additional depth informa-

tion to improve the tracking results.

- The use of camera calibration for background modeling to better accurate target movements.
- The study of techniques to better handle false positive detections from reflected pedestrians.

REFERENCES

ANDRILUKA, M.; ROTH, S.; SCHIELE, B. People-tracking-by-detection and people-detection-by-tracking. In: . [S.l.: s.n.], 2008.

BAE, S.; YOON, K. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1218–1225.

BERGMANN, P.; MEINHARDT, T.; LEAL-TAIXE, L. **Tracking without bells and whistles**. 2019.

BERNARDIN, K.; STIEFELHAGEN, R. Evaluating multiple object tracking performance: The clear mot metrics. **EURASIP Journal on Image and Video Processing**, v. 2008, n. 1, p. 246309, May 2008. ISSN 1687-5281. Available from Internet: <https://doi.org/10.1155/2008/246309>.

BERNARDIN KENI, S. R. Evaluating multiple object tracking performance: The clear mot metrics. **J. Image Video Process.**, Hindawi Publishing Corp., New York, NY, United States, v. 2008, p. 1:1–1:10, jan. 2008. ISSN 1687-5176. Available from Internet: <http://dx.doi.org/10.1155/2008/246309>.

BEWLEY, A. et al. Simple online and realtime tracking. **2016 IEEE International Conference on Image Processing (ICIP)**, IEEE, Sep 2016. Available from Internet: <http://dx.doi.org/10.1109/ICIP.2016.7533003>.

Breitenstein, M. D. et al. Online multiperson tracking-by-detection from a single, uncalibrated camera. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 33, n. 9, p. 1820–1833, Sep. 2011.

CHAOBO, Y.; QIANCHUAN, Z. Advances in assignment problem and comparison of algorithms. In: **2008 27th Chinese Control Conference**. [S.l.: s.n.], 2008. p. 607–611.

Chen, L. et al. Aggregate tracklet appearance features for multi-object tracking. **IEEE Signal Processing Letters**, v. 26, n. 11, p. 1613–1617, Nov 2019.

CHU, P. et al. **Online Multi-Object Tracking with Instance-Aware Tracker and Dynamic Model Refreshment**. 2019.

FAN, L. et al. A survey on multiple object tracking algorithm. In: **2016 IEEE International Conference on Information and Automation (ICIA)**. [S.l.: s.n.], 2016. p. 1855–1862.

Ferryman, J.; Shahrokni, A. Pets2009: Dataset and challenge. In: **2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance**. [S.l.: s.n.], 2009. p. 1–6.

Gaddigoudar, P. K. et al. Pedestrian detection and tracking using particle filtering. In: **2017 International Conference on Computing, Communication and Automation (ICCCA)**. [S.l.: s.n.], 2017. p. 110–115.

GIRSHICK, R. **Fast R-CNN**. 2015.

GIRSHICK, R. B. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. **CoRR**, abs/1311.2524, 2013. Available from Internet: <http://arxiv.org/abs/1311.2524>.

GUAN, H.; XUE, X.; ZHIYONG, A. Online video tracking using collaborative convolutional networks. In: **2016 IEEE International Conference on Multimedia and Expo (ICME)**. [S.l.: s.n.], 2016. p. 1–6. ISSN 1945-788X.

HEINDL, C. **PY-Motmetrics**. 2019. <https://github.com/cheind/py-motmetrics>. [Online; accessed 27-October-2019].

Jianguo, W.; Peikun, H.; Wei, C. Study on the hungarian algorithm for the maximum likelihood data association problem. **Journal of Systems Engineering and Electronics**, v. 18, n. 1, p. 27–32, March 2007.

KASTURI, R. et al. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 31, n. 2, p. 319–336, Feb 2009. ISSN 0162-8828.

KUHN, H. W. The Hungarian Method for the Assignment Problem. **Naval Research Logistics Quarterly**, v. 2, n. 1–2, p. 83–97, March 1955.

KUMAR, R. In: . [S.l.: s.n.], 2014.

LEAL-TAIXé, L. et al. MOTChallenge 2015: Towards a benchmark for multi-target tracking. **arXiv:1504.01942 [cs]**, abr. 2015. ArXiv: 1504.01942. Available from Internet: <http://arxiv.org/abs/1504.01942>.

LEAL-TAIXé, L. et al. **2D MOT 2015 Results**. 2019. <https://motchallenge.net/results/2D_MOT_2015/?chl=2&orderBy=MOTA&orderStyle=ASC&det=Public>. [Online; accessed 30-November-2019].

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature Cell Biology**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 5 2015. ISSN 1465-7392.

Lei, Y.; Huang, Z. Research on pedestrian detection algorithm based on monocular vision. In: **2018 International Conference on Robots Intelligent System (ICRIS)**. [S.l.: s.n.], 2018. p. 161–163.

LEON, F.; GAVRILESCU, M. A review of tracking, prediction and decision making methods for autonomous driving. **ArXiv**, abs/1909.07707, 2019.

Li, T.; Li, Y.; Qian, Y. Improved hungarian algorithm for assignment problems of serial-parallel systems. **Journal of Systems Engineering and Electronics**, v. 27, n. 4, p. 858–870, Aug 2016.

LI, X. et al. A survey of appearance models in visual object tracking. **CoRR**, abs/1303.4803, 2013. Available from Internet: <http://arxiv.org/abs/1303.4803>.

Li, Y.; Huang, C.; Nevatia, R. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 2953–2960. ISSN 1063-6919.

LIU, W. et al. Ssd: Single shot multibox detector. **Lecture Notes in Computer Science**, Springer International Publishing, p. 21–37, 2016. ISSN 1611-3349. Available from Internet: <http://dx.doi.org/10.1007/978-3-319-46448-0_2>.

MALLICK, M. et al. Introduction to the issue on multitarget tracking. **IEEE Journal of Selected Topics in Signal Processing**, v. 7, n. 3, p. 373–375, June 2013. ISSN 1932-4553.

MAYBECK, P. S. et al. **Stochastics Models, Estimation, and Control: Introduction**. 1979.

Mekonnen, A. A.; Lerasle, F. Comparative evaluations of selected tracking-by-detection approaches. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 29, n. 4, p. 996–1010, April 2019.

MILAN, A. et al. Mot16: A benchmark for multi-object tracking. **arXiv:1603.00831 [cs]**, mar. 2016. ArXiv: 1603.00831. Available from Internet: <http://arxiv.org/abs/1603.00831>.

OpenVINO. 2019. <https://docs.openvinotoolkit.org/2019_R1/_person_reidentification_retail_0031_description_person_reidentification_retail_0031.html>. [Online; accessed 02-December-2019].

REDMON, J. et al. **You Only Look Once: Unified, Real-Time Object Detection**. 2015.

REDMON, J.; FARHADI, A. **YOLOv3: An Incremental Improvement**. 2018.

REN, S. et al. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**. 2015.

SERMANET, P. et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. **CoRR**, abs/1312.6229, 2013. Available from Internet: <http://arxiv.org/abs/1312.6229>.

SORENSON, H. W. Least-squares estimation: from gauss to kalman. **IEEE Spectrum**, v. 7, n. 7, p. 63–68, 1970.

WANG, Z. et al. Towards real-time multi-object tracking. **ArXiv**, abs/1909.12605, 2019. Available from Internet: <https://arxiv.org/pdf/1909.12605.pdf>.

WELCH, G.; BISHOP, G. An introduction to the kalman filter. 2006.

Zheng, L. et al. Scalable person re-identification: A benchmark. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1116–1124. ISSN 2380-7504.

APPENDIX A — TG1

Pedestrian Tracking-by-Detection from Static Monocular Camera using Deep Learning Object Detector and Filtering Algorithm

Mauricio da Costa Justo Ize[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil

mcjize@inf.ufrgs.br

Abstract. *Automated video analysis is a growing demand since technology has been allowing the creation of visual data in increasing rates coming from different real-world applications. Pedestrian detection and tracking are specific needs of this demand, as they enable the automation of applications such as retail analysis and video surveillance. Using the tracking-by-detection approach, this work proposes the implementation of a pedestrian tracking system using deep learning object detector with a recursive filtering algorithm for tracking. It also proposes the use of an algorithm for the assignment problem due to the probable presence of multiple pedestrians on the video frame.*

Resumo. *A análise automatizada de vídeo é uma demanda crescente uma vez que a tecnologia permite a criação de dados visuais em taxas crescentes provenientes de diferentes tipos de aplicações. A detecção e rastreamento de pedestres são uma necessidade particular dessa demanda ao permitir automações de aplicações como análise de varejo e vigilância por vídeo. Usando a abordagem de tracking-by-detection, este trabalho propõe a implementação de um sistema de detecção e rastreamento de pedestres usando um detector baseado em técnicas de aprendizado profundo para a detecção e um algoritmo de filtragem recursiva para o rastreamento. Também propõe o uso de um algoritmo para o problema de associação devido à provável presença de múltiplos pedestres no frame de vídeo.*

1. Introduction

1.1. Motivation

The technology has been leading us toward a world in which it is incredibly affordable to have equipment to capture and store visual data. In video surveillance, it is increasingly common to install video cameras in public or private spaces. However, the visual data from these cameras is usually monitored without proper attention or not at all [Dick and Brooks 2003]. New Scientist Magazine once said, “there are too many cameras and too few pairs of eyes to keep track of them” [Hogan 2003]. In retail analysis, the use of camera equipment allows to capture visual information about the customer, but an automated application can go further allowing the creation of better statistics about customer behaviour allowing retailers to make better business decisions. In both applications, the amount of visual data being generated is vast, making the manual analysis of such applications impractical.

In the context of manual video surveillance, usually, the cameras capture the monitored environment and send the images to another room containing small screens where the images are visualized, often these screens contain the video from more than one camera, inside a room that can contain multiple monitors. In this scenario, the human operator must be watching all the videos seeking for some suspect which requires skills that are beyond human capacity, especially the ability to maintain concentration. For example, a study shows that humans watching a single video monitor for more than 20 minutes lose 95% of the ability to maintain concentration enough to distinguish important events [Green 1999]. This only corroborates the demand for automated video analysis.

This problematic context creates the demand for automated video analysis. Furthermore, the constant increase in computational power by CPUs and the parallel processing by GPUs, as well as the advances in technologies provided by researches in the computer science community, only increase the demand. Therefore, the computer vision community is always trying to find new approaches for algorithms be able to extract high-level information from visual data. As a result, it allows applications to automate tasks before performed by the human visual system, such as video surveillance. Computer vision is an interdisciplinary field that makes use of various technologies to reach its goals. Moreover, it has been gaining much attention in recent years due to the potential of applications in various areas of human knowledge, from biological to aerospace.

The visual object tracking (VOT), visual tracking or target tracking is one of the most fundamental processes for an understanding of video content [Jalal and Singh 2012]. It can be defined as the process of keeping track of one or more moving objects in a video stream [Acharya and Ray 2005]. It is time-expansive due to a large amount of data that must be processed, and consequently, it is a complex problem to propose a solution that satisfies both accuracy and speed. Even though several approaches have already been proposed, it remains an open problem in the community.

The VOT can be separated into two broad categories: single object tracking (SOT) and multiple object tracking (MOT). While SOT will be concerned with detecting and tracking only a single object for the duration of the video, MOT will need to detect and track multiple targets that may appear and disappear for the duration of the video. Both groups share almost all the challenges for visual object tracking. However, the last one will have to handle new types of challenges due to occlusion and collision between targets, identifying among multiple objects, and birth/death of targets.

The MOT problem is one of the most challenging areas of research in the computer vision [Guan et al. 2016, Li et al. 2013]. It can be defined as the process of creating a record of the trajectory of each target in time and space in each frame of a sequence of generated images [Fan et al. 2016, Li et al. 2013]. The problem has a long history spanning about 50 years [Mallick et al. 2013], receiving a lot of attention in recent years due to its potential applications in the research community and commercial fields [Fan et al. 2016]. It allows a wide range of applications in various areas of human knowledge such as public safety, air traffic control, autonomous vehicles, statistics generation, robotics, oceanography, astronomy, molecular biology [Guan et al. 2016].

The pedestrian tracking falls under multiple object tracking and its most popular application is video surveillance [Jalal and Singh 2012], due to the fact that it is not

just digitally recording some space in video: it also includes more complex tasks such as extracting information about the movement of targets, and recently even analyzing suspicious behaviors from the watched scene [M. Shah and Shafique 2007]. Though the MOT application could track several targets such as vehicles, airplanes, and even microorganisms, the pedestrian class is the primary tracking target for the video surveillance application. In addition, other pedestrian tracking applications are sports analysis, retail analysis, file/video comprehension, drive assist, autonomous vehicles.

For retail analysis, a pedestrian tracker can be a powerful tool for generating information about customer behavior inside the store. For example, the application in a store could track all pedestrians who frequent a particular space over some time to generate a heat map with the areas most frequented by customers. Additionally, the application could generate data about how long customers are waiting to be serviced in a particular line. As a result, all of these statistical data would allow the retailer to make better decisions to benefit their customers and their business.

The implementation of a pedestrian tracking system is the first step to propose solutions for those applications because it allows knowing where people have been walking giving each pedestrian a unique identifier through the whole video sequence.

The tracking-by-detection approach has been shown in the literature that helps to tackle the inherent object tracking challenges that will be addressed in Section 1.2 [Mekonnen and Lerasle 2018]. In this approach, the system consists of three main components: detection, tracking, and assignment. However, it is in the object detector which plays the main role. That is, the system relies on it to perform the main tasks involved in tracking such as localization, classification and representation of each pedestrian.

Traditionally, the pedestrian trackers have been tackled by using geometric algorithms to correlate objects between frames, for example, the optical flow to analysis moving objects, leaving a great territory for machine learning techniques [Bhooshan and Garg 2017]. Moreover, machine learning techniques have been improving results in several areas of human knowledge, including specific areas of computer vision, from the reconstruction of scenes to the detection of objects. The first approaches of object detectors using machine learning techniques improved the accuracy of detection, but they required a lot of processing, as result their speed were low. New approaches known as single-shot object detectors presented great results in both accuracy and speed as shown by Table 1.

1.2. Challenges

The tracker needs to extract the right information to well describe the target over the whole duration of the video. To be able to achieve that, the information needs to be generic enough to handle the indeed variation of appearance that each pedestrian could have. However, at the same time, it also needs to be precise enough to be able to keep a unique identification to each pedestrian through the frames of the video sequence. This leads to a complex overall challenge since all objects are just projections onto the image domain. Additionally, it is necessary taking into account the amount of information to be processed because a large amount of data that needs to be processed is one of the challenges of tracking.

A video V can be defined as a sequence of images I_i of same resolution where i

is the frame number. Each image is composed by a set of pixels $p_{i,j}$ where i and j are the pixel coordinate on the image plane. Additionally, each pixel p is represented by one or more integer number depending on the color model used. Altogether, the amount of data is related to both image resolution and color model. As an example, an 800×600 image will have 480,000 pixels if the image has only one color channel, but if the RGB color model is being used it will be three times that or 1,440,000 integer numbers per frame. A video with a frame-rate of 30 frames per second can have on average 40 megabytes per second of uncompressed data to be processed by the application.

The majority of the challenges are due to the variations that the pixels composing an object could suffer due to factors such as ambient illumination or occlusions. Even state-of-art visual object trackers cannot handle well in severe occlusions and illumination changes. The pedestrian class offers additional challenges due to the dynamics of the human being. For example, they have a significant internal class variability that can be caused simply by the reason that each pedestrian could be wearing clothes with different color and texture.

The challenges due to the temporal variations of the pixels that represent each target through the video sequence can cause a phenomenon called *clutter* in which two target are very similar between themselves or between it and the background. This phenomenon can cause two symptoms that affect the operation of the tracker. The first symptom occurs when the tracker can no longer find the targets that are present in the video frame. The second occurs when it merges two different targets into a single identification. The reasons that temporal variations of the pixels occur are varied. The main ones are due to the indeed dynamics of the targets, ambient illumination and occlusions.

The dynamics of the target such as changes in pose modifies the pixels that are being projected into the image. Types of pose changes include deformation, rotation, and translation. An example of deformation could occur when a pedestrian crouches to pick up some object that is dropped on the ground. Because of the crouched pose, the target has no more the expected proportion of a pedestrian or very different pixel values that used to be in previous frames.

The ambient lighting can interfere with how objects are represented in the image. A sunny day or a cloudy day in an external environment, the intensity and color of artificial lights in an indoor environment, and even the light color temperature can change the appearance of the targets. Shadows and reflections also play a role. For example, a problem that a reflection can cause is in the scenario where the reflection of a pedestrian appears in some mirror. This can lead to some state where the tracker detects a new object from the reflection.

The occlusions of a target are problematic because they can not only alter the appearance of targets, called partial occlusion, but also cause the object to no longer be detected for a considerable period, called total occlusion. If the tracker uses some filtering estimator, theoretically it can maintain the trajectory for a period, but it is subject to the target *drifting* problem if the target moves abruptly under occlusion. Another challenge of occlusion is the lack of context of the movement. For example, a target that fails to appear for some frames needs to be finalized if he does not come back to the scene or continue with its identification if the occlusion is temporary. It could be a challenge for the tracker

to decide whether to finalize target detection or to maintain identification in the hope that it will reappear.

The camera is the hardware that creates input data to feed the tracker, so its characteristics will shape the approach that the tracker will use to perform its function. The position, viewpoint, static/moving, image color model, video framerate, noise, and calibration parameters play an important step when dealing with tracking of objects.

A calibrated camera can help to optimize the tracking since calibration parameters such as the main point, the focal length and the distortion coefficients are known. The known parameters could used to do some image processing, such as measuring the size of an object, determining the location of a camera in a scene, correcting distortions of lens, and get better estimate on target movement. Additionally, the camera position can change the amount of information per target. For example, an external surveillance camera placed at the top of a building will produce images in which the targets will be projected as points due to the large distance between the camera and the targets. On the other hand, a surveillance camera inside a convenience store will be closer to the targets, being able to capture more information about them such as their shape, color or texture.

All mentioned problems can be reduced to perform specific tests. For example, collecting only videos that do not contain target occlusions or only videos generated from static cameras may be useful for generating specific evaluation analysis.

1.3. Main Goals

This work proposes the development and implementation of a pedestrian tracking algorithm using the tracking-by-detection approach. The main objective is to evaluate the use of state-of-art deep learning object detectors coupled with classical filtering algorithm for the tracking of pedestrian from a video sequence that is in some way similar with what could be found in video surveillance or retail analyses. The secondary goals are enumerated as the following.

i) Study the use of state-of-art deep learning object detectors;

ii) Study filtering approaches to improve the estimation of the tracks;

iii) For MOT, study and develop methods for target association.

The rest of this paper is organized as follows. Section 2 gives a brief background in visual object tracking architectures and components. It also presents some of the related work available in the literature. Section 3 presents more details about the work proposal of this paper. Lastly, section 4 concludes the paper.

2. Background and Related Work

2.1. Pedestrian Tracking

The goal of an application that tracks pedestrian is to be able to give a unique identification to each detected pedestrian on the video to create a trace from the first moment it is visible (target-birth) to the last frame (target-death). It is a particular case of MOT, and a vast number of approaches have been proposed in the literature. This section will present a

brief overview of some conventional approaches already used in pedestrian tracking. Next section will focus only on the tracking-by-detection approach, the focus of this work.

The tracker applications are complicated systems made up of several separate components [Wang et al. 2015]. According to [Maggio and Cavallaro 2011], an object tracker architecture can be divided into five main components: extraction, representation, propagation, birth/death management and metadata extraction. In another taxonomy presented in [Fiaz et al. 2018], the tracking process is divided into four components: initialization, appearance modeling, motion estimation, and target positioning. This work will focus on the tracking-by-detection approach where it is common to split the application into three main components: detection, tracking and association module [Mekonnen and Lerasle 2018].

The features are the useful information the tracker will extract to try to find and represent the target. In the sense of complexity, there are three classifications for features: low, middle and high-level features. The low-level features are the most basic ones such as position, size and color. The mid-level features are more complex and normally extracted from a region of pixels from the image, may using low-level features as well. Example of mid-level features could be image edges or orientations. Finally, the high-level features are the most complex ones and could represent a whole object.

In [Mekonnen and Lerasle 2018], it was presented a short history of more complex features used in the context of pedestrian tracking. The haar-like features were one of the first successes in pedestrian detection and it was inspired by the haar wavelets [Viola and Jones 2001]. Later, the histogram of oriented gradients (HOG) improved the state-of-art and it was followed by using more than one features in a technique called *feature pooling*. The work [Dalal and Triggs 2005] proposed the use of HOG to human detection. Next significant improvement towards better pedestrian detection was presented with deformable parts model (DPM), which models an object as a set of parts conditioned in the spatial arrangement. Lastly, the channel features which an intermediate layer filtering low-level features in combination with a boosted decision forest which obtained top performance on the challenging Caltech and KITTI datasets [Zhang et al. 2015].

The HOG+SVM detector is a traditional detector and it was proposed in [Dalal and Triggs 2005] as a better descriptor for human detection. This approach originally proposed for the detection of people, however, future articles presented results for objects in general [Siji Joseph 2017]. It calculates the histogram of the gradient orientation on an equally spaced grid, generating a vector gradient, and uses a linear Support Vector Machine (SVM) as a classifier. It is immune to deformation as it stores gradient magnitudes and normalization is realized to try to maintain illumination invariant. However, the [Mekonnen and Lerasle 2018] says it performs poor in recent benchmarks when compared with new approaches. The related work [Sugano et al. 2010] explores the use of HOG for optimized pedestrian tracking using the GPU to perform parallelism of processing.

The sliding window is a classical tracking approach, but it has a high computation cost since it relies on exhaustive search over the image, applying some similarity function between the multiple patches that are being extracted from the image and a predefined template of the target. The comparison can be done with any image similarity algorithm,

usually proposed for the image retrieval problem, such as normalized cross-correlation. The candidate with the closest similarity to the original template is chosen to be the target of the current frame. Although this example is very simplified, this approach, with proper optimization, remains one of the very important for object detection. However, this approach may not be good for pedestrian tracking, since it will be hard to create a universal model for all possible pedestrians that could appear in the video.

Another approach is to use information about the pixel's movement to detect and track targets. The motion is a powerful information that can be extracted in videos when two or more consecutive frames are compared to detect which areas have been moved. The optical flow is a vector field of apparent motion that can be used to detect moving targets, such as pedestrians. The features that are moving together could be treated as belonging to the same object. This may cause a problem in videos with a high density of moving objects since they are too close to be detected as a single object. Additionally, another downside of this approach is that issues can happen if the target stop to move for some time, also called the *sleeping object* problem. Related work of this approach can be found at [Hariyono J. 2014] where optical flow and HOG was used to detect pedestrian using motion information to optimize the time-consuming sliding window approach. In the work of [Viola et al. 2003], the information about motion over two consecutive frames was used to detect pedestrians. They also presented an efficient representation of image motion.

The background subtraction is a key technique for automatic video analysis in the domain of video surveillance [Mekonnen and Lerasle 2018]. The basic concept of background subtraction is to create an initial template to describe the background before the video analysis start. Next, it uses this model to subtract the background in each video frame, so only the foreground will be visible allowing the detection of targets. In [Kumar and Yadav 2016], it was used this approach to extract and track objects in complex environments. It also uses an adapting Kalman filter to track the targets. Additionally, the work [Zhang and Ding 2012] uses adaptive background subtraction to track moving objects.

2.2. Tracking-by-Detection

The tracking-by-detection approaches rely on an object detector to start, update, restart, or terminate a tracker [Elie Moussy 2015]. It is not necessary to have any information from previous frames since each frame is analyzed separated, applying an object detector to locate and classify the targets. The tracking-by-detection approach has been shown in the literature that helps to tackle the inherent object tracking challenges [Mekonnen and Lerasle 2018]. It can be divided into three main components: detection, tracking and assignment. A basic framework can be seen in Figure 1.

The detector component performs the localization and classification of the targets, creating a model that represents the object such as a bounding-box containing information about location, size and class-score, and then delivering it to the assignment component.

In each frame, once the detector finishes the detection of all the target, the association algorithm take place to keep the same identification for each target. After, the filtering process is started, and the tracker takes place to create a trace of each target through the frames of the video sequence.

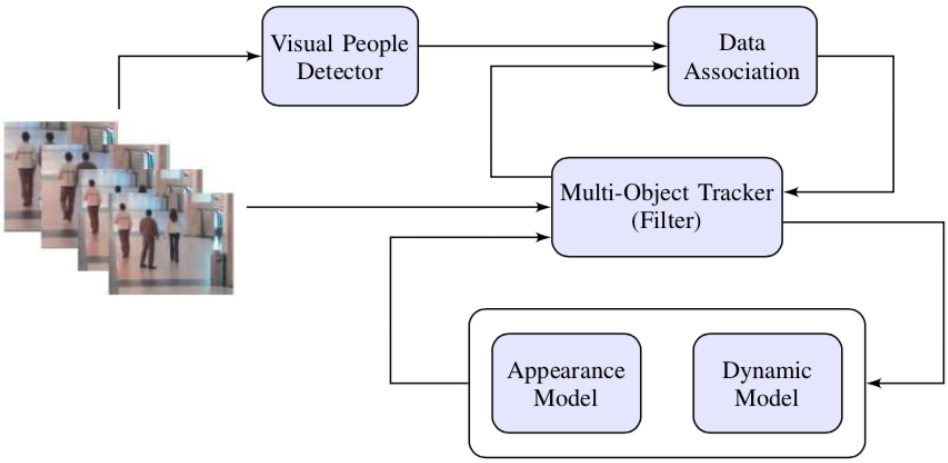


Figure 1. A basic framework of a tracking-by-detection approach from [Mekonnen and Lerasle 2018]

2.2.1. Detection Component

In the tracking-by-detection approach, the detection of objects is the crucial step. The detection of objects by itself is a challenging computer vision problem. A large number of researches have already been done in the area of object detection in the last two decades involving several areas such as image processing, linear algebra, statistical/probability, machine learning [Prasad 2012].

Thus, there are several approaches to address the problem such as the classical ones mentioned in the previous section, and also more advanced ones like the aggregated channel features (ACF) and locally decorrelated channel features (LDCF).

Recently, with advances in technology in machine learning techniques, region-based convolutional neural networks (RCNN) have been used to detect objects with interesting accuracy results [Mekonnen and Lerasle 2018]. The approaches presented good results in accuracy, but not in speed because of one way or another they still doing an extensive search over the image to detect objects. In other words, they are an optimized sliding window approach which provides region of proposals followed by a classifier to classify these proposals.

Examples of the use of CNN for object detection were presented with OverFeat published in 2003 [Sermanet et al. 2013]. Shortly after the publication of OverFeat, a method using regions was proposed in which it defined around 2000 potential regions of interest to apply a classifier in each of them [Girshick et al. 2013]. The Fast RCNN [Girshick 2015] and Faster RCNN [Ren et al. 2015] presented an optimization by using a neural network to propose objects of interest to avoid the extensive search of applying the classifier in each patch thousands of times.

A time later, new approaches started to propose a combination of those two tasks, i.e. the creation of regions of proposals and the classifier, into one single network. They presented techniques to analysis the whole image once instead of a sliding window or selective region of interest allowing much better results in speed. Those type of network

will be the focus for the pedestrian detection problem proposed by this work.

As mentioned earlier, the first approaches using CNN used to apply a classifier to several regions of interest extracted from the image. It has a cost of speed since it is necessary to send a huge number of regions to a classifier with the goal of finding some target. An alternative to this approach is the known as single-shot object detector where the entire image is passed only once to the classifier. Examples of these types of detectors are the single-shot multibox detector (SSD), the you-only-look-once (YOLO), the region-based fully convolutional network (R-FCN), the feature pyramid networks (FPN) and the focal loss (FL).

An example of a popular algorithm that uses this method is YOLO, which is a system proposed in 2015 for real-time object detection [Redmon et al. 2015]. The author proposed a new approach using a single CNN to locate and classify the objects in one evaluation instead of the previous pipeline which performed the classification on many regions of interest. The YOLO passes the whole image through the neural network only once, and it outputs class probabilities and bounding boxes for predictions. Using this approach, the system has the best speed when compared with other methods, as shown by Table 1, it also has very good accuracy.

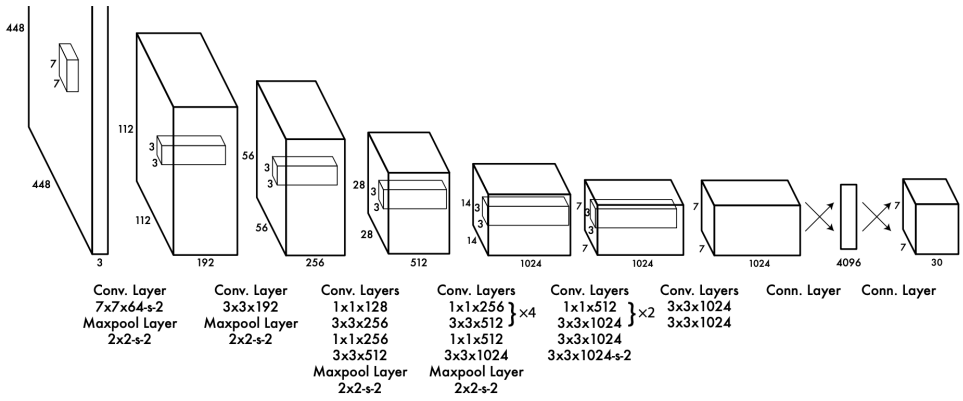


Figure 2. Yolo framework [Redmon et al. 2015]

The YOLO framework follows a standard convolutional neural network, with convolutional and pooling layers and connected in the end by two fully connected layers. The system splits the input image in a grid of $S \times S$ cells. Each cell will be responsible for detecting one object which geometric center falls on it. The output of the system is a bounding-box and a classification score. Each bounding box contains four variables that localize the object into the image by giving the position (x, y) and the size (w, h) of the detected object, also the class-score of the object inside the box which represent the probability of the detected object be a specific class label.

2.2.2. Tracking Component

In the tracking-by-detection approach, the tracker is the process where the state of the detected object is propagated through the frames of the video sequence. The use of filtering algorithms is widely used because they allow the target state in the current frame to be

Table 1. Comparison of some object detectors

Algorithm	Pascal 2007 mAP	Speed
DPM v5	33.7	.07 FPS
R-CNN	66.0	.05 FPS
Fast R-CNN	70.0	.5 FPS
Faster R-CNN	73.2	7 FPS
YOLO	69.0	45 FPS

optimally estimated from the information of the previous frame and the information of the current frame as well. The filter has this name in the sense that it is discretized in the time domain. In VOT, the use of estimators is interesting because the state of the object can be estimated even if the detector fails due to some occlusion event.

Following the taxonomy provided by [Mekonnen and Lerasle 2018], the tracker component can be divided into two classes: the purely probabilistic based on the Monte Carlo approach and the stochastic/deterministic ones. They also can be organized in decentralized and centralized. In the first one, each target receives a unique instance of the filtering algorithm while the last one does not assume that. The trackers can also be classified between online and offline. The online trackers perform the tracking as the data is received, while the offline tracker analyzes the entire video before performing the tracking.

There are several algorithms in the literature such as the Kalman filter (KF), the extended Kalman filter (EKF), the particle filter (PF), the decentralized particle filter (DPF), the reversible jump Markov chain Monte Carlo (RJMCMC), the simple online and real-time tracker (SORT), the Markov decision process (MDP), the kernelized Correlation filter (KFC), and the Rao-Blackwellised particle filter, to list some. The survey provided in [Mekonnen and Lerasle 2018] is a good source of information.

The filters accomplished the optimal estimation by an iterative process of prediction and correction. The predictions step is made using a motion model to estimate the current frame state from the previous frame. The correction step is calculated based on the observation model in the way that minimizes the error of the estimated parameters in an optimal way. Three motion models are commonly found in MOT: random walk model, linear autoregressive model, and non-linear models [Mekonnen and Lerasle 2018].

For example, the Kalman filter is a single-hypothesis optimal linear estimator that uses a series of measurement data that may contain a statistical Gaussian noise to estimate the object state statistically minimizing the mean square error by getting a joint probability distribution over the variables in each frame. In other words, the algorithm recursively estimates the state of the target x_k from the previous frame state x_{k-1} where k is a time interval such as millisecond or frame number. [Kalman 1960] proposed this recursive technique for the discrete-data linear filtering problem.

Usually, it is used in systems that have a linear motion and noise model with a Gaussian distribution. For non-linear systems, other methods may work better such as the Extended Kalman filters or particle filters. The Kalman filter has been established as the optimal solution for many tracking problems [Lacey]. When used with the Hungarian algorithm, they are two extremely efficient algorithms to handle the mo-

tion prediction and data association components of the tracking problem respectively [Mekonnen and Lerasle 2018]. Related work using Kalman filter in the context of pedestrian tracking can be seen at the work of [Mittal et al. 2012] where the Kalman filter is used together with DPM detector to track pedestrian. Also, [Beymer and Konolige 1999] made use of a Kalman filter based tracker and a template based person detector to track the targets.

The Kalman filter estimates the state $\vec{x} \in \Re^n$ of a system assuming the linear stochastic equation (1). The measurement $\vec{z} \in \Re^n$ at time k is a linear combination between the state values and the measurement noise v given in equation (2).

$$\vec{x}_k = \mathbf{A}\vec{x}_{k-1} + \mathbf{B}\vec{u}_{k-1} + w_{k-1} \tag{1}$$

$$\vec{z}_k = \mathbf{H}_k\vec{x}_k + v_k \tag{2}$$

The matrix $\mathbf{A}$ is the transition model and relates the state from the previous frame to the state of the current frame. The matrix $\mathbf{B}$ relates the control input model applied to the control vector $\vec{u}_k$. The matrix $\mathbf{H}$ is the observation model and relates the space state with the measured space. The variables w is the prediction noise and v is the measurement noise, they represent a random noise assuming they have a Gaussian distribution, where $\mathbf{Q}$ is the process noise covariance and $\mathbf{R}$ is the measurement noise covariance.

$$p(w) \sim N(0, \mathbf{Q}) \tag{3}$$

$$p(v) \sim N(0, \mathbf{R}) \tag{4}$$

The first practical step is to define the state variables to be estimated over time. For a pedestrian tracker that will receive the output directly from the object detector, the state of each pedestrian can be defined as containing at least four variables: x, y, v_x and v_y. Respectively the horizontal and vertical position, and the horizontal and vertical velocity.

The prediction of the state $\hat{\mathbf{x}}_k$ from $\vec{x}_{k-1}$ and the covariance $\hat{\mathbf{P}}_k$ from $\mathbf{P}_{k-1}$ a priori is given by

$$\vec{x}_k = \mathbf{A}\vec{x}_{k-1} + \mathbf{B}\vec{u}_{k-1} \tag{5}$$

$$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \tag{6}$$

Where matrix $\mathbf{A}$ and $\mathbf{B}$ are from equation (1) and $\mathbf{Q}$ from equation (3).

The measurement update is given by

$$K_k = \mathbf{P}_k\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}_k\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1} \tag{7}$$

$$\vec{x}_k = \vec{x}_k + K_k(\vec{z}_k - \mathbf{H}\vec{x}_k) \tag{8}$$

$$\mathbf{P}_k = (\mathbf{I} - K_k\mathbf{H})\mathbf{P}_k \tag{9}$$

Where equation (7) calculates the Kalman gain K. The equation (8) corrects the predicted state. And the equation (9) updates the error covariance. The $\mathbf{I}$ is the identity matrix.

After each prediction and correction steps, the process is repeated to recursively estimate the optimal estate. More deep information about the filter can be found at [Sorenson 1970, May , Welch and Bishop 2006].

2.2.3. Association Component

In MOT, the problem of associating targets between frames needs to be solved. The goal is to match the detection of the current frame with the same detected pedestrian of the previous frame. In other words, once a pedestrian is detected and given a numeric identification, it must continue with it throughout the entire lifespan. The Hungarian algorithm is commonly used for the association problem [Mekonnen and Lerasle 2018], although a basic greedy algorithm also could solve it.

The Hungarian algorithm solves the assignment problem given a cost matrix that defines the cost between target from two consecutive frames. For example, a cost matrix $S_{i,j}$, where j represents the jth target from the previous frame and i the ith target on the actual frame, that contains metrics such as distance and scale similarity can be modeled as following

$$S_{i,j} = \alpha_d \begin{bmatrix} x_i - x_j \\ y_i - y_j \end{bmatrix} + \alpha_s \left[s_i - s_j \right] \tag{10}$$

where α_d and α_s are adjustable parameters for image distance and scale, respectively. The x and y are the geometric center of the objects. And s could represent the area of the detected target.

2.2.4. Evaluation

The evaluation could be made using public dataset and standards metrics. The time per frame of the tracking processing is a basic metric for speed. For object detection, there are two important metrics: precision and recall. They are defined according to equations (11) and (12) [Mekonnen and Lerasle 2018].

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

TP stands for the number of true positives, FP for false positives and FN for false negatives. A false positive happens when the module detects an object that is not in the image. A false negative when it does not detect an object that is in the image. And true positive when the module correctly detects the object. More detail about the calculation is presented in the work [Dollar et al. 2012].

In MOT, the two most important metrics were presented by CLEAR-MOT metrics as Multi-Object Tracking Accuracy (MOTA) and Multi-Object Tracking Precision (MOTP) [Kasturi et al. 2009, Mekonnen and Lerasle 2018], equations (13) and (14), respectively.

$$MOTA = 1 - (F_p + F_n + Id_{sw}) \quad (13)$$

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t C_t} \quad (14)$$

From equation (13), the variable $F_p = \sum_t \frac{FP_t}{g_t}$ is the total number of false positives, $F_n = \sum_t \frac{FN_t}{g_t}$ the total number of false negatives, and $Id_{sw} = \sum_t \frac{Id_{sw,t}}{g_t}$ the total number of identification switches. The g_t is the number of ground truth objects from the whole dataset. From equation (14), the d_t^i is the Euclidean distance between the matched ground truth location and the tracker target location. The C_t is the total number of matches made.

The MOTA accounts for all errors such as missed target or false positives. The MOTP try to account for the ability of the tracker to estimate precise positions, independent of its results from the detection step, keeping consistent trajectories. Detailed information is presented in [Bernardin and Stiefelhagen 2008].

3. Work Proposal

As mentioned in Section 1.3, this work proposes the implementation of a pedestrian tracking using the tracking-by-detection approach where the main objective is to evaluate the use of state-of-art deep learning object detectors coupled with classical filtering algorithm for the tracking of pedestrian from a video sequence that is in some way similar with was could be found in video surveillance or retail analyses.

The video sequences used will first come from the standardized public data-set for the SOT/MOT problem. Some personal video may be created for personal test using a cellphone. Due to the complexity of the tracker implementation due to the variety of challenges to achieve accuracy and speed, the use of video sequences with restrictions may be applied, for example, the number of pedestrian and occlusion events.

The software output will be a sequence of locations for each target in the video sequence.

Table 2. Schedule of activities

Activity	Jan	Feb	Mar	Apr	May	Jun
Detection component	X	X				
Filtering component		X	X			
Assignment component		X	X			
Integration of components		X	X	X	X	
Tests and Results			X	X	X	
Monograph writing		X	X	X	X	

4. Conclusion

This article presented a brief description of approaches to track pedestrians, including the tracking-by-detection approach, in which the system consists of three main and independent components to detect, track and assign, respectively.

The proposed work will be useful to evaluate the use a pedestrian tracking using the tracking-by-detection approach where the main objective is to evaluate the use of state-of-art deep learning object detectors coupled with classical filtering algorithm for

the tracking of pedestrian from video sequence that is in some way similar with what could be found in video surveillance or retail analyses.

5. References

Acharya, T. and Ray, A. K. (2005). *Image Processing - Principles and Applications*. Wiley-Interscience, New York, NY, USA.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309.

Beymer, . and Konolige, K. (1999). Real-time tracking of multiple people using continuous detection. *IEEE International Conference on Computer Vision*.

Bhooshan, S. and Garg, A. (2017). Multi-object tracking (mot) with deep learning.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

Dick, A. R. and Brooks, M. J. (2003). *Issues in automated visual surveillance*.

Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761.

Elie Moussy, Alhayat Ali Mekonnen, G. M. F. L. (2015). Tracking-by-detection' approaches. *IEEE International Conference on Advanced Video*.

Fan, L., Wang, Z., Cail, B., Tao, C., Zhang, Z., Wang, Y., Li, S., Huang, F., Fu, S., and Zhang, F. (2016). A survey on multiple object tracking algorithm. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 1855–1862.

Fiaz, M., Mahmood, A., and Jung, S. K. (2018). Tracking noisy targets: A review of recent object tracking approaches. *CoRR*, abs/1802.03098.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Green, M. W. (1999). The appropriate and effective use of security technologies in u.s. schools. a guide for schools and law enforcement agencies.

Guan, H., Xue, X., and Zhiyong, A. (2016). Online video tracking using collaborative convolutional networks. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

Hariyono J., Hoang VD., J. K. (2014). Motion segmentation using optical flow for pedestrian detection from moving vehicle. *Computational Collective Intelligence. Technologies and Applications*, 8733.

Hogan, J. (2003). Smart software linked to cctv can spot dubious behaviour. `https://www.newscientist.com/article/`

dn3918-smart-software-linked-to-cctv-can-spot-dubious-behaviour/. Accessed: 2018-11-20.

Jalal, A. and Singh, V. (2012). The state-of-the-art in visual object tracking. *Informatica (Slovenia)*, 36:227–248.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., and Zhang, J. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336.

Kumar, S. and Yadav, J. S. (2016). Video object extraction and its tracking using background subtraction in complex environments. *Perspectives in Science*, 8:317 – 322. Recent Trends in Engineering and Material Sciences.

Lacey, T. Tutorial: The kalman filter. http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf. Accessed: 2018-11-15.

Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A. R., and van den Hengel, A. (2013). A survey of appearance models in visual object tracking. *CoRR*, abs/1303.4803.

M. Shah, O. J. and Shafique, K. (2007). Automated visual surveillance in realistic scenarios. *IEEE MultiMedia*, 14(1):30–39.

Maggio, D. E. and Cavallaro, D. A. (2011). *Video Tracking: Theory and Practice*. Wiley Publishing, 1st edition.

Mallick, M., Vo, B., Kirubarajan, T., and Arulampalam, S. (2013). Introduction to the issue on multitarget tracking. *IEEE Journal of Selected Topics in Signal Processing*, 7(3):373–375.

Mekonnen, A. A. and Lerasle, F. (2018). Comparative evaluations of selected tracking-by-detection approaches. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1.

Mittal, S., Prasad, T., Saurabh, S., Fan, X., and Shin, H. (2012). Pedestrian detection and tracking using deformable part models and kalman filtering. In *2012 International SoC Design Conference (ISOCC)*, pages 324–327.

Prasad, D. (2012). Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP)*, 6:441.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

Siji Joseph, A. P. (2017). Object tracking using hog and svm. *International Journal of Engineering Trends and Technology (IJETT)*, 48.

Sorenson, H. W. (1970). Least-squares estimation: from gauss to kalman. *IEEE Spectrum*, 7(7):63–68.

Sugano, H., Miyamoto, R., and Nakamura, Y. (2010). Optimized parallel implementation of pedestrian tracking using hog features on gpu. In *6th Conference on Ph.D. Research in Microelectronics Electronics*, pages 1–4.

Viola, Jones, and Snow (2003). Detecting pedestrians using patterns of motion and appearance. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 734–741 vol.2.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. pages 511–518.

Wang, N., Shi, J., Yeung, D., and Jia, J. (2015). Understanding and diagnosing visual tracking systems. *CoRR*, abs/1504.06055.

Welch, G. and Bishop, G. (2006). An introduction to the kalman filter.

Zhang, R. and Ding, J. (2012). Object tracking and detecting based on adaptive background subtraction. *International Workshop on Information and Electronics Engineering*, pages 1351–1355.

Zhang, S., Benenson, R., and Schiele, B. (2015). Filtered channel features for pedestrian detection. *CoRR*, abs/1501.05759.