TÉCNICO LISBOA

Vehicle tracking in urban environment

Sofia Teodoro Bebiano

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. João Paulo Salgado Arriscado Costeira
Dr. Zita Alexandra Magalhães Marinho

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. João Paulo Salgado Arriscado Costeira
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

February 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my mom Mena and my siblings for encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank Sabri for her understanding and support throughout all.

I would also like to acknowledge my dissertation supervisors Prof. João Paulo Salgado Arriscado Costeira and Zita Alexandra Magalhães Marinho for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. AA I thank you.

To each and every one of you – Thank you.

Abstract

The main purpose of this work is to develop a vehicle tracking system based on video images. The developed system uses a webcamera and the objects, in this case vehicles, are detected using the You Only Look Once (YOLO) system. The matching between detections is done with a Kanade-Lucas-Tomasi (KLT) feature tracker. Which by using corner point features from the initial detections and tracking them to the following frames, it is able to match detections across frames. Once matched, the projection of the objects in the camera plane onto the ground plane is calculated and presented. The main challenges of this task are object occlusion and object association across frames. The approach for this was the integration of the feature tracking method into the object detection method. This way when the detector fails (occlusion for example) it is still possible to track features and keep an object track continuous. The feature tracking also aids with object association since each feature will be associated with an object. The system was tested and evaluated in a real traffic scenario of a crossroad. It performs well in regular traffic, while being able to keep tracks in situations with small and medium occlusion. Of the completed tracks, 24% of instances had no YOLO detections. This work shows how using both a static object detection method and a dynamic feature tracking method results in a more robust multi object tracking system.

Keywords

YOLOv3; Kanade-Lucas-Tomasi; Homography; Multiple vehicle Tracking; Object Detection on Image.

Resumo

O objetivo principal deste trabalho é desenvolver um sistema de seguimento de veículos baseado em imagens de video. O sistema desenvolvido utiliza uma *webcamera* e os objetos, neste caso veículos, são detetados utlizando o sistema You Only Look Once (YOLO). A correspondência entre deteções é feita através de um *tracker* Kanade-Lucas-Tomasi (KLT), que fazendo o seguimento de *features* da deteção inicial é capaz de corresponder deteções através das *frames*. Assim que existe correspondência, a projeção dos objetos do plano da camâra para o plano do chão é calculada e apresentada. Os principais desafios deste trabalho são a oclusão e associação de objetos ao longo das textitframes. Para lidar com estes desafios um método de seguimento de features foi integrado com a deteção de objetos. Assim quando o detetor falha, por exemplo quando existe oclusão, continua a ser possível seguir as *features* e obter um seguimento contínuo do objeto. O seguimento de *features* auxilia também a associação de objetos pois cada *feature* estará associada a um objeto. O sistema foi testado e avaliado num caso de trânsito real. Apresenta um bom desempenho em situações de trânsito habituais, conseguindo manter o seguimento de vários veículos em situações com pequena e média oclusão. Para os *tracks* completos, 24% das instâncias não possuía detecções do YOLO. Este trabalho mostra que usando um método estático de deteção de objeto em conjunto com um método dinâmico de *tracking* de features resulta num sistema mais robusto de *tracking* de múltiplos objetos.

Palavras Chave

YOLOv3; Kanade-Lucas-Tomasi; Homografia; Tracking de múltiplos veiculos; Detecção de objetos em imagem;

Contents

List of Figures

List of Tables

Acronyms

ITS	Intelligent Traffic Systems
DBT	Detection Based Tracking
DFT	Detection Free Tracking
MOT	Multiple Object Tracking
VOT	Visual Object Tracking
SIFT	Scale Invariant Features Transformation
SURF	Speeded Up Robust Features
FAST	Features from the Accelerated Segment Test
MSER	Maximum Stable Extremal Regions
HoG	Histogram of the Gradients
SVM	Support Vector Machine
CNN	Convolutional Neural Network
DPM	Deformable Part Model
GMM	Gaussian Mixture Model
DCNN	Deep Convolution Neural Network
RCNN	Regions with CNN
RoI	Region of Interest
SPPNet	Spatial Pyramid Pooling Network
SSD	Single-Shot-Multibox-Detector
YOLO	You Only Look Once
RP	Region Proposal
LK	Lucas-Kanade
KLT	Kanade-Lucas-Tomasi

CAMShift	Continuously Adaptive Mean Shift
fps	frames per second
DoF	Degrees of Freedom
m/s	metres per second
km/h	kilometres per hour

1

Introduction

Contents

1.1 Motivation

Images are extremely rich sources of information and coupled with the widespread availability of digital cameras is making image based technologies a major source to monitor and model human activity. Thus, being able to detect and follow objects is an indispensable resource to recognise different patterns and trends in our world. The knowledge provided by the understanding of these patterns becomes particularly relevant when it comes to the creation of models that are used to analyse and predict future actions and distinguish typical behaviours from atypical ones. Researchers study these patterns in many different fields, for a variety of purposes. In zoology, wild animal behaviour is studied through the analysis of free-ranging animals' trajectories in order to uphold and preserve ecosystems. Meteorologists also use trajectory data and complex models to predict the weather.

The work developed in this thesis applies a set of image processing techniques in the area of human mobility, a focus of extensive study, with various purposes such as crowd management, public space design, intelligent environments and visual surveillance.

Video tracking is a broad subject with many applications. In particular, surveillance is useful in real world applications. To name a few: monitoring the behaviour of human drivers to improve autonomous vehicle development; predicting ship and plane anomalies by analysing their past trajectories; and improving security by detecting anomalous behaviours in crowded environments.

In this dissertation some of the steps necessary for tracking objects are shown, specifically vehicles (cars, buses) in an urban setting. As mentioned above, these procedures have already been subject to extensive research. A further analysis will be performed, particularly on integrating two techniques to improve vehicle tracking systems, since it can be of great help with traffic management, city planning and development which, in a constantly changing and rapidly growing society, is an increasingly relevant topic.

1.2 Problem statement

This work deals with complex tracking of objects in video, particularly in scenes with multiple moving objects. Tracking objects involves detecting the intended target and following it throughout the various frames of the video providing valuable data for further analysis and inference.

While this might seem a simple task well covered by the state of the art, it involves several challenges which increase when dealing with multiple objects and demanding scenes. One of the issues worth mentioning is the frequent failure of current detectors under severe occlusion of the objects being tracked. Another issue, that comes from tracking multiple objects, is the association of detections from one frame to the other. This thesis attempts to solve these by implementing feature tracking alongside object detection. This approach will allow for features of an object to continue to be tracked even if

the detector misses, resulting in a continuous tracking of the object. Feature tracking also aids with object association since a feature will be associated to an object and thus making the match between detections simpler. Furthermore, one has to consider the sudden appearance of desirable targets in the frame. The approach in this case is to continuously use the object detection algorithm. In this thesis a combination of object detection techniques with feature tracking techniques is proposed, in order to achieve a multi-vehicle tracking system in a real city traffic monitoring scenario. The combination of a static (detection of the objects) and a dynamic (tracking features) method results in a more robust system where the former method's weak points will be remedied by the latter's and vice versa. In particular, it will be shown that this method overcomes detection errors in a few simple but very frequent situations.

1.3 Thesis outline

This thesis is organized as follows: In chapter 2, the state of the art for object tracking and its components, detection and data association, is presented. In chapter 3, a multi-vehicle tracking system merging object detection with point tracking is proposed and the methods used are explained in detail. The chapter 4 displays the results obtained. Lastly, the derived conclusions are presented in chapter 5.

2

Background and Theory of Object Tracking

Contents

Object detection and tracking represent one of the main challenges in the field of image processing. Therefore, only those most relevant to this work will be mentioned and referred, namely traffic monitoring systems, automatic accident detectors and vehicle counting systems, which rely heavily on image processing. Many methods have been proposed to meet these challenges of image processing, to name two: background subtraction methods or more complex methods that are based on movement estimation techniques. Furthermore, factors such as occluded vehicles, obstacles or changing atmospheric conditions can influence and affect vehicle tracking.

2.1 Tracking

Vehicle tracking can be considered a research hotspot in the field of image processing, given its importance in the development of intelligent transport and recognition of traffic systems. For instance, the estimation of the number of vehicles in a traffic video sequence is a substantial component of Intelligent Traffic Systems (ITS). These estimations supply valuable traffic flow information that can subsequently be used in traffic management and the development of dynamic traffic lights.

Tracking can be understood as the localisation of one or more moving objects over time. It is important to distinguish object tracking from object detection since the latter is a prerequisite of the former. In other words, in object detection, an object of interest is localised in one single frame, while object tracking associates the detection of the object of interest throughout several frames. This means that the accurate detection of a moving object is the necessary condition for a tracking system.

There are currently two main object tracking frameworks: Detection Based Tracking (DBT) and Detection Free Tracking (DFT) [1], as shown in fig. 2.1. DFT needs to manually initialize the tracking target, so it is only applicable when tracking a specified target. It is not able to automatically detect and track a new target that appears in the monitoring process. DBT integrates detection and tracking and can automatically detect the emergence of new targets or the disappearance of existing targets. Thus, DBT is capable of meeting the actual requirements of the random disappearance of targets or the dynamic change of targets in the monitoring scene.

Since fixed cameras will be used to monitor scenes that are changing dynamically in specific areas, this work is carried out based on the DBT framework. The detection and tracking of vehicles are the main focus and include various steps. Firstly, to initiate the tracking step, one must verify and locate the presence of a specific object in an image sequence. Secondly, during the object tracking phase, spatial and temporal changes were monitored, including its presence and position. This took place by solving a temporal correlation problem. This problem involves matching the target area in successive frames of a sequence of images that were taken at closely spaced time intervals. These two processes, detection and tracking, are closely linked to each other [2]. Tracking usually begins with the detection of an object

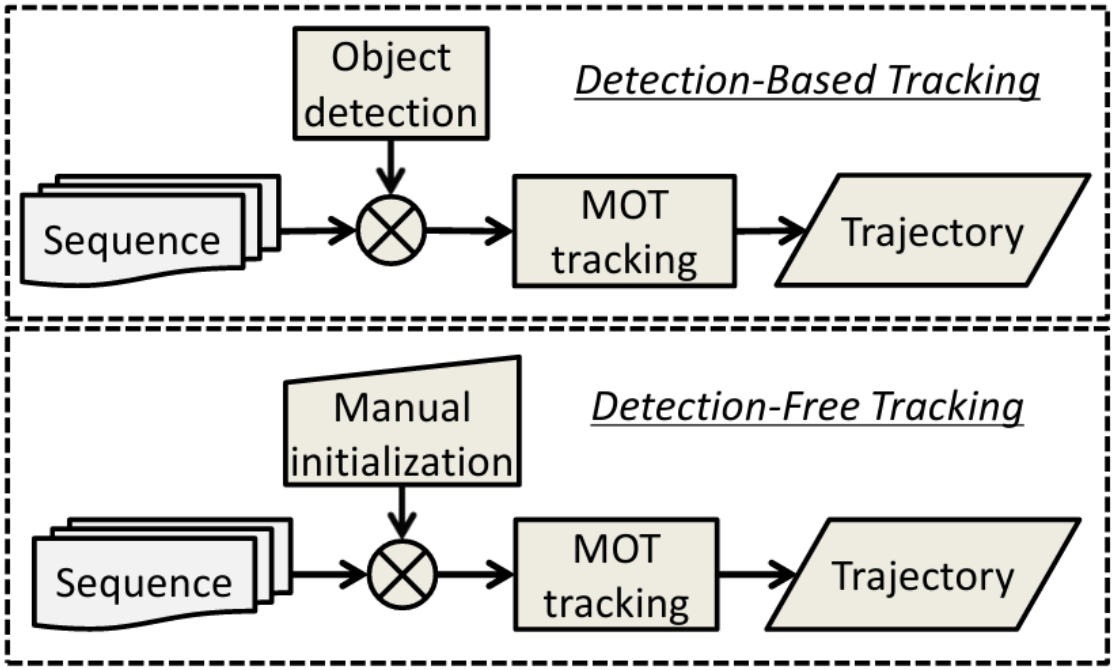


Figure 2.1: Procedure flow of two main object tracking frameworks. Top: DBT, bottom: DFT. Source: [1].

while the repeated detection of an object in a sequence of images is frequently required to verify tracking as it helps to adjust confidence in the tracking phase.

Concerning the quantity of tracked objects in traffic sequences, vehicle tracking can be divided into two main fields: single-vehicle tracking and multiple vehicle tracking.

Traditionally, Multiple Object Tracking (MOT) algorithms have been customised for scenarios with multiple distant objects far from the sensor and each other. Some examples of settings where MOT algorithms are of great relevance are people in a crowd [3], players in sports, and in this case vehicles in traffic. MOT based on small objects is a highly complex problem due to sensor noise, missed detections, the sudden appearance of objects of interest in the frame, major object occlusion, and an unknown and time-varying number of targets.

In recent years, the increasing use of the artificial neural networks in the field of object tracking has led to the improvement of performance in dealing with such challenges. However, to this day, object detection and tracking problems are very active fields of research, as the performance limits continue to expand every year. An example of this effort is the Visual Object Tracking (VOT) challenges [4], which focuses on assessing individual object tracking approaches, and the MOT challenge [5], which focuses on assessing multiple object tracking approaches, both of which are conducted annually.

2.1.1 Detection

The state of the art in the field of moving detection involves mainly the following four methods: the optical flow method [6, 7], the background subtraction method [8–10], the frame difference method [8, 11] and the deep learning method [12–14].

The purpose of object detection lies in obtaining an object's position and classification in an image. Generic object detection remains a very complex task, since it is difficult to design a detector that can successfully detect targets in multiple and different scenarios. For instance, a detector must take the various characteristics of target objects into account, such as size, colour, background, foreground, texture, orientation, and many more. Additionally, a detector must be able to distinguish the above-mentioned variety of characteristics from the background image in various landscapes. A detection process can be done by extracting and processing certain image properties: area, colour, contours, shape, texture, orientation, blob radius and centre, among others [15]. The combination of these properties comprises an image feature. Consequently, the quality of the feature extraction affects the efficiency of the detection.

In feature-based object detection, it is important to find invariable image features. Such features can be seen in successive frames. The aim is to model objects of interest based on these extracted features and not in raw pixels. This concept is key to understanding how images are normally processed. First, start with the most basic information about an image, the raw pixels, and try to extract more meaningful information from it, namely the features. Detection using this approach usually involves two steps. In the first phase, the specific features are calculated in two or more consecutive frames. Feature extraction will simultaneously reduce the amount of information to be processed and obtain a better understanding of the scene. In the second step, features are matched between frames.

Some of the most commonly used visual features are:

- the Scale Invariant Features Transformation (SIFT) [16],
- the Speeded Up Robust Features (SURF) [17],
- the Features from the Accelerated Segment Test (FAST) [18],
- the Maximum Stable Extremal Regions (MSER) [19],
- the Histogram of the Gradients (HoG) [20] and more recently,
- the Convolutional Neural Network (CNN) features.

This two-step approach is often used when calculating optical flow [6]. In the past, it relied mainly on artificially designed features such as SIFT, HOG and Haar-like [21] and then inserted these features into the classifier for training, such as Support Vector Machine (SVM) and Adaboost (fig. 2.2). Felzenszwalb established the Deformable Part Model (DPM) using HoG and SVM [22]. This method will perform better

when the object has any deformation or scale change. However, it cannot adapt to large rotations and is slow to calculate.

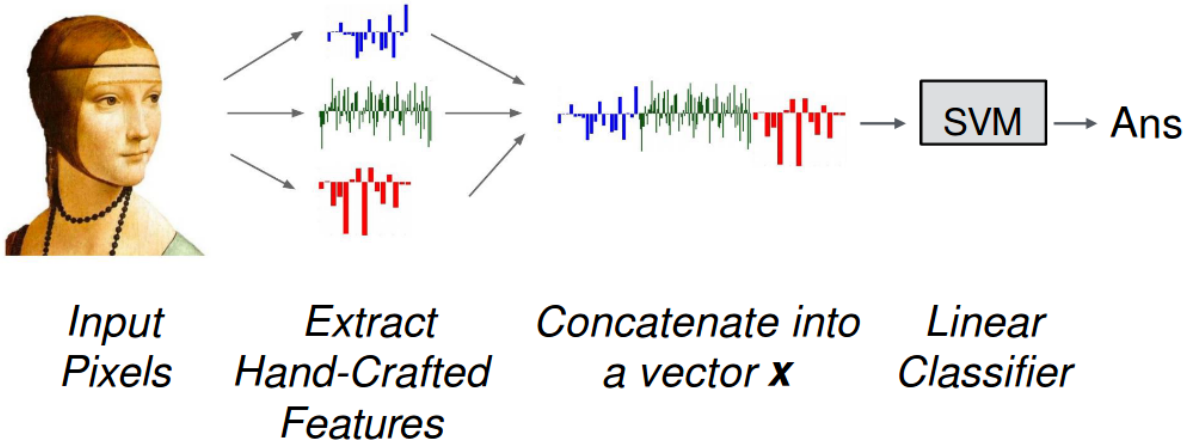


Figure 2.2: Life before Deep Learning.
Source: [23].

Another approach is dividing each video image into foreground and background to detect moving objects according to the difference in pixel intensity or colour distribution. In literature [6, 7, 9, 10, 24–27], researchers have proposed various methods of artificial design for foreground extraction. The commonly applied frame difference method is based on the grey level difference between two adjacent video images to assess the movement. It is simple to implement, without requiring background modelling, but it is vulnerable to noise and to complex scenes with motion in the background. The Gaussian Mixture Model (GMM) is more robust but requires several frames for the modelling. Furthermore, it updates the background iteratively, which results in a high computational effort and difficulty in processing the video frames with illumination variation, rare moving object and camouflage. After motion detection, connected regions labelling algorithms can obtain coordinates of regions. By scanning the input image several times, the scan mask techniques add a label to each pixel and divide the target according to the label. To achieve a faster speed, block-based methods are presented as a solution. However, these algorithms still consume too many computational resources and cannot merge objects split by noise.

Deep Convolution Neural Networks (DCNNs) for object detection and classification have attracted a lot of interest in recent years because of their powerful learning capabilities. With their ability to learn parameters themselves, a higher degree of accuracy can be achieved. The accuracy has been vastly improved compared to traditional classification algorithms, which form the foundation of deep learning-based object detection research.

With the appealing fast results of AlexNet [28], Girshick *et al.* [29] suggested the concept of object detection using a CNN. Girshick *et al.* used the Region Proposal (RP) method and proposed Regions with CNN (RCNN). The main idea is composed of two steps. First, using selective search (providing

region proposals that potentially contain objects), it identifies a manageable number of bounding-box object region candidates (Region of Interest (RoI)). Then it extracts CNN features from each region independently for classification. Given the slow detection speed of RCNN, He *et al.* proposed Spatial Pyramid Pooling in deep convolutional Networks for the visual recognition network (Spatial Pyramid Pooling Network (SPPNet)) [30]. He *et al.* also suggested Fast RCNN [31] and added a bounding box regression and a multi-task loss function. However, it maintains selective search, which is a slow and time-consuming process, making RP the bottleneck of its performance. Therefore, Ren *et al.* proposed the removal of selective search algorithms and letting the network learn the region proposals by adding a new RP network based on the Fast RCNN that resulted in the Faster RCNN [32]. The accuracy of the Faster RCNN has been greatly improved. In comparison to current detection algorithms, it is rated the best, but speed is still one of the disadvantages. Therefore, Liu *et al.* [33] proposed an end-to-end detection algorithm, a Single-Shot-Multibox-Detector (SSD), which receives suggested areas through uniform extraction and significantly improves the detection speed. Redmon *et al.* suggested a new method for detecting objects called You Only Look Once (YOLO) [34]. The RP phase was totally dropped and a single convolutional network was used as seen in fig. 2.3. YOLO splits the entire picture into the grids of $S \times S$. Each grid has a probability of class C, B as the locations of the bounding box, and a likelihood for each box. Removing the RP step enhances detection efficiency. YOLO can detect the objects while operating around 45 frames per second (fps) in real-time. YOLOv2 [35] and YOLOv3 [36] are the later improvements on YOLO. YOLOv3 makes use of multiple scale predictions and improves the basic classification network with fast detection speed, low false detection rate, and great versatility while being able to process images at 30 fps.

Figure 2.3: The YOLO Detection System. The system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. Source: [34].

In summary, state of the art object recognition networks include RCNN and its variants, SSD and its variants and YOLO and its variants. RCNN and its variants are based on a region proposal that is accurate but is time-consuming. YOLO and its variants are known for their high speed and high efficiency. SSD and its variants combine the advantages of these two methods. Although, since YOLO is already pre-trained with the classes needed (cars, trucks, buses, etc) and it performs well this will be

the framework used.

2.1.2 Data Association and Tracking

Currently, there are two main solutions for object tracking that can be subdivided into the described methods:

- Initially, objects must be detected for each frame of the video sequence. Then, one has to complete the tracking based on the detection results of consecutive frames to finally obtain the trajectory information.
- Firstly, objects must be detected in the initial frame to get the features. Then, the area matching the features in the subsequent image sequence must be found. Lastly, the objects must be tracked to get the trajectory information.

In the first solution, Meyer *et al.* [37] proposed a contour-based target detection and tracking method that obtains good results. However, this method has two big disadvantages: poor noise suppression capability and a large amount of computation. Furthermore, the object tracking based on deep learning has a higher detection accuracy than the conventional algorithm. Even so, due to the unstable nature of the detection, it may miss the target and result in a tracking failure.

The second solution is less based on object detection, which avoids the disadvantage regarding the first solution mentioned above. Extracting reference features is the key.

One approach is to extract the feature of the entire object, such as shapes, textures, colour histograms or image edges. By combining several features, the reliability of the object is improved. After feature extraction, the object is redetected using the similarity measurement to obtain object tracking.

Another approach is extracting feature points from the object, for which the Harris Corner [38] and SIFT [16] are frequently used methods. On the one hand, feature point-based methods can adapt to changes in rotation and lighting of the object. But on the other hand, excessive feature extractions often result in difficulty in matching, while too few feature extractions can easily lead to false positives. Moreover, the feature extraction process is complicated and time-consuming.

One example of the feature-based tracking algorithm is the Lucas-Kanade (LK) method proposed in [39] and later improved in [40]. Even if part of the object is occluded, the tracking task can continue using the feature information. However, it is still sensitive to image noise or blur. The quality of features is hugely dependent on the setting of extraction parameters. Shi and Tomasi features are proposed in [41] to deal with the issue of selecting features that can be tracked well. The Kanade-Lucas-Tomasi (KLT) feature tracker is the result of LK method with these good features to track. Additionally, the correspondence between consecutive frames is a challenge and has an impact on tracking performance.

3

Proposed Approach: Merging object detection with point tracking

Contents

The key elements to automatic vehicle tracking based on the DBT framework are the object detector, the object tracker design and the strategy for integrating the detector and the tracker.

To better illustrate the problem and how this work proposes to tackle it, refer to fig. 3.1, where three consecutive images of a typical real traffic situation are depicted. Common systems rely on simple nearest neighbour matching to associate detections between consecutive frames. However, due to the high speed of vehicles, one missed detection in the second frame of fig. 3.1, prevents the correct association to the correct vehicle in the third frame.

Figure 3.1: Example of the problem and proposed solution. Top image - Object detector and feature selection identify vehicles and image features inside the bounding box. Middle image - object detectors often miss bluntly. However feature trackers can still match corresponding points (blue line). Bottom image- if objects are detected again, the feature tracker allows the association of the correct object (red lines).

In a concise way, the main idea is that by combining object detection with feature tracking, whenever the detector fails to recognise one vehicle, the image features are still detected and can be tracked. This way one is still able to keep track of the vehicles and know in the following frames which vehicles are which, as shown in fig. 3.1 where the identification of each bounding box matches the identification on the first frame. Considering the complexity and application feasibility of the algorithm, in the object detection step, the detection result provided by YOLOv3 were post-processed and then used as the input for the tracker. For the tracker, an KLT tracker was implemented to extract and track image features. Finally, these features are combined with YOLO detections to track multiple vehicles in traffic scenes.

In summary, in this work the tracking of multiple vehicles will be divided into three main steps: detection, prediction, and data association:

- Detection: Using a state of the art CNN-based object detector, locate vehicles in video frames
- Prediction: Predict the object locations in the next frame by tracking feature points using reliable methods
- Data association: Use the predicted locations to associate detections across frames to form detection tracks

In the next sections, the choice of techniques to handle each of the subproblems is specified and details on their inner workings and how the whole system was implemented are provided.

3.1 Object detection - You Only Look Once (YOLO)

The object detection stage aims to identify the category and location of the vehicle object in a picture. Object detection algorithms for natural images can be roughly divided into two categories. One based on traditional handcrafted features, commonly used until 2013, and a dominance of deep learning thereafter.

Since the emergence of deep learning, object detection has made a huge breakthrough. The two most important kinds of deep learning are:

(a) region proposal-based method represented by RCNN such as Fast-RCNN, Faster-RCNN, among others;

(a) regression-based method represented by YOLO such as YOLOV3, SSD and others.

The former is superior in accuracy, and the latter in speed. Because the deep learning method has an excellent performance in object detection in real time, the YOLOV3 algorithm was selected to implement the detection task.

YOLO is an algorithm of object detection of images using a single CNN and in a single inference. In the initial paper, the workflow of YOLO works as follows: (1) Pre-train a CNN network on the image classification task. (2) Divide an image into $S * S$ cells. If an object's centre falls into a cell, that cell is responsible for detecting the existence of that object. Each cell will propose a) the location of B bounding boxes, b) a confidence score and c) a probability of object class conditioned on the existence of an object in the bounding box. In total, one image contains SxSxB bounding boxes with each box corresponding to 4 location prediction, 1 confidence score and C conditional probabilities for object classification. (3) The final layer of the pre-trained CNN is modified to output a prediction tensor of size $S * S * (B * 5 + C)$ fig. 3.2 ilustrates YOLO detection model.

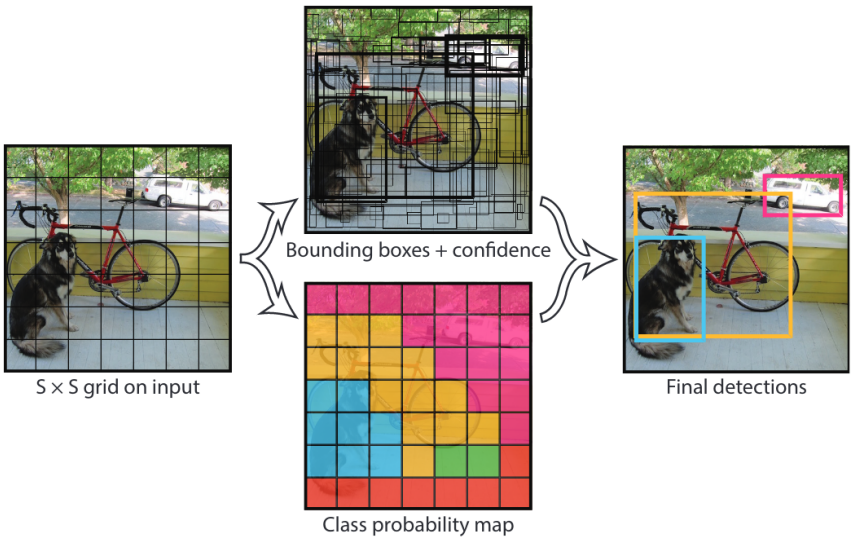


Figure 3.2: YOLO system models detection as a regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes, confidence for those boxes,and C class probabilities. These predictions are encoded as an $S * S * (B * 5 + C)$ tensor.
Source: [34].

The loss consists of two parts, the localisation loss for bounding box offset prediction and the classification loss for conditional class probabilities. Both parts are computed as the sum of squared errors. The loss function only penalises classification error if an object is present in that grid cell. It also only penalises bounding box coordinates error if that predictor is responsible for the ground truth box.

YOLO's base model is similar to GoogleNET [42] and uses Darknet architecture. The final prediction of shape $S*S*(B*5+C)$ is produced by two fully connected layers over the whole convolutional feature map.

YOLO is improved in the second paper and YOLOv2 is born. Since some of the complaints about YOLO were related to the difficulty in detecting small objects, modifications were made in that regard. One of these modifications was fine-tuning the base model with high-resolution images to improve the detection performance. The method of predicting bounding boxes was also changed. Rather than predicting the bounding box position with fully-connected layers over the whole feature map, YOLOv2 uses convolutional layers to predict locations of anchor boxes, like in Faster RCNN. The prediction of spatial locations and class probabilities are no longer coupled. This lead to an increase in recall. Unlike Faster RCNN, which uses hand-picked sizes of anchor boxes, YOLOv2 runs k-means clustering on training data to find good priors on anchor box dimensions. YOLOv2 also formulates the bounding box predictions differently, in a way that it would not diverge from the centre location too much. Multi-scale training is implemented to be robust to an input of different sizes. For that, a new size input dimension is randomly sampled every 10 batches. YOLOv2 adopts a different base model, lighter, DarkNet-19 supplemented with 11 more layers for object detection. With a 30 layer architecture, YOLOv2 often

struggles with small object detection.

Finally, YOLOv3 is created by applying several design changes to YOLOv2. These changes are based on recent advances in object detection. Firstly, for the prediction of confidence score for each bounding box, instead of the sum of squared errors used on previous versions, YOLOv3 uses logistic regression. Since one image might have multiple labels and not all are guaranteed to be mutually exclusive, YOLOv3 uses multiple independent logistic classifiers for each class rather than one softmax layer. Inspired by image pyramid, YOLOv3 has multi-scale prediction by making predictions at three different scales among the added convolutional layers. The base model is also yet again changed. YOLOv3 relies on the new Darknet-53. This variant of Darknet originally has a 53 layer network trained on Imagenet. For the task of detection, 53 more layers are staked onto it, giving a 106 layer fully convolutional underlying architecture for YOLOv3, shown on fig. 3.3. This is one of the reason behind the slowness of YOLOv3 compared to YOLOv2. Another one is that YOLOv3 predicts more bounding boxes than YOLOv2, for an input of the same size. This is due to YOLOv3 predicting boxes at 3 different scales. Less speed has been traded off for a boost in accuracy. While the earlier variant ran on 45 fps on a Titan X, the current version clocks about 30 fps but it is more accurate.

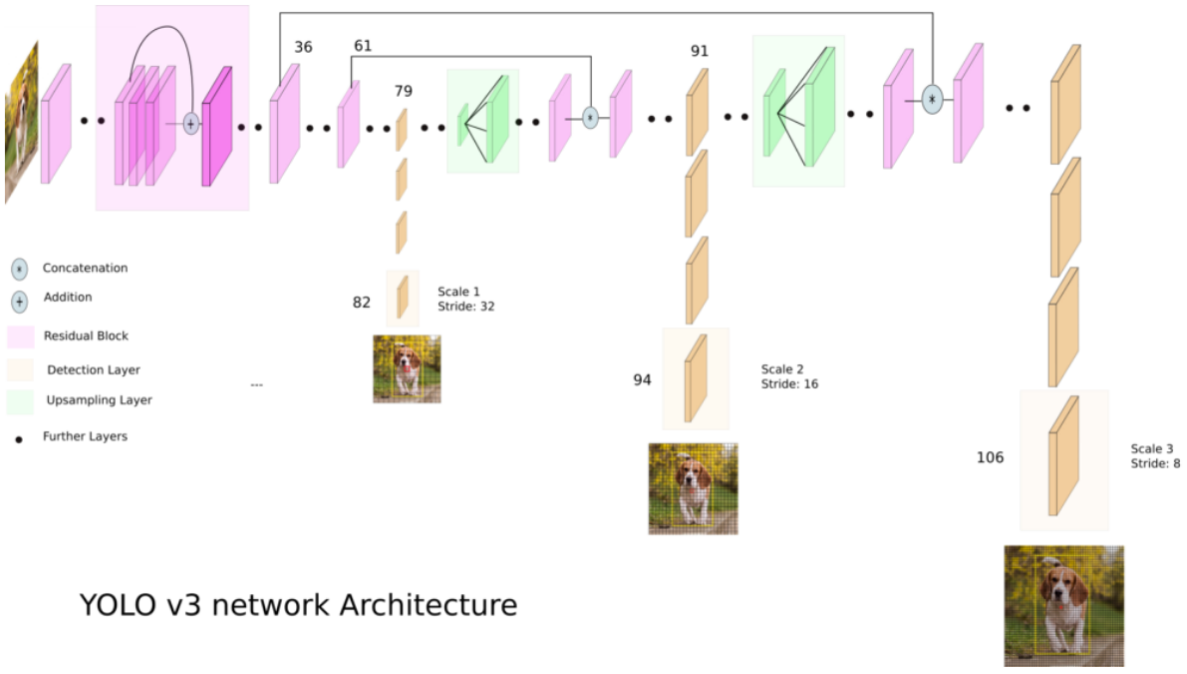


Figure 3.3: YOLOv3 architeture.
Source: [43].

3.2 Prediciton and Tracking: The Kanade-Lucas-Tomasi Tracker (KLT)

Tracking is the process of locating a moving object or multiple objects over time in a video stream. In general, tracking is used in scenes where the displacement between consecutive frames is very small compared to image size. In short tracking can be viewed as the 2D position along time $[x(t), y(t)]$ of selected image points.

The LK tracker is a feature tracker technique firstly proposed by Lucas and Kanade in the 1980's [39], improved in 1991 by Tomasi and Kanade [40] and in 1994 by Shi and Tomasi that included a keypoint selection method [41]. In this work we used the later form of it, the KLT tracker.

The KLT feature tracker, hinges on Taylor series approximations of the image sequence and reduces the cost of the traditional image registration techniques by lowering the dimensionality of the problem, and achieving the 'best match' of an image by using a reduced number of potential matches. Furthermore, relying on image pyramids to reduce the inter-frame displacement (at each level), it exhibits impressive precision but can also cope with significant displacements between consecutive images.

3.3 A summary of the Kanade-Lucas-Tomasi (KLT) approach to feature extraction

The KLT begins with a first data set which is cleaned/transformed with the purpose of getting new non-redundant and informative data allowing to facilitate the next steps of learning and generalisation, often leading to easier interpretations.

Some details related with the applied technique are described. The registration algorithm description includes a brief description of one-dimensional case, and an alternative derivation is included to motivate the generalisation to multiple dimensions,. Also the detection and tracking of point features, improvements and variations are detailed briefly.

1. Image registration problem

   Considering two images as being represented by two functions $F(x)$ and $G(x)$ (which gives the respective pixel values at each location x in two images, where x is a vector), the registration problem can be understood as searching for the best offset h (disparity vector) that minimises some measure of the difference between the value of F in a neighbourhood of x, $F(x + h)$, and the value of $G(x)$ when x belongs to the region under study R, as show in fig. 3.4.

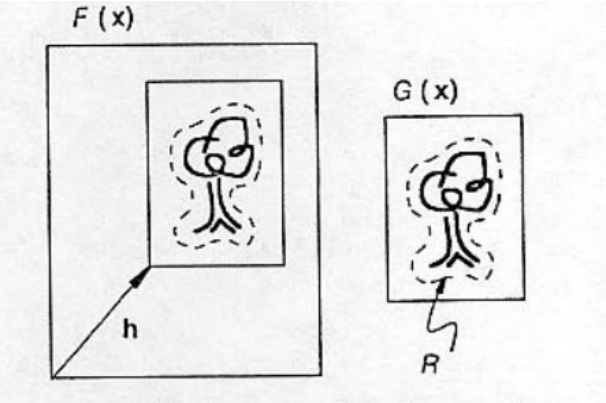


Figure 3.4: The image registration problem.
Source: [39].

Some examples of these measures consider the sum of absolute value of the difference $\Delta_x = F(x+h) - G(x)$ when $x \in R$,

$$\sum_{x \in R} |\Delta_x| \,, (\textbf{L}_\textbf{1} \textbf{ norm}) \,; \tag{3.1}$$

Others consider the square root of the sum of the power 2 of the difference Δ when x $\in R$,

$$\sqrt{\sum_{x \in R} \Delta_x^2}, (\textbf{L}_\textbf{2} \textbf{ norm}) \,; \tag{3.2}$$

Alternatively, one can use a standardised measure, e.g. the symmetric of the correlation coefficient

$$\frac{-\sum_{x \in R} F(x+h)G(x)}{\sqrt{\sum_{x \in R} F(x+h)^2} \sqrt{\sum_{x \in x} G(x)^2}}. \tag{3.3}$$

2. Registration algorithm

The KLT feature tracker based on [39] and [40] proposes that local searches use gradients weighted by an estimate of the second derivative inverse of the image. To illustrate, one can describe the one-dimensional case. Consider the displacement h between two images $F(x)$ and $G(x) = F(x+h)$. When h is small enough, the derivative $F'(x)$ is approximated by the incremental rate given by

$$F'(x) \approx \frac{F(x+h) - F(x)}{h} = \frac{G(x) - F(x)}{h} \tag{3.4}$$

and

$$h \approx \frac{G(x) - F(x)}{F'(x)} \tag{3.5}$$

The estimate of h clearly depends on each x. To get a better estimate of h the simple mean of all

estimates obtained for each x of the interest region R is computed

$$h = \frac{\sum_{x \in R} h_x}{\sum_{x \in R} 1} = \frac{\sum_{x \in R} \frac{G(x)-F(x)}{F'(x)}}{\sum_{x \in R} 1} \tag{3.6}$$

or a weighted mean proposed in [39] where each weigh is related with the absolute value of second derivative of $F(x)$,estimated by

$$F''(x) \approx \frac{G'(x) - F'(x)}{h}. \tag{3.7}$$

In this way, the improved estimate of h using the weighted mean is given by

$$h = \frac{\sum_{x \in R} w_x h_x}{\sum_{x \in R} w_x} = \frac{\sum_{x \in R} w_x \frac{G(x)-F(x)}{F'(x)}}{\sum_{x \in R} w_x} \tag{3.8}$$

with

$$w(x) = \frac{1}{|G'(x) - F'(x)|}. \tag{3.9}$$

For each estimate of h, the estimate of F is updated. The iterative process is performed using the expression

$$\begin{cases} h_o = 0 \\ h_{k+1} = h_k + \dfrac{\sum_{x \in R} w_x \frac{G(x)-F(x+h_k)}{F'(x_k)}}{\sum_{x \in R} w_x} \end{cases}. \tag{3.10}$$

An alternative derivation can be done, using the usual linear estimate of a function in a neighbourhood of x. One can easily see from the incremental rate given by

$$F'(x) \approx \frac{F(x+h) - F(x)}{h} \tag{3.11}$$

that

$$F(x+h) \approx F(x) + hF'(x) \tag{3.12}$$

considering small values h. The optimal value of h is the one that minimises the sum of the square errors SSE defined by the L_2 norm

$$SSE = \sum_{x \in R} \left(F(x+h) - G(x)\right)^2. \tag{3.13}$$

This process is similar to the least squares method. As usual, to minimise SSE, one must impose that $\frac{\partial SSE}{\partial h} = 0$ and $\frac{\partial^2 SSE}{\partial h^2} > 0$.

The h estimate is

$$h \approx \frac{\sum_{x \in R} F'(x) \left[G(x) - F(x)\right]}{\sum_{x \in R} F'(x)^2}, \tag{3.14}$$

taking into account that the iterative process is

$$\begin{cases} h_o = 0 \\ h_{k+1} = h_k + \dfrac{\sum_{x \in R} w_x F'(x + h_k) \left[G(x) - F(x + h_k) \right]}{\sum_{x \in R} w_x F'(x + h_k)^2} \end{cases} \tag{3.15}$$

with $w_x = F'(x + h_k)^2$.

For the multidimensional case, the process is similar, generalising the least squares errors approach: the n-dimensional vector **h** estimate is obtained by minimising the sum of square errors L_2 norm

$$SSE = \sum_{x \in R} \left(F(\mathbf{x} + \mathbf{h}) - G(\mathbf{x}) \right)^2 \tag{3.16}$$

where $\mathbf{x}$ is a n-dimensional vector. Again, the linear approximation is used

$$F(\mathbf{x} + \mathbf{h}) \approx F(\mathbf{x}) + \mathbf{h} \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} \right)^T , \tag{3.17}$$

with $\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}}$ being the Jacobian matrix (matrix of gradients). The condition $\frac{\partial SSE}{\partial \mathbf{h}} = 0$ conduces to the estimate

$$\mathbf{h} \approx \left[\sum_{\mathbf{x} \in R} \left[G(\mathbf{x}) - F(\mathbf{x}) \right] \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} \right) \right] \left[\sum_{\mathbf{x} \in R} \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} \right)^T \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} \right) \right]^{-1} . \tag{3.18}$$

3. Improvements

The KLT feature tracker allows for the extension to more elaborated transformations [40] (e.g. rotation, scaling, shearing). In this case one has $G(\mathbf{x}) = F(A\mathbf{x} + \mathbf{h})$, where A is some linear transformation matrix. In this case one must minimise the sum of square errors L_2 norm

$$SSE = \sum_{x \in R} \left(F(A\mathbf{x} + \mathbf{h}) - G(\mathbf{x}) \right)^2 . \tag{3.19}$$

The following linear approximation must also be considered

$$F((A + \Delta A)\mathbf{x} + (\mathbf{h} + \boldsymbol{\Delta}\mathbf{h})) \approx F(A\mathbf{x} + \mathbf{h}) + (\Delta A\mathbf{x} + \boldsymbol{\Delta}\mathbf{h}) \left(\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} \right)^T , \tag{3.20}$$

Another extension of KLT technique can be achieved when the brightness in the two images is distinct. In this situation, one can consider this issue as a linear transformation, $F(x) = \alpha G(\mathbf{x}) + \beta$, where α is related with the contrast correction and β is related with the brightness correction. The minimisation of the L_2 norm relative to A, $\mathbf{h}$, α and β is

$$SSE = \sum_{x \in R} \left(F(A\mathbf{x} + \mathbf{h}) - (\alpha G(\mathbf{x}) + \beta) \right)^2 . \tag{3.21}$$

Still in [40], the authors propose a criteria that allows to decide which tracking features are adequate for the tracking algorithm. It consists in the imposition that the eigen values of gradient matrix are larger than a certain established threshold so the tracking feature is chosen. This issue allows to avoid the badly conditioned matrix problems.

Another improvement of KLT tracker can be found in [41], where a scheme that verifies if the chosen features were correctly tracked is implemented.

3.3.1 Implementation

In this thesis, an implementation in MATLAB2018a was used, namely in the computer vision toolbox that provides video tracking algorithms, such as Continuously Adaptive Mean Shift (CAMShift) and KLT. The vision.PointTracker tool is applied.

The point tracker object tracks a set of points using the KLT feature-tracking algorithm. The point tracker can be used for video stabilisation, camera motion estimation, and object tracking. It works particularly well for tracking objects that do not change shape and for those that exhibit visual texture. The point tracker is often used for short-term tracking as part of a larger tracking framework which is the case in this thesis.

As the point tracker algorithm progresses over time, points can be lost due to lighting variation, out of plane rotation, or articulated motion. To track an object over a long period of time, one needs to reacquire points periodically. In this work the points are reacquired for every detection.

3.4 Association

Data association is the process of associating detections corresponding to the same physical object across frames. The temporal history of a particular object consists of multiple detections and is called a track. A track representation can include the entire history of the previous locations of the object. Alternatively, it can consist only of the object's last known location and its current velocity. In this case, a track consists of the previous locations of features associated with the object.

In this work, the association of detections across frames is done using the KLT method in combination with the results from YOLOv3. Starting with the YOLOv3 bounding boxes, features are extracted from each bounding box according to the Shi-Tomasi criteria for good features [41]. These features are tracked into the following frame using a KLT tracker. To each group of tracked features, a rectangle is

fitted and then compared with the YOLOv3 bounding boxes for the current frame. If a certain degree of overlap happens then it is considered to be the same bounding box. If there is no correspondence with YOLOv3 bounding boxes, the fitted rectangle is considered the bounding box for the current frame and is propagated, compensating for failures in the detection step. Since features are extracted from each bounding box, features in the background can happen. To avoid propagation of bounding boxes with only background features (without proper detection from YOLOv3) a score is given to these boxes. If there's a match with a YOLOv3 detection the score is increased and for each iteration without detection, the score is decreased.

3.5 Homography

In computer vision, homography is a transformation matrix H which, when applied on a projective plane, maps it to another plane (or image). In this work, the intention is to produce a bird's eye view image of the scene. It is assumed the world is flat on a plane and maps all pixels from a given view point onto this flat plane through homography projection. This assumption works well in the immediate vicinity of the camera. For faraway features in the scene, blurring and stretching of the scene is more prominent during perspective projection, shown in fig. 3.5.

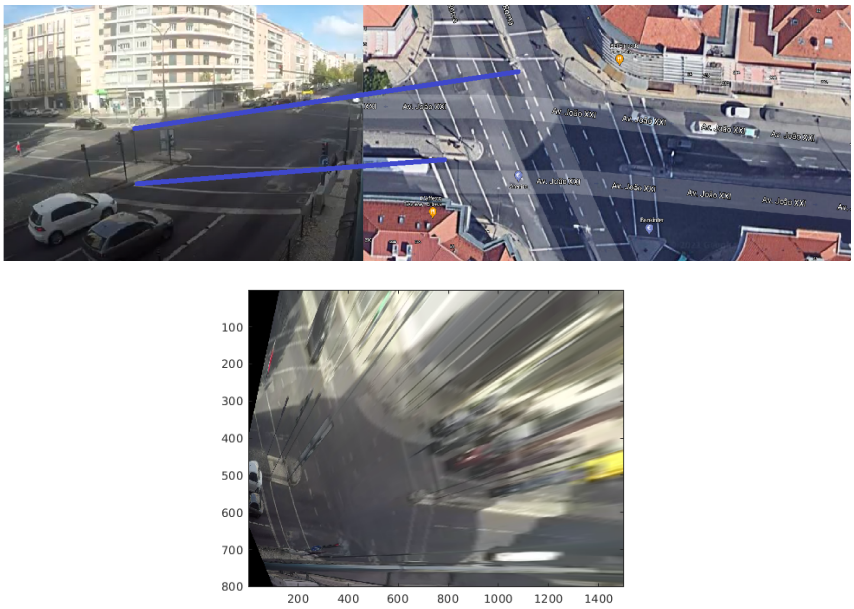


Figure 3.5: Example of projection from camera perspective into a bird's eye perspective. The bottom image shows the blurring on the scene further away from the camera. The homography is computed by mapping at least 4 corresponding points in both images.

The usefulness of this mapping rests on the fact that Google Maps images are registered to terrain maps, so they can be used to build such bird's eye view as well as obtaining metric measurements.

Considering the world flat and having a fixed camera makes the formulation of this homography a simple case. The planar homography relates the transformation between two planes (up to a scale factor) and is presented in eq. (3.22), where homogeneous coordinates of the corresponding points are x' and x.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.22}$$

The homography matrix is a 3x3 matrix but with 8 Degrees of Freedom (DoF) and can be estimated up to a scale by linear methods, namelly the DLT (Direct Linear Transform). In general there are two common normalisations: normalising one element, for example

$$h_{33} = 1 \tag{3.23}$$

or normalising its norm, that is imposing $\|H\|^2 = 1$

$$h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{21}^2 + h_{22}^2 + h_{23}^2 + h_{31}^2 + h_{32}^2 + h_{33}^2 = 1 \tag{3.24}$$

In this work, we used MATLAB tools and functions, namely $cpselect$ to select the homologous corner points in both perspectives as mentioned before. For the best results, the selection will be spread out across the image and more than 20 pairs of points will be chosen. The function $cp2tform$ will be used to compute the transformation. Afterwards it will be possible to project the detection from the camera plane onto the ground plane.

4

Experiments and Results

Contents

In this section, the implementation of the system is described, as well as, the steps to evaluate each phase of the system. The dataset used for each are also characterised. Following that, the final result of the tracking system are presented and analytics are calculated. In the end, the results and analytics are commented and analysed.

4.1 Implementation of the detection and tracking pipeline

In the object detection phase, the framework YOLOv3 available on Redmon's webpage was used [44]. After running the video frames throughout YOLOv3, the output was then processed into a MATLAB structure. Which stores bounding boxes' coordinates, confidence and class, as well as frame number and file path of the frame.

As the idea of this proposal is to track vehicles and show their trajectories, the homography was calculated before the tracking phase to allow the projection of the tracking boxes onto the map plane during the tracking. Using MATLAB function $cp2tform$ and $imtransform$ the spatial transformation was calculated between the camera plane and the bird's view of the traffic intersection filmed. This way it is possible to project the detected vehicles into a map and make the trajectory more perceptible for the human eye.

Afterwards the tracker phase was implemented using the MATLAB function $vision.PointTracker$ to track corner point features from one frame onto the next one, using the KLT method. These features are extracted from each of the bounding boxes from YOLOv3. Each has an identifier for its corresponding bounding box. After tracking the features, a rectangular bounding box is fitted to features with the same identifier. Then they are compared with the bounding boxes obtained from YOLOv3 for the analogous frame. To match boxes, the intersection of the fitted bounding box with the bounding box obtained with YOLOv3 must be at least 40% of the area of the box fromYOLOv3. If none satisfies this condition, the algorithm considers it a false negative from the detector and stores the fitted boxes as a detection.

A video of a crossroad will be the base to test this system.

4.2 Evaluation

A reliable tracking requires a reliable detection. For that, it is necessary to evaluate YOLOv3 results. To quantify YOLOv3's performance, the number of true and false detections needs to be assessed. The definitions are as follow:

- False positive - incorrect detections, vehicles that were detected when they should not have been;
- True positive - correct detections, vehicles that were detected;

- False negative - vehicles that were not detected;

To assess YOLOv3 performance we computed key indicators such as precision, eq. (4.1) and recall,eq. (4.2), in a specific dataset created "in-house". Subsequently, KLT parameters were tuned and the tracker performance evaluated. In the next sections we will characterise the datasets used and describe the key indicators.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.1}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.2}$$

Dataset Characterization

All datasets used were extracted from a webcam video of a crossroad with resolution of 560x690 pixels.

The dataset used to assess YOLOv3 performance consists of 30 randomly selected frames with 244 annotations of vehicles in total. This initial dataset was resized four times into 90% ,80% ,70% and 60% of the original size.

The datasets used to tune KLT parameters consists of 8 sets of 100 frames extracted at 15 fps. Each set starts in a random frame of the video.

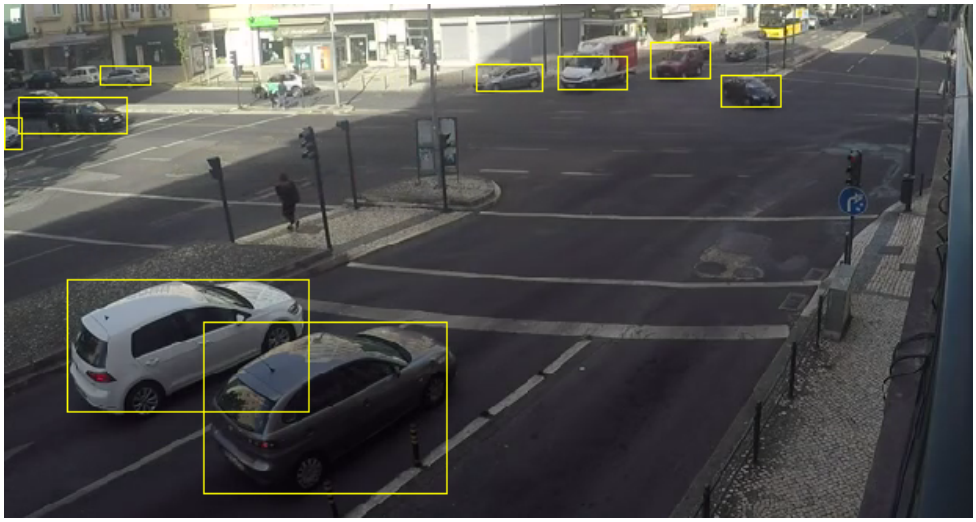
Finally, the dataset used to obtain the final results of the whole system consists of 450 frames extracted at 15 fps.

YOLO Performance

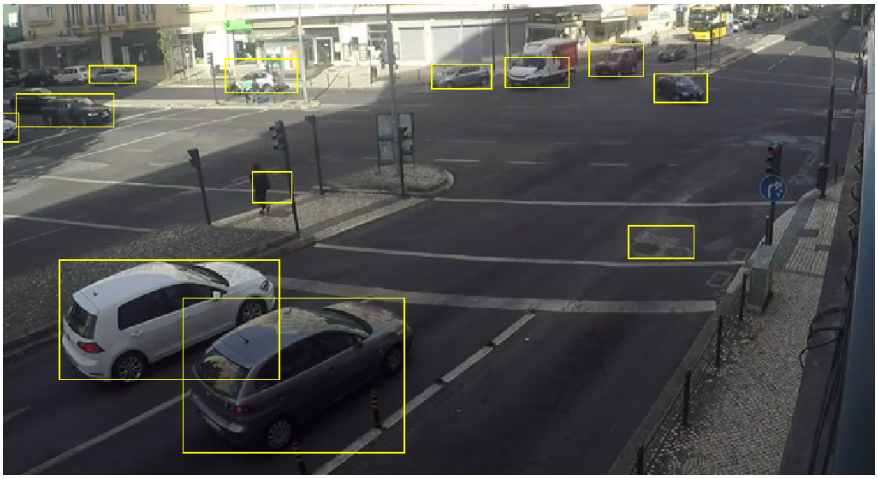
Assessing YOLOv3 performance involves comparing YOLOv3's results for different sized inputs. For the comparison , the confidence score of YOLOv3 entry parameter was set to 0.7 for all the frames, meaning it only considers detections with a confidence higher than 70%. This value was obtained throughout several tests, as to have a lower amount of false positives while still being able to detect the majority of the vehicles in the scene. The fig. 4.1 and fig. 4.2 show examples of the frames used. The recall and precision calculated for our dataset is presented in table 4.1.

Table 4.1: YOLOv3 performance for several dimensions of input in the context of this thesis. The dataset used had 30 frames.

	True Positive	False Positive	False Negative	Precision	Recall
original size	190	4	54	97.9%	77.9%
90% size	197	10	47	95.2%	80.7%
80% size	189	8	55	95.9%	77.5%
70% size	128	13	116	90.7%	52.5%
60% size	111	4	133	96.5%	45.5%

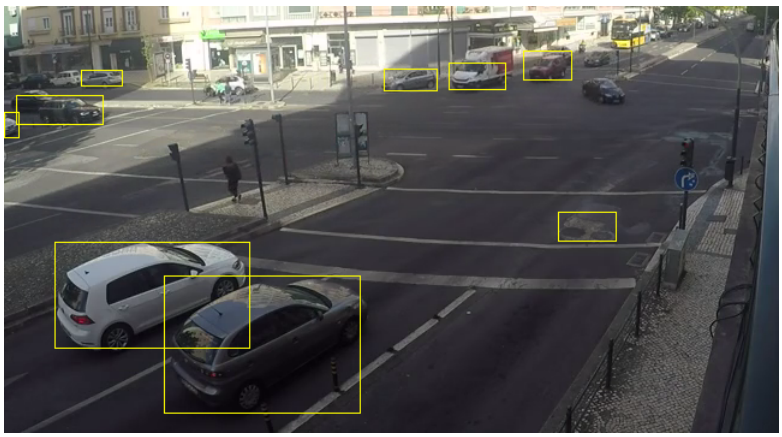


(a) Original size.

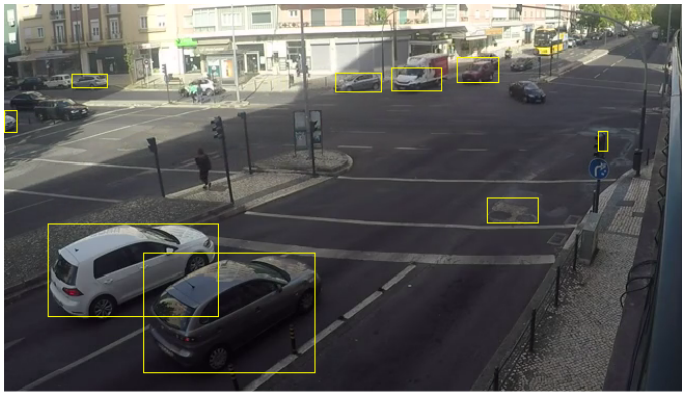


(b) 90% original size.

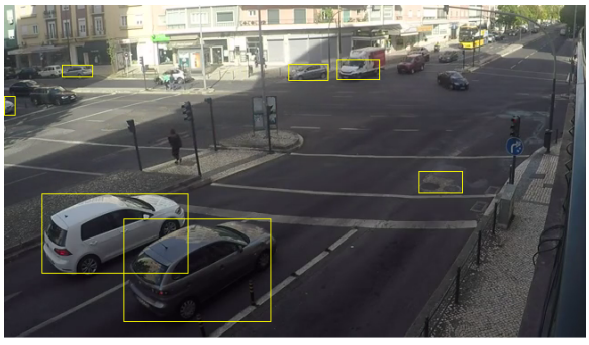
Figure 4.1: Example frames used in testing YOLOv3 performance.



(a) 80% original size.



(b) 70% original size.



(c) 60% original size.

Figure 4.2: Example frames used in testing YOLOv3 performance (cont.).

The initial step of tracking needs to have totally reliable detection, which means no false positives. Therefore the original size is the most appropriate as it has the highest precision. Although the original size corresponds to the second-best recall, this only means a higher number of false negatives. Since the system is using a KLT tracker, it will compensate for that lack of detections.

Tuning of KLT

In the tracking phase, some parameters can also be set. In this case, the used MATLAB function $vision.PointTracker$ allows for the configuration of the number of pyramid levels as well as the forward-backward error threshold.

The point tracker implementation of the KLT algorithm uses image pyramids. The tracker generates an image pyramid, where each level is reduced in resolution by a factor of two in width and height compared to the previous level. The point tracker begins tracking each point in the lowest resolution level. It continues tracking until convergence and will propagate the result to the next level as the initial guess of the point locations. A higher number of pyramid levels allows the algorithm to handle larger displacements of points between frames. However, the computation cost also increases.

Using forward-backward error allows the tracker to track each point from the previous to the current frame, then track the same points back to the previous frame and calculate the bidirectional error. This value will be the distance, in pixels, from the original location of the points to the final location after the backward tracking. When the error is higher than the set value, the points are considered invalid. Meaning that, by using bidirectional error, points that could not be reliably tracked are eliminated.

Since the dataset used was obtained by sampling 15 frames per second from a crossroad, generally speaking, the displacement of points will not be large. Therefore the values set were: 2 pyramid levels and 3 pixels for the forward-backward error threshold. This allows for reliable tracking while not being very computationally heavy.

Another point worth evaluating is how well the KLT tracking component performs. How good would the results be if based exclusively on tracking the initially detected features? This instance was evaluated by firstly randomly selecting a frame of the video. Then obtaining the YOLOv3's detection and extracting features within these detections. Finally, initialising tracking and running it for 100 frames. It ran for 8 different initial frames with the results presented in table 4.2.

As displayed in fig. 4.3, features were tracked throughout most of the frames, with an average of 88.8% being tracked until the end. Features will inevitably be lost, either due to being weaker or because they leave the frame. Since a vehicle takes around 100 frames to enter and leave the frame, most of the features lost will probably either correspond to vehicles leaving frame or to weaker features not being reliably tracked. The number of features lost by frame for each iteration, as well as the average, are shown in fig. 4.4.

Table 4.2: KLT tracker performance for 8 iterations of tracking across 100 frames, with randomly selected initial frames.

Iteration	1	2	3	4	5	6	7	8	average
#detected features initially	356	338	336	406	434	540	428	417	406.8
# of features tracked until frame 25	343	338	336	392	433	399	407	353	375.1
# of features tracked until frame 50	342	338	336	391	420	380	397	328	366.5
# of features tracked until frame 75	342	338	336	391	396	379	375	323	360
# of features tracked until frame 100	342	338	336	391	391	372	354	317	355.1
% of features tracked until frame 100	96.1%	100%	100%	96.3%	90.1%	68.9%	82.7%	76.0%	88.8%

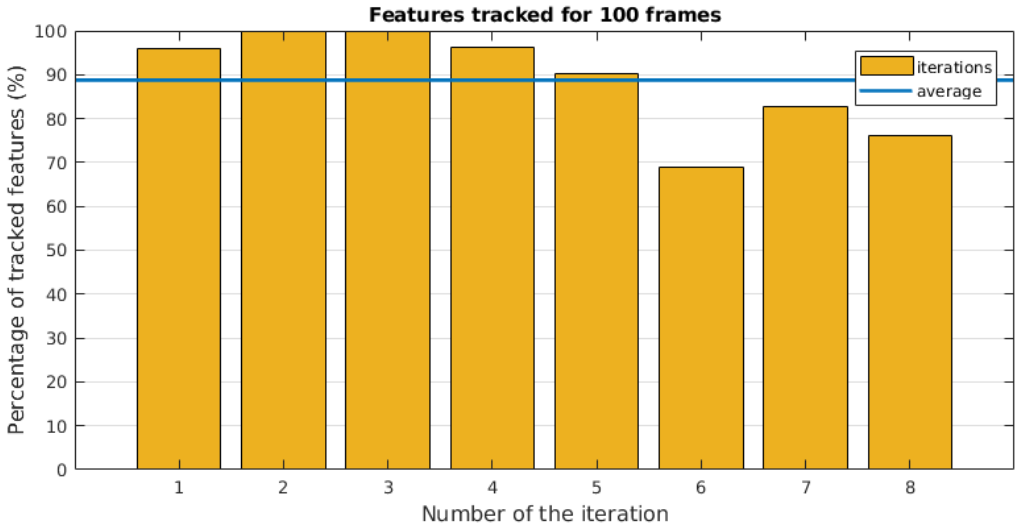


Figure 4.3: Percentage of features tracked until the 100th frame. 8 iterations of tracking for 100 frames, starting from a random video frame.

It is interesting to understand why iterations 2 and 3 have 100% of features tracked for 100 frames, while iteration 6 only has less than 70%. When looking at the movement of the vehicles for each of the iterations we can discover why. For iterations 2 and 3, the vehicles are mostly waiting for the traffic light. On iterations 1, 4 and 5, the vehicles are mostly moving in parallel to the camera. Vehicles on iteration 6 are mostly facing forward, starting on the traffic light furthest away from the camera. During the first half, the ones moving are turning to the left. For iterations 7 and 8, about half of the vehicles are waiting on the traffic lights while the other half is coming closer towards the camera.

If vehicles are not moving features will be easily tracked, therefore explaining the results for 2 and 3. The rest can be explained by the fact that KLT essentially considers the movement of the features is a translation. Although it deals better with scaling and rotation than standard LK, it still has its limitations. Which mean when a car is coming towards the camera, it is ”zooming in” on the features and in the long run the approximation of that movement to a translation is no longer possible leading to the loss of features. Another difficulty is the turning of the vehicles. Iteration 6 has a high amount of lost features for the first 40 frames as features will suffer a high rotation and deformation, as well as disappearing on the side of the car no longer visible when turning.

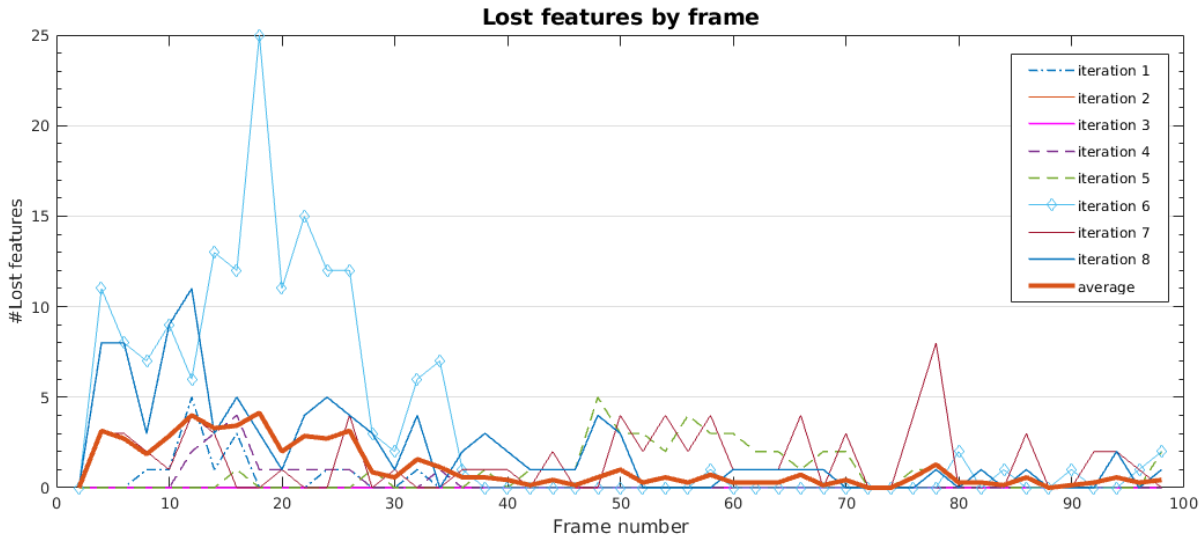


Figure 4.4: The amount of features lost from one frame to the next. 8 iterations of tracking for 100 frames, starting from a random video frame.

If we only looked at this situation from a tracking perspective, a KLT tracker would not be an appropriate choice for this system. However, we are not using KLT for tracking itself but to connect YOLO's detections. Therefore, it is an appropriate approximation for short distances.

Bounding boxes association

For the KLT tracker phase it was important to find a balance for the matching of YOLOv3 bounding boxes with the predicted bounding boxes from KLT. This matching is based on the area of the intersection between both boxes being at least 40% of the area of the box from YOLOv3. When none of the detected boxes meets this criterion then the predicted box will be considered a detection. New features will be detected inside these matched boxes and used to predict the boxes in the next frame. The threshold of 40% was obtained by testing several values and assessing which lead to better association results. If YOLOv3 detects two vehicles very close, their bounding boxes will cover part of the other vehicle. Then, using too low of a threshold the system will increase identity switches. While using too high of a threshold will be too restrictive and rarely able to match. When using 40%, as shown in fig. 4.5, a good level of matching is obtained even when bounding boxes with significant overlap exist.

(a) The grey car on the right is identified as 1 and the white car on the left is identified as 2.

(b) The grey car on the right is still identified as 1 and the white car on the left is still identified as 2.

Figure 4.5: Example of using 40% as the threshold of intersection area for matching.

4.3 Results and Discussion

4.3.1 Results

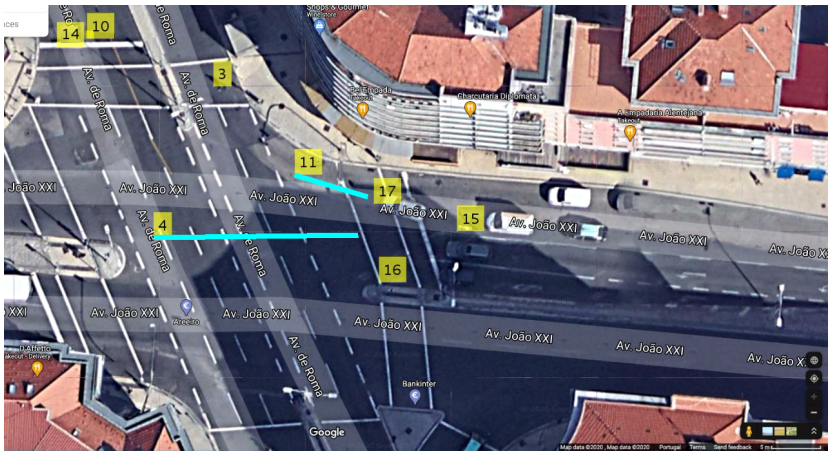
The final tracking results of this system were obtained using a real traffic scenario video, this dataset has been characterised in the evaluation section. The tracking results concerning the camera view are represented in fig. 4.6, while the ones concerning the map perspective are illustrated in fig. 4.7. The trajectories displayed in fig. 4.7 correspond to the last detected positions for each vehicle, projected onto a map view from Google Maps. The benchmark for this work will be presented in the following section as well as a small analysis of the results. Additionally , metrics about the tracked vehicles, such as the spatial occupation of the road and velocity by region, are calculated and displayed later.

(a) Initial frame.

(b) 30 frames later.

Figure 4.6: Experimental results - camera view.

(a) Initial frame.



(b) 30 frames later.

Figure 4.7: Experimental results - map view.

Benchmark

For this work, the benchmark will evaluate how many times, for a full track, did YOLOv3 not detected the vehicle. During tracking, 60 vehicles were identified. For the 450 frames in this dataset there were 4943 instances of track. On average, a full track lasts for 82 frames, with the longest track being 427 frames and the shortest only 12. From the 4943 instances, there was no YOLOv3 detection for 1284, which means 25.9% of instances came from the KLT tracking. Since in this tracking, there are 2 vehicles that are detected in every frame and their tracks last for 427 frames it is relevant to calculate the average based on each individual track. Considering the individual track length and individual lack of YOLOv3 detection, one has, on average, 21.8% of instances from KLT exclusively.

By analysing the final results, with examples displayed in fig. 4.6 and fig. 4.7, we can conclude the system devised in this thesis achieves a reliable matching and tracking of multiple vehicles. During missed detections from the detector and partial occlusion of vehicles, the tracking performs well and is able to keep following the vehicles. As it can be seen in benchmarking, this is a more robust tracking system than a tracking system solely based on detections, as this would have considered new vehicles when YOLOv3 lacks detection on an intermediate frame.

4.3.2 Analytics

After the tracking and using all the stored data that comes from it, some metrics about occupation and velocity were calculated and presented in fig. 4.8, fig. 4.9 and fig. 4.10. In fig. 4.8, the occupation of the crossroad during the duration of the tracking is represented as a heatmap. In it, the shape of the crossroad is clearly visible, with some hot spots existing near the areas where traffic lights are positioned.

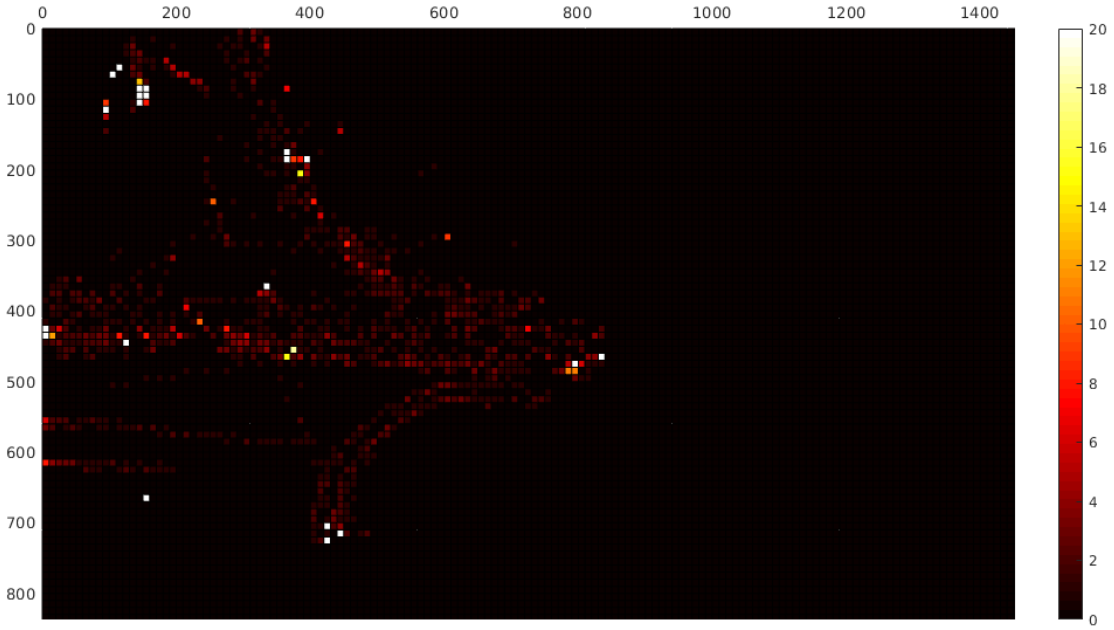
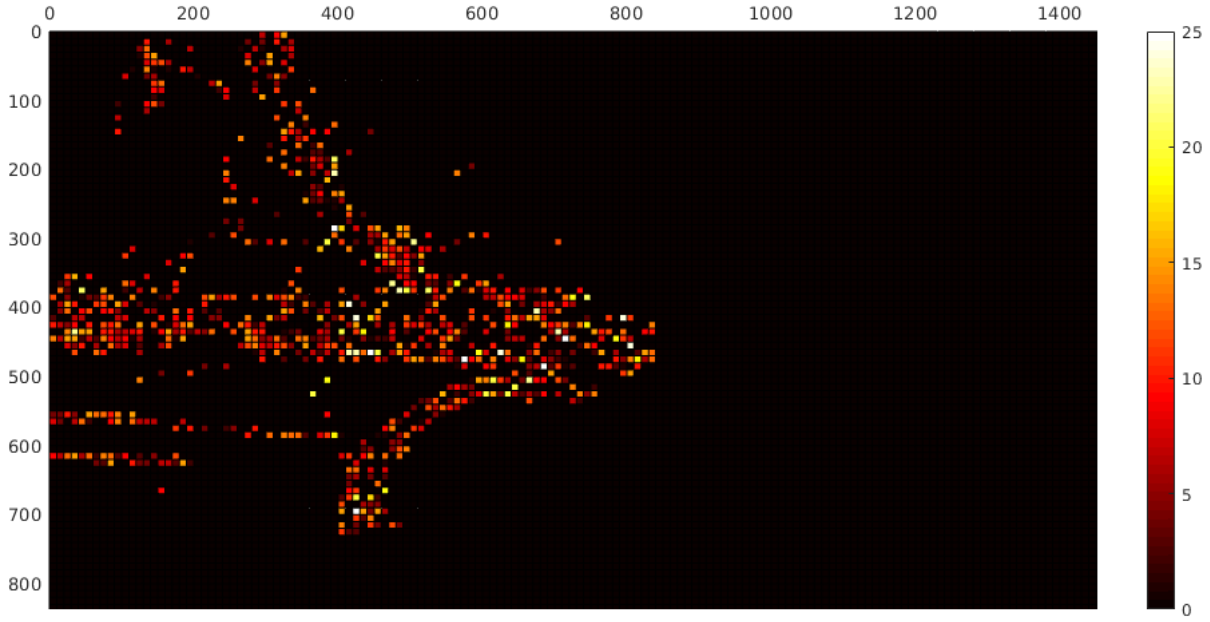
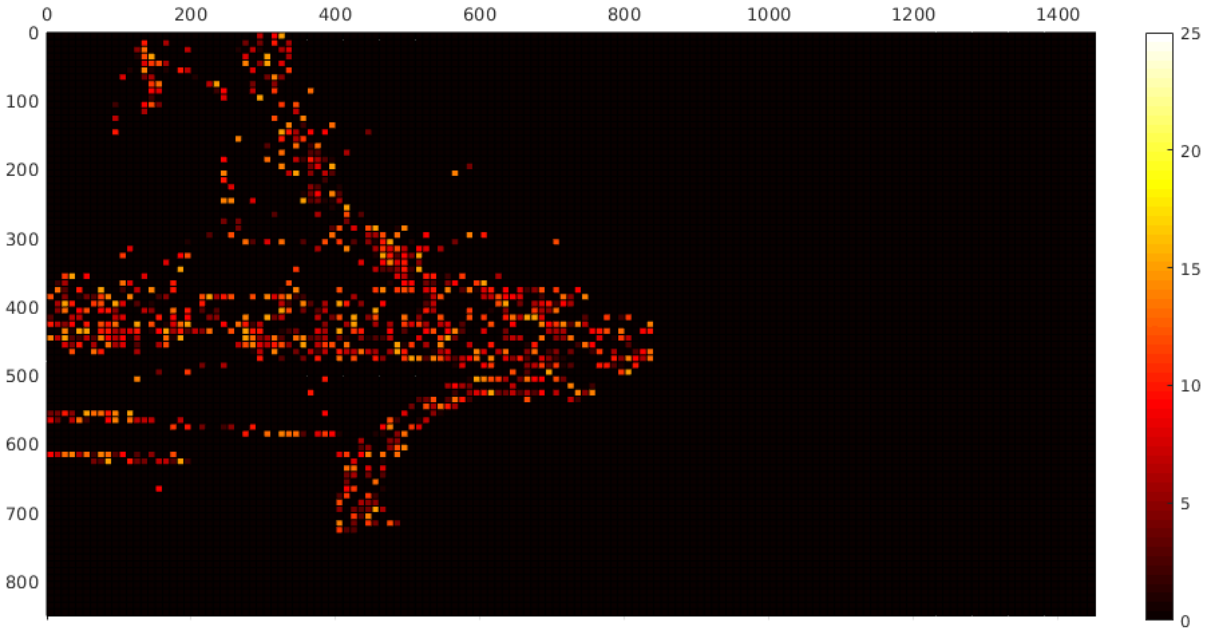


Figure 4.8: Heatmap of vehicles positions throughout the tracking.

The maximum and average velocities locally are presented in fig. 4.9(a) and fig. 4.9(b), respectively. The velocities were calculated based on the scale of the Google Maps and the time difference between frames is 1/15 seconds. All velocities displayed are based in meter per second. The first velocity heatmap displays the maximum velocity, in metres per second (m/s), reached in each section of the map. The higher values are more concentrated in the middle of the crossroad with some hotspots between 20 to 25 m/s. This makes sense as it will be the zone with more movement, therefore higher chance of speeding. The second velocity heatmap displays the average velocity, in m/s, in each section of the map. In this one, the majority of velocities displayed are on a lower range of 7 to 15 m/s. These are approximately equivalent to 25 to 54 kilometres per hour (km/h) and are within the expected values since it is an average velocity in a crossroad scenario, within a zone of 50 km/h speed limit (legal speed limit inside Lisbon).



(a) Heatmap of maximum velocity, in m/s, obtained throughout the tracking.



(b) Heatmap of average, in m/s, velocity by region.

Figure 4.9: Velocity heatmaps of experimental results.

Lastly, fig. 4.10 shows the average velocity, in m/s, for each of the detected and tracked vehicles. Around 70% of vehicles' average velocity is in the range of 6 to 10 m/s, that is approximately 20 to 36 km/h. Once again, since it is an average and it is in the context of a crossroad with traffic lights, these values make sense as most vehicle would have not been moving at some point waiting for the green light.

All these metrics show how this system could be helpful for traffic management, as it enables the extraction of metrics for the traffic in one area, allowing for a better understanding of how the traffic flows. This would provide valuable information for future city planning and public transportation reconfiguration, making streets safer and freer.

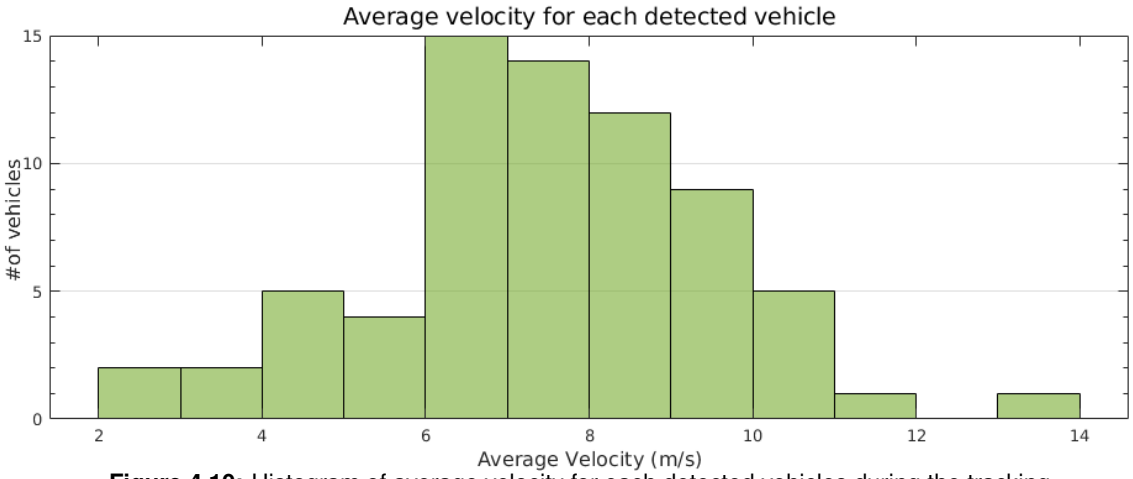
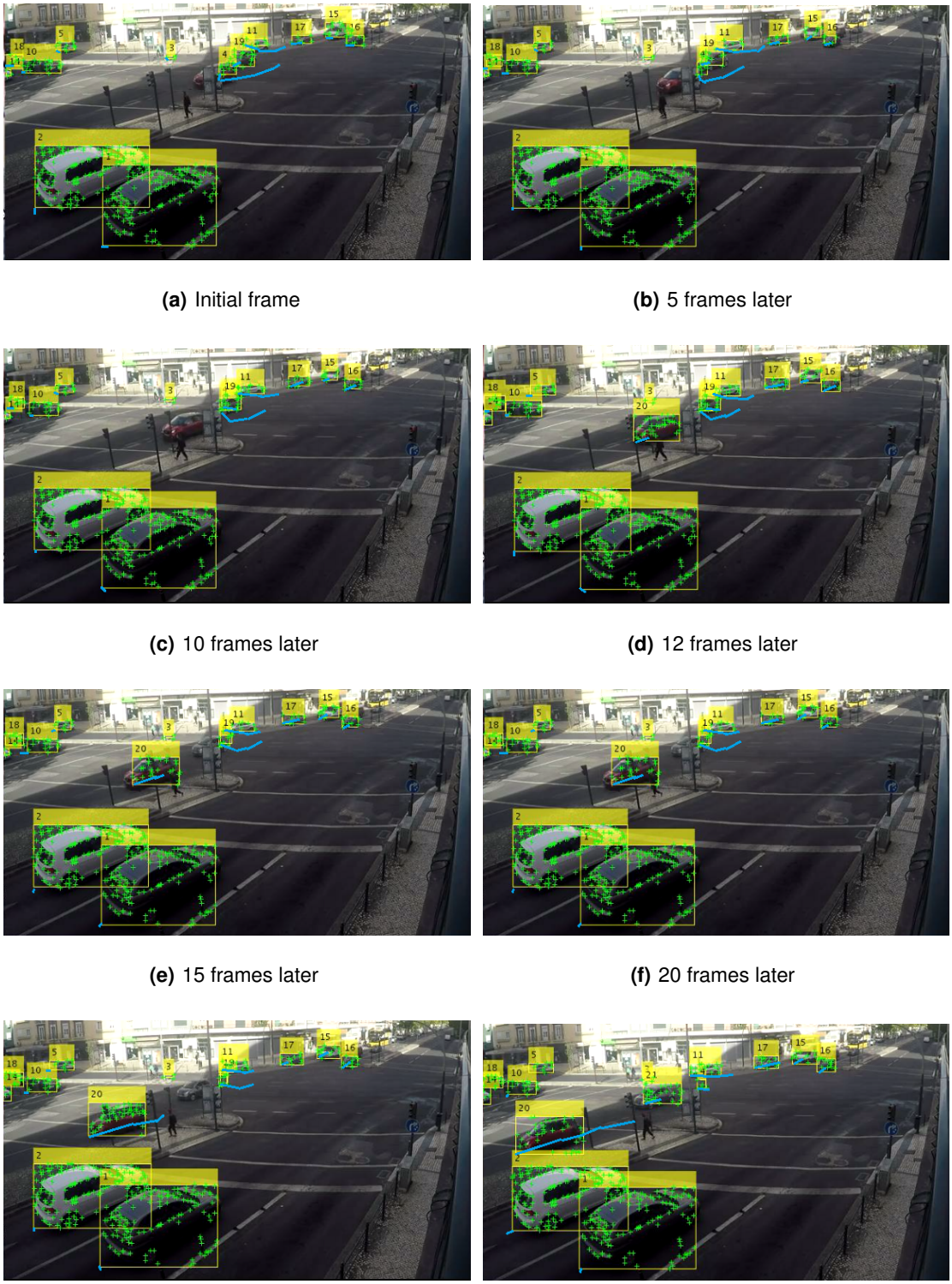


Figure 4.10: Histogram of average velocity for each detected vehicles during the tracking.

4.3.3 Failure Mode

For every technology developed there are always cases where it will not work. For this work, an important point to mention is that using the KLT tracker helps counteract false negatives in the detection phase although it still can fail if the tracker fails. An example of this can be seen in fig. 4.11. Initially, the system can track the partially occluded vehicle with id 4, fig. 4.11(a)-fig. 4.11(b), and displays the bounding box based on the previously tracked features. Since the tracker in fig. 4.11(b) only has tracked features in the back part of the vehicle, when in the next frame, fig. 4.11(c), the back portion of the car is occluded, the previous features are not tracked into this frame. When YOLOv3 detects the vehicle again in fig. 4.11(d) there are no previously tracked features therefore the system considers this a new vehicle. For the following frames, the tracking of this "new" vehicles goes smoothly.



(a) Initial frame

(b) 5 frames later

(c) 10 frames later

(d) 12 frames later

(e) 15 frames later

(f) 20 frames later

(g) 22 frames later

(h) 25 frames later

Figure 4.11: Experimental results - tracker and detector fail

5

Conclusions

Contents

”Corona didn’t get me so in the end I had to finish this!” - Ancient sage about the great quarantine, circa 2020

”Corona did not beat me! Neither did my advisor nor my thesis ! I beat them all, over the finish line ... and it felt so good!!!” - A lesson students take for life and becomes a sweet memory many years past by anonymous former student and currently an advisor

5.1 Lessons Learned and Final Remarks

The major objective of this work is to track the moving vehicles (cars) on the road. Various concepts of deep learning and computer vision have been utilised for this purpose. The track by detection framework was applied for multiple vehicle tracking. The YOLOv3 object detection system was used to detect the vehicles and the concept of the KLT algorithm was applied for tracking. By combining object detection with feature tracking, the proposed system enhances the tracking performance by reducing the number of identity switches.

The proposed approach solves the problems with applying either object detection or feature tracking by themselves. If using only feature tracking, the system would fail since features would eventually get lost. On the other hand, the movement of the vehicles is not ideal for a KLT tracker. This is because vehicles turning will not preserve many features and those coming closer or further away from the camera will suffer big feature deformation. Having said that, while YOLO is a fast and effective detector if the system relies exclusively on YOLO, it would divide many tracks when detection failed. YOLO has high precision but only 77% of recall, in the case of this thesis. This means that in 23% of the cases, YOLO will not detect a vehicle. By combining it with the KLT, the system does not need YOLO to detect in every frame since the KLT will track features of previous detections. Combining both methods also improves the feature tracking phase since it will give it an equivalent of a ground truth to refresh features and compensate for the scaling and deformation.

This work benchmark is that, for every completed track, YOLO missed detection on 24% of cases. Taking this into consideration, one can see that this system performs better for the used dataset. In cases where big images with high resolution are used, YOLO will have good results and this system will not bring a big improvement. However, when using smaller images, YOLO is not enough and this system will have better results of tracking. Surveillance systems and traffic cameras usually have a lower resolution which would make YOLO faster but less useful. In these cases, this system would be the answer since it complements YOLO’s detections and allows for continuous tracking.

To summarise, the pros and cons of the proposed system are as follow:

- Pros

    1. Fast-tracking due to the use of a fast and effective object detector.

2. High accuracy and reduced identity switches.
3. Good performance in lower resolution videos, which is particularly useful in traffic management.
4. Good capacity to deal with small occlusion.

- Cons

1. When dealing with big images, the system does not bring significant advantages compared to using just YOLO.
2. The effectiveness of the algorithm is reduced if the bounding boxes from the object detector are too big because too much background information is captured in the features.
3. Will fail if prolonged occlusion happens.

5.2 Future work

For future work it would be interesting to improve the merging of tracks for a more optimal final trajectory. As shown in fig. 4.11 the tracker occasionally is not able to track features for long enough to match with a future detection and, thus, considers the appearance of new vehicles in the middle of the frame. One way of improving would be to impose constraints on the specific areas of the frame in which vehicle entry or exit is possible. This would not allow new vehicles to just appear in the middle of the frame. Another approach could be tracking the features to frames further ahead instead of a tracking only to the frame immediately after. By tracking the features onto a frame 5 frames ahead, for example, it would allow the system more flexibility when periods of occlusion happen.

Bibliography

[1] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, 4 2021.

[2] A. Barbu, A. Michaux, S. Narayanaswamy, and J. M. Siskind, "Simultaneous object detection, tracking, and event recognition," *Advances in Cognitive Systems*, vol. 2, pp. 203–220, 4 2012. [Online]. Available: http://arxiv.org/abs/1204.2741

[3] A. Ur-Rehman, S. M. Naqvi, L. Mihaylova, and J. A. Chambers, "Multi-target tracking and occlusion handling with learned variational bayesian clusters and a social force model," *IEEE Transactions on Signal Processing*, vol. 64, pp. 1320–1335, 3 2016.

[4] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav, L. He, Y. Zhang, S. Yan, J. Yang, G. Fernández, A. Hauptmann, A. Memarmoghadam, Á. García-Martín, A. Robinson, A. Varfolomieiev, A. H. Gebrehiwot, B. Uzun, B. Yan, B. Li, C. Qian, C.-Y. Tsai, C. Micheloni, D. Wang, F. Wang, F. Xie, F. J. Lawin, F. Gustafsson, G. L. Foresti, G. Bhat, G. Chen, H. Ling, H. Zhang, H. Cevikalp, H. Zhao, H. Bai, H. C. Kuchibhotla, H. Saribas, H. Fan, H. Ghanei-Yakhdan, H. Li, H. Peng, H. Lu, H. Li, J. Khaghani, J. Bescos, J. Li, J. Fu, J. Yu, J. Xu, J. Kittler, J. Yin, J. Lee, K. Yu, K. Liu, K. Yang, K. Dai, L. Cheng, L. Zhang, L. Wang, L. Wang, L. Van Gool, L. Bertinetto, M. Dunnhofer, M. Cheng, M. M. Dasari, N. Wang, N. Wang, P. Zhang, P. H. S. Torr, Q. Wang, R. Timofte, R. K. S. Gorthi, S. Choi, S. M. Marvasti-Zadeh, S. Zhao, S. Kasaei, S. Qiu, S. Chen, T. B. Schön, T. Xu, W. Lu, W. Hu, W. Zhou, X. Qiu, X. Ke, X.-J. Wu, X. Zhang, X. Yang, X. Zhu, Y. Jiang, Y. Wang, Y. Chen, Y. Ye, Y. Li, Y. Yao, Y. Lee, Y. Gu, Z. Wang, Z. Tang, Z.-H. Feng, Z. Mai, Z. Zhang, Z. Wu, and Z. Ma, "The eighth visual object tracking vot2020 challenge results," in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds. Cham: Springer International Publishing, 2020, pp. 547–601.

[5] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv:2003.09003[cs]*, Mar. 2020, arXiv: 2003.09003. [Online]. Available: http://arxiv.org/abs/1906.04567

[6] Y. Xin, J. Hou, L. Dong, and L. Ding, “A self-adaptive optical flow method for the moving object detection in the video sequences,” *Optik*, vol. 125, 10 2014.

[7] Z. Wang, X. Sun, W. Diao, Y. Zhang, M. Yan, and L. Lan, “Ground moving target indication based on optical flow in single-channel sar,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, 7 2019.

[8] Z. Xu, D. Zhang, and L. Du, “Moving object detection based on improved three frame difference and background subtraction,” in *2017 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*. IEEE, 12 2017, pp. 79–82.

[9] Z. Zhong, B. Zhang, G. Lu, Y. Zhao, and Y. Xu, “An adaptive background modeling method for foreground segmentation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, 5 2017.

[10] S. Jiang and X. Lu, “Wesambe: A weight-sample-based method for background subtraction,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, 9 2018.

[11] G. Shi, J. Suo, C. Liu, K. Wan, and X. Lv, “Moving target detection algorithm in image sequences based on edge detection and frame difference,” in *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE, 10 2017, pp. 740–744.

[12] W. Wang, J. Shen, and L. Shao, “Video salient object detection via fully convolutional networks,” *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 38–49, 2018.

[13] X. Ou, P. Yan, Y. Zhang, B. Tu, G. Zhang, J. Wu, and W. Li, “Moving object detection method via resnet-18 with encoder-decoder structure in complex scenes,” *IEEE Access*, vol. 7, pp. 108 152–108 160, 2019.

[14] P. W. Patil and S. Murala, “Msfgnet: A novel compact end-to-end deep network for moving object detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4066–4077, 2019.

[15] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer London, 2011. [Online]. Available: http://szeliski.org/Book/

[16] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[17] H. Bay, T. Tuytelaars, and L. V. Gool, “Surf: Speeded up robust features,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, 2006, pp. 404–417.

[18] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, vol. 2, 2005, pp. 1508–1515 Vol. 2.

[19] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and Vision Computing*, vol. 22, 9 2004.

[20] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, 2005, pp. 886–893 vol. 1. [Online]. Available: http://lear.inrialpes.fr

[21] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[22] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[23] F.-F. Li, J. Johnson, and S. Yeung, “Cornell computer vision lecture notes,” 2020. [Online]. Available: http://www.cs.cornell.edu/courses/cs5670/2020sp/lectures/lec21_cnns_for_web.pdf

[24] H. Sajid and S.-C. S. Cheung, “Universal multimode background subtraction,” *IEEE Transactions on Image Processing*, vol. 26, 7 2017.

[25] T. S. Haines and T. Xiang, “Background subtraction with dirichletprocess mixture models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, 4 2014.

[26] Z. Bian and X. Dong, “Moving object detection based on improved gaussian mixture model,” in *2012 5th International Congress on Image and Signal Processing*. IEEE, 10 2012, pp. 109–112.

[27] H. Santosh, P. Venkatesh, P. Poornesh, and N. Rao, “Tracking multiple moving objects using gaussian mixture model,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, 5 2013. [Online]. Available: https://www.researchgate.net/publication/305709395

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, pp. 84–90, 5 2012. [Online]. Available: http://code.google.com/p/cuda-convnet/

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 6 2014, pp. 580–587.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *2015 IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 6 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10578-9_23

[31] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 4 2015, pp. 1440–1448. [Online]. Available: http://arxiv.org/abs/1504.08083

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2016, pp. 779–788. [Online]. Available: https://arxiv.org/abs/1506.02640

[35] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 7 2017, pp. 6517–6525. [Online]. Available: https://arxiv.org/abs/1612.08242

[36] J. Redmon, "Yolov3: An incremental improvement," University of Washington, Tech. Rep., 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[37] D. Meyer, J. Denzler, and H. Niemann, "Model based extraction of articulated objects in image sequences for gait analysis," in *Proceedings of International Conference on Image Processing*. IEEE Comput. Soc, 1997, pp. 78–81 vol.3.

[38] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*. British Machine Vision Association and Society for Pattern Recognition, 4 1988, pp. 147–151.

[39] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, 1981. [Online]. Available: https://www.researchgate.net/publication/215458777

[40] C. Tomasi and T. Kanade, "Detection and tracking of point features," International Journal of Computer Vision, Tech. Rep., 1991.

[41] J. Shi and Tomasi, “Good features to track,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. IEEE Comput. Soc. Press, 1994.

[42] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. [Online]. Available: http://arxiv.org/abs/1409.4842

[43] A. Kathuria, “What’s new in yolo v3?” 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

[44] Joseph and A. R. Farhadi, “Yolo: Real-time object detection.” [Online]. Available: https://pjreddie.com/darknet/yolo/