Vehicle tracking in urban environment

Sofia Teodoro Bebiano
sofia.teodoro.bebiano@tecnico.ulisboa.pt
Instituto Superior Técnico, Lisboa, Portugal

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. 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.

I. Introduction

I-A. 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.

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.

I-B. State-of-the-art

Object detection and tracking represent one of the main challenges in the field of 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.

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]. DFT needs to manually initialise 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.

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. 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.

Detection

The purpose of object detection lies in obtaining an object's position and classification in an image. Generic object detection remains a complex task, since it is difficult to design a detector that can successfully detect targets in multiple and different scenarios. A detection process can be done by extracting and processing certain image properties: area, contours, shape, blob radius and centre, among others [2]. 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. The aim is to model objects of interest based on these extracted features and not in raw pixels. First, start with the most basic information about an image, the raw pixels, and try to extract more meaningful information from it. 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.

Deep Convolution Neural Networks (DCNNs) for object detection and classification have attracted a lot of interest in recent years. With their ability to learn parameters themselves, a higher degree of accuracy can be achieved. Girshick *et al.* [3] suggested the concept of object detection using a Convolutional Neural Network (CNN). Girshick *et al.* used the Region Proposal (RP) method and proposed Regions with CNN (RCNN). Given its the slow detection speed, He *et al.* proposed Spatial Pyramid Pooling in deep convolutional Networks for the visual recognition network (SPPNet) [4]. He *et al.* also suggested Fast RCNN [5]. 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, based on the Fast RCNN that resulted in the Faster RCNN [6]. 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. Redmon *et al.* suggested a new method for detecting objects called You Only Look Once (YOLO) [7]. The RP phase was totally dropped and a single convolutional network was used as seen in 1. Removing the RP step enhances detection efficiency. YOLO can detect the objects while operating around 45 fps in real-time. YOLOv2 [8] and YOLOv3 [9] 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.

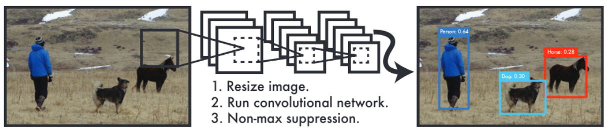


Fig. 1: The YOLO Detection System. The system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence. Source: [7].

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.* [10] 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.

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 [11] and SIFT [12] 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 method proposed in [13] and later improved in [14]. 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 [15] to deal with the issue of selecting features that can be tracked well. The Kanade-Lucas-Tomasi 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.

II. Background and Proposed Approach

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 2, 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 2, prevents the correct association to the correct vehicle in the third frame.

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 2 where the identification of each bounding box matches the identification on the first frame.

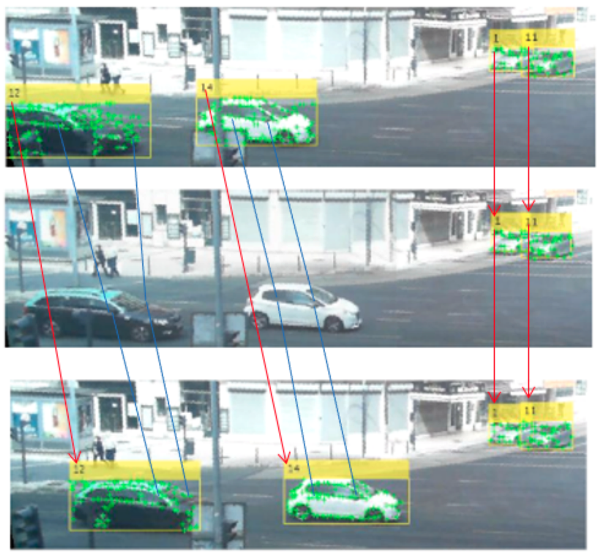


Fig. 2: Example of the problem and proposed solution. Top image - Object detector and feature selection identify vehicles and image features inside the bounding box. Middle image - object detectors often miss bluntly. However feature trackers can still match corresponding points (blue line). Bottom image- if objects are detected again, the feature tracker allows the association of the correct object (red lines).

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: 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

II-A. Object detection - You Only Look Once

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: region proposal-based method represented by RCNN such as Fast-RCNN, Faster-RCNN, among others; 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 follow: Pre-train a CNN network on the image classification task. 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. The final layer of the pre-trained CNN is modified to output a prediction tensor of size $S{*}S{*}(B{*}5{+}C)$, 3 ilustrates YOLO detection model.

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

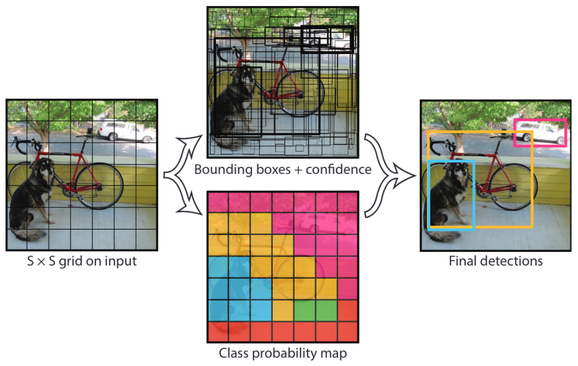


Fig. 3: YOLO system models detection as a regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes, confidence for those boxes,and C class probabilities. Source: [7].

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. YOLOv2 runs k-means clustering on training data to find good priors on anchor box dimensions. 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, 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 4. 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.

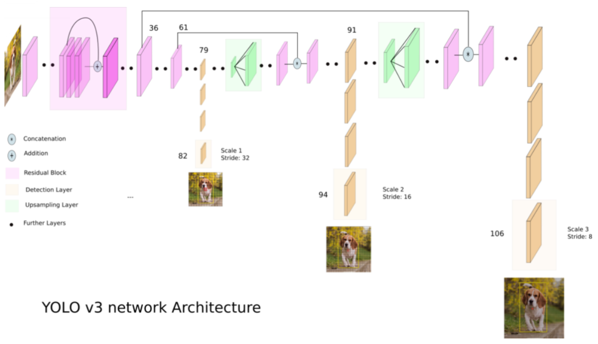


Fig. 4: YOLOv3 architeture. Source: [16]

II-B. Prediciton and Tracking: The Lucas Kanade 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.

The Lucas-Kanade tracker (KLT) is a feature tracker technique firstly proposed by Lucas and Kanade in the 1980's [13], improved in 1991 by Tomasi and Kanade [14] and in 1994 by Shi and Tomasi that included a keypoint selection method [15]. 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.

In this work, an implementation in MATLAB2018a was used. The vision.PointTracker tool is applied. The point tracker object tracks a set of points using the KLT feature-tracking algorithm. The point tracker is often used for short-term tracking as part of a larger tracking framework which is the case in this work. 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 points are reacquired for every detection.

II-C. 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 [15]. 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.

II-D. 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 5.

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 1, where homogeneous coordinates of the corresponding points are x' and x.

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

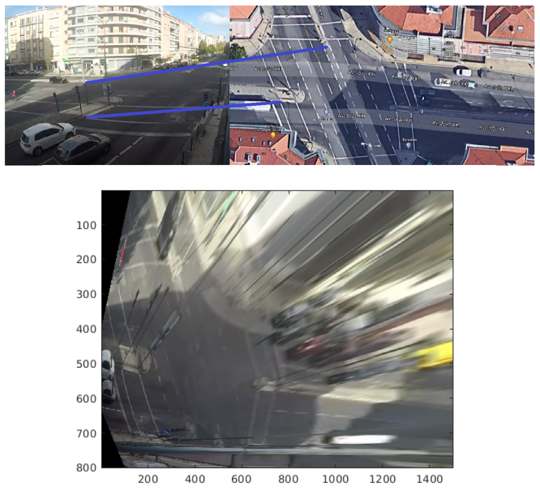


Fig. 5: Example of projection from camera perspective into a bird's eye perspective. The bottom image shows the blurring on the scene further away from the camera. The homography is computed by mapping at least 4 corresponding points in both images.

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

$$h_{33} = 1 \tag{2}$$

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

$$h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{21}^2 + h_{22}^2 + h_{23}^2 + h_{31}^2 + h_{32}^2 + h_{33}^2 = 1 \tag{3}$$

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.

III. Implementation of the Detection and Tracking Pipeline

In the object detection phase, the framework YOLOv3 available on Redmon's webpage was used [17]. 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 during the tracking. Using MATLAB function $cp2tform$ 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 from YOLOv3. 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.

III-A. Evaluation

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

- False positive - detections that are incorrect
- True positive - correct detections
- False negative - vehicles that were not detected

To assess YOLOv3 performance we computed key indicators such as precision, 4 and recall, 5, in a specific dataset created "in-house". Subsequently, KLT parameters were tuned and the tracker performance evaluated.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5}$$

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 recall and precision calculated for our dataset is presented in table I.

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

	Precision	Recall
original size	97.9%	77.9%
90% size	95.2%	80.7%
80% size	95.9%	77.5%
70% size	90.7%	52.5%
60% size	96.5%	45.5%

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

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. A higher number of pyramid levels allows the algorithm to handle larger displacements 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, 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. 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 average results presented in table II.

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

	average
#detected features initially	406.8
# of features tracked until frame 25	375.1
# of features tracked until frame 50	366.5
# of features tracked until frame 75	360
# of features tracked until frame 100	355.1
% of features tracked until frame 100	88.8%

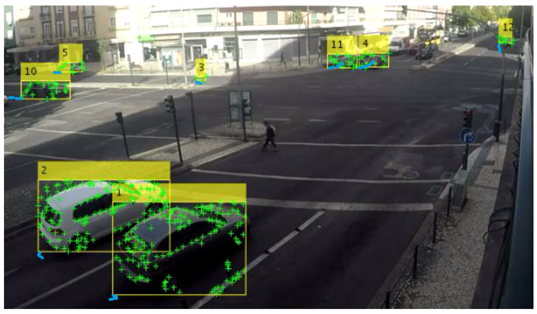
Features were tracked throughout most of the frames, with an average of 88.8% being tracked until the end. Features will inevitably be lost, either due to being weaker or because they leave the frame. Since a vehicle takes around 100 frames to enter and leave the frame, most of the features lost will probably either correspond to vehicles leaving frame or to weaker features not being reliably tracked.

Bounding boxes association

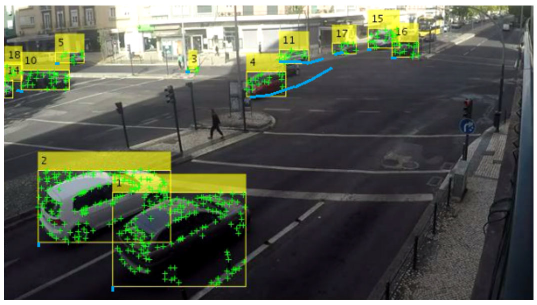
For the KLT tracker phase it was important to find a balance for the matching of YOLOv3 bounding boxes with the predicted bounding boxes from KLT. This matching is based on the area of the intersection between both boxes being at least 40% of the area of the box from YOLOv3. When none of the detected 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% a good level of matching is obtained even when bounding boxes with significant overlap exist.

IV. Results

The final tracking results of this system were obtained using a real traffic scenario video. The tracking results concerning the camera view are represented in 6, while the ones concerning the map perspective are illustrated in 7. The trajectories displayed in 7 correspond to the last detected positions for each vehicle, projected onto a map view from Google Maps.

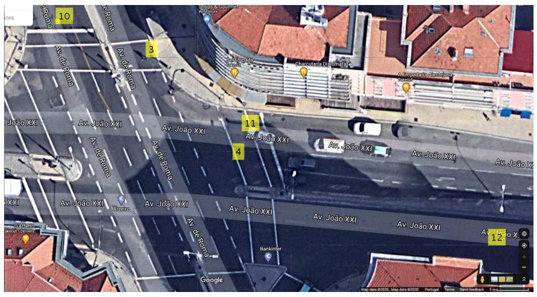


(a) Initial frame

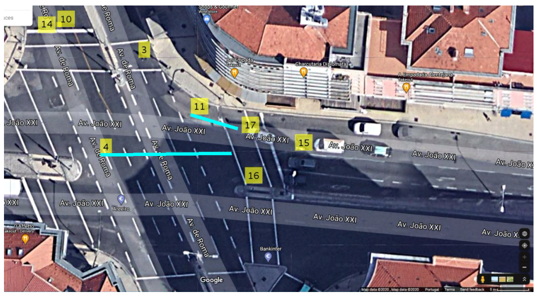


(b) 30 frames later

Fig. 6: Experimental results - camera view.



(a) Initial frame



(b) 30 frames later

Fig. 7: Experimental results - map view.

Benchmark

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, 6 and 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, even when YOLOv3 lacks detection on an intermediate frame.

IV-A. Analytics

Using all the tracking data, metrics about occupation and velocity were calculated. In 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.

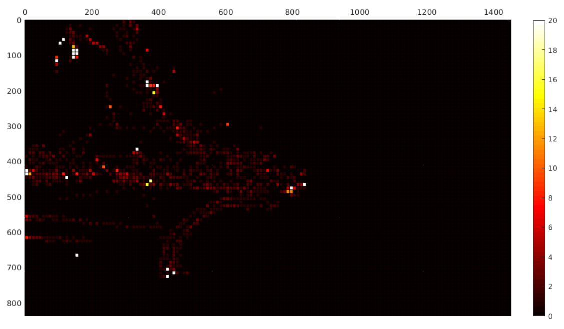
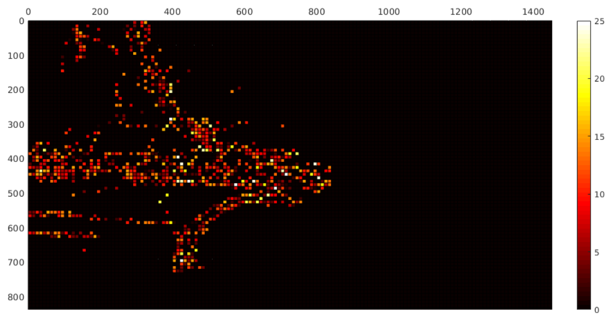
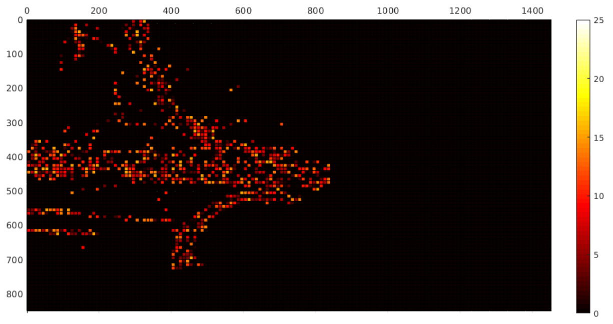


Fig. 8: Heatmap of vehicles positions throughout the tracking.

The maximum and average velocities locally are presented in 9(a) and 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. In 9(a) displays the maximum velocity 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. As it will be the zone with more movement, higher the chance of speeding. In 9(b) displays the average velocity 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 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.

Fig. 9: Velocity heatmaps of experimental results.

Lastly, 10 shows the average velocity 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.

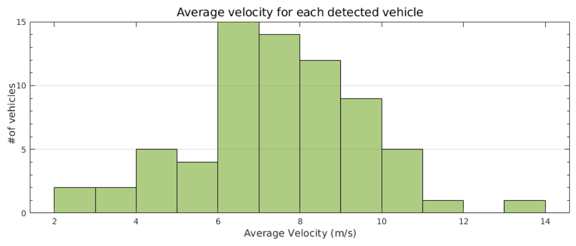


Fig. 10: Histogram of average velocity for each detected vehicles during the tracking.

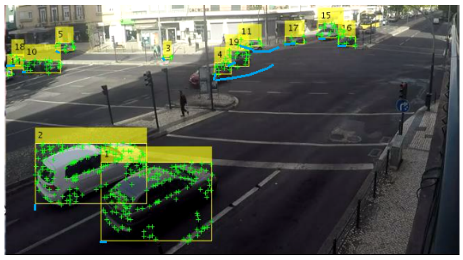
IV-B. Failure Mode

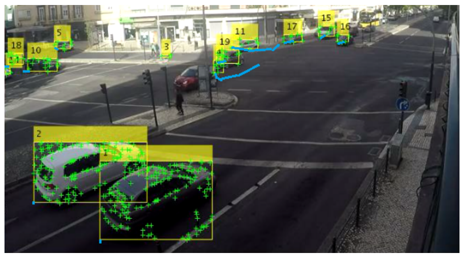
For every technology developed there are always cases where it will not work. For this work, an important point to mention is that using the KLT tracker helps counteract false negatives in the detection phase although it still can fail if the tracker fails. An example of this can be seen in 11. Initially, the system can track the partially occluded vehicle with id 4, 11(a)-11(b), and displays the bounding box based on the previously tracked features. Since the tracker in 11(b) only has tracked features in the back part of the vehicle, when in the next frame, 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 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.

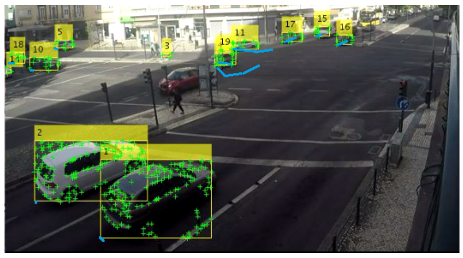
V. Conclusions

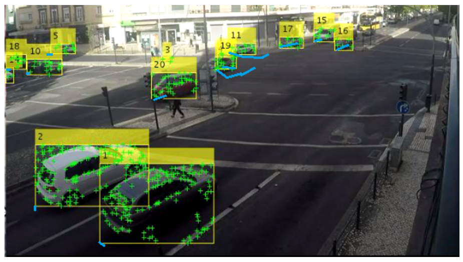
V-A. Lessons Learned and Final Remarks

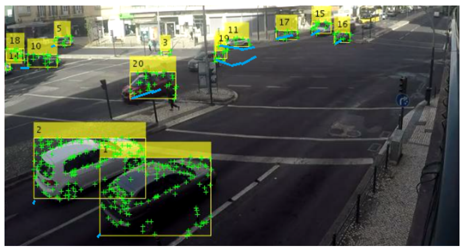
The major objective of this work is to track the moving vehicles on the road. The track by detection framework was applied for multiple vehicle tracking. The YOLOv3 object detection system was used to detect the vehicles and the concept of the KLT algorithm was applied for tracking. By combining object detection with feature tracking, the proposed system enhances the tracking performance by reducing the number of identity switches. 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. 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. Taking into consideration, for every completed track, YOLO missed detection on 24% of cases, 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.


(a) Initial frame


(b) 5 frames later


(c) 10 frames later


(d) 12 frames later


(e) 15 frames later

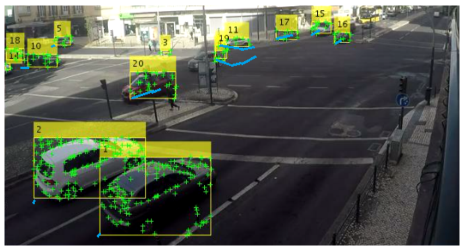

(f) 20 frames later

Fig. 11: Experimental results - tracker and detector fail

V-B. Future work

For future work it would be interesting to improve the merging of tracks for a more optimal final trajectory. As shown in 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.

Acknowledgements

I would like to thank my mom, my supervisors and my pets.

References

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

[3] 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.

[4] 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.

[5] 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

[6] 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

[7] 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

[8] 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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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

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

[15] 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.

[16] A. Kathuria, "What's new in yolo v3?" 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

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