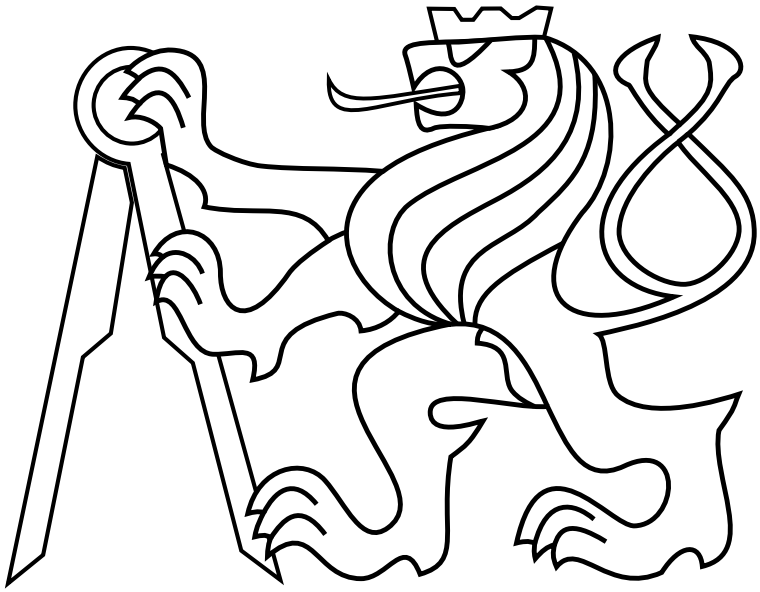
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Jan Bouček

Tracking vehicles across multiple non-overlapping fisheye cameras in a city environment

Department of Cybernetics

Thesis supervisor: **Ing. Michal Reinštein, Ph.D.**

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and I have listed all sources of information used within in the accordance with the methodical instructions for observing the ethical principles in the preparation of university thesis.

Prague, date..... .....

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bouček** Jméno: **Jan** Osobní číslo: **425059**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sledování vozidel přes více nepřekrývajících se fisheye kamer v městském prostředí

Název diplomové práce anglicky:

Tracking vehicles across multiple non-overlapping fisheye cameras in a city environment

Pokyny pro vypracování:

The task is to design, implement and experimentally evaluate a deep neural network based solution for tracking vehicles across multiple non-overlapping fisheye cameras in a city environment. The proposed solution should include creation of training, validation, and testing datasets, and thorough experimental evaluation of the proposed architecture with respect to the state-of-the-art methods. Instructions are as follows:
1. Explore the current state-of-the-art solutions of video detection and tracking vehicles from multiple cameras.
2. Design a new approach for vehicle detection and tracking with known location of the cameras with fisheye lenses and non-overlapping views.
3. Using data provided by GoodVision s.r.o., implement the proposed approach in TensorFlow deep learning framework.
4. Evaluate the solution and compare it to the state-of-the-art methods on real world scenarios.

Seznam doporučené literatury:

[1] Goodfellow, Ian, et al. ?Deep Learning?, MIT Press, 2016
[2] Liu, Wei, et al. "SSD: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.
[3] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
[4] Szegedy, Christian, et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." AAAI. 2017. APA
[5] He, Kaiming, et al. "Mask R-CNN" arXiv preprint arXiv:1703.06870 (2017).
[6] Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." Proceedings of the IEEE International Conference on Computer Vision. 2015.
[7] Abadi, Mart?n, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." Software available from tensorflow. org.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Michal Reinštein, Ph.D., vidění pro roboty a autonomní systémy FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **17.01.2018** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2019**

Ing. Michal Reinštein, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

Acknowledgements

I would like to thank my thesis supervisor Ing. Michal Reinštein Ph.D for his great support, leadership and expertise that helped greatly during this thesis. I would also like to thank the Good Vision company for great cooperation. Finally, I thank my friends and family for their support during my whole studies.

Abstract

This thesis deals with tracking vehicles over multiple non-overlapping cameras. The goal is to design and implement a system able to detect vehicles in a city environment and track their position. The cameras have a fish-eye view and are mounted to street lamps. We present a deep neural network for vehicle detection utilizing the video information, a single camera vehicle tracking algorithm based on optical flow, a deep neural network trained to compute similarities between vehicles and a probabilistic graph representation of a city. The conducted real world experiments verified the capability of the whole system.

Keywords: Object detection, Multi camera tracking, fisheye cameras.

Abstrakt

Tato práce se zabývá sledováním vozidel pomocí více nepřekrývajících se kamer. Cílem je návrh a realizace systému, který je schopný rozpoznat vozidla v městském prostředí a sledovat jejich polohu. Kamery mají objektiv typu rybí oko a jsou umístěny v pouličních lampách. Představujeme hlubokou neuronovou síť pro detekci vozidel využívající informace z videa, algoritmus pro sledování vozidel na jedné kameře založený na optical flow, hlubokou neuronovou síť pro počítání podobností mezi vozidly a pravděpodobnostní grafovou reprezentaci města. Provedené experimenty reálného světa ověřily schopnosti celého systému.

Klíčová slova: Detekce objektů, sledování přes více kamer, objektiv rybí oko

Contents

List of Figures

1 Introduction

Estimated 80% of the world data is in form of images or videos [44] and the percentage will likely increase. There is a lot of useful information hidden in videos, but it is very hard to extract it. The vast majority of the videos is processed manually by people, who make mistakes and are expensive. There is an incredible need for automated video processing in many branches of the industry for decreasing cost and for increasing speed and accuracy. Being able to accurately detect and track vehicles can provide valuable data about transportation to governments. Reidentification and tracking objects over multiple cameras in real time can help reinforcement agencies to effectively fight crime or surveillance agencies to prevent intrusion.

The computer vision field has been experiencing an incredible advancement in the last couple of years thanks to the introduction of convolutional neural networks. They are successful on many problems from image classification to object detection and tracking. The state of the art in deep learning in computer vision was explored in this thesis.

This thesis was developed in cooperation with the company Good Vision s.r.o [1] for a law enforcement company from Brazil, to develop and deploy smart city solutions in South America. Good Vision provides a smart video analysis from street cameras , while the Brazilian partner provides the infrastructure. This thesis developed a multi-camera tracking of a vehicle in a city, that will be used by the police in many South American cities.

When a crime is committed and a suspect drives away, there is a need for an automatic tracking of the vehicle. This is a difficult task and can easily fail when performed by people. This thesis introduces an automatic approach based on artificial intelligence and deep learning allowing fast and reliable tracking of a vehicle in a city.

The difference from standard setups is that the cameras are fisheye and they are mounted directly into street lamps above the vehicles as shown in the figure 1.1. There are no public datasets for these kinds of images or videos and custom datasets had to be created from videos provided by the Brazilian partner.

For the final algorithm to be accurate, different subproblems had to be solved separately. That involves object detection, tracking, similarity, and reidentification as well as two datasets generation.

A deep learning object detector was introduced. It is based on the SSD [91], but utilizes the information from video by temporal difference and feeds it as an additional input layer. Experiments show, that the introduced approach achieves 91.6% mAP on the presented domain, which is far better than the state of the art SSD network with just 63.2% mAP when trained and tested on the same data.

Deep convolutional neural network Facenet [124] was successfully trained to recognize similar vehicles, which was used for reidentification a vehicle on a different camera. The overall algorithm was tested on a real-world scenario and can re-identify a vehicle with the 88 % probability.

1.1 Problem statement

Figure 1.1: The fisheye camera is mounted in a lamp, therefore the view is directly from above.

The smart city project is in development and only several cameras have been mounted. The cameras were built directly to street lamps and their streams are sent to a server. The cameras have a 360 degrees view thanks to their very short focal lengths and their locations in the city are known.

When there is a crime committed and the suspect is driving away in a vehicle, a camera operator marks the car and the goal of the project is to detect the vehicle, track its position in a single camera and be able to recognize it on different cameras.

The camera resolution and optics don't allow using license plate recognition for reidentification.

1.2 Overview of methodology

The whole thesis was divided to subproblems and solved more or less separately. These solutions were connected into one for the final experiments.

- A thorough state of the art analysis was performed in the section 2 to be able to select

the best approaches for different subproblems and to be able to compare developed solution to other approaches.

- The camera parameters were not known and a mathematical model of the camera lens had to be created in section 3 based on calibration data. This section also solves the transformations between the real world coordinates and their projection in the frame, which allows accurate localization of the vehicle.
- The possibility of distributed computing directly in the cameras was explored as described in the section 4. That included fast set of algorithms for detection [109], tracking [7] and classification [52] running on CPU. This approach could not be used directly because of a bad performance on high traffic scenes. However, it was used for a semi-supervised dataset generation and allowed training of a network for computing similarity between vehicles.
- An annotation tool was used as described in the section 6.2 for creating training and validation dataset for object detection.
- SSD [91] neural network architecture was selected for vehicle detection because of its state of the art performance. It was extended for an additional input layer of temporal difference and additional feature layer to better recognize small moving objects and were trained on NVIDIA GeForce GTX 1080 for four days.
- The original and the improved SSD neural networks were trained on the same dataset and their accuracies were compared in the section 8.2.
- Google Facenet [124] was selected for computing similarities between vehicles and retrained on a custom dataset from section 4.
- A city and vehicles representation was based on Markov chain and introduced in the section 7.5. It utilized the object similarity and relations between cameras in reidentification.
- A real-world experiments were performed on cameras from a city and the implemented multi-camera tracking was tested and evaluated in the section 8.4.

1.3 Contribution

Problems of various types were solved in this thesis, such as:

- An improved version of SSD [91] was introduced and implemented, which achieved the mAP 91.6% compared to the SSD from [91] with the 63.2% mAP on the wide-angle domain.
- A dataset for vehicle detection on fisheye camera containing over 1600 images was created by standard annotating methods. Another dataset for training the similarity network was generated by an object detection and tracking algorithm containing over 9000 images.

- The Facenet [124] network was retrained for vehicle similarity achieving the classification accuracy of 81%.
- A mathematical probabilistic representation of the multi-camera tracking problem was introduced based on Markov chains.
- Multi-camera tracking experiments on real-world scenario were performed achieving the 88% reidentification accuracy.

2 Related work

The computer vision field has made an incredible leap forward in the last couple of years. Thanks to the increasing computational capabilities of computers and recent advancements in deep learning, we are able to do tasks, that we could not imagine. Image classification, location, and detection are tasks, that have gone through an incredible evolution in the last 5 years. The face recognition, autonomous driving, surveillance and many more fields have been the driving force for computer vision. The tracking of objects over multiple cameras is valuable in retail, traffic monitoring, and surveillance.

2.1 Classification

Image classification is a task, where given an image, one class has to be assigned to the previously known set of classes. This is a hard task because of the variance in lighting, pose, rotation, scale, as well as intraclass variation. The detection task described in the section 2.2 is linked to the classification problem, where the deep learning detectors use image classification networks.

Accuracy is measured as the proportion of correctly classified images in the test set. Two metrics are used. In top 1 accuracy, only one prediction is made. In top 5 accuracy, 5 predictions are made and an image is considered to be correctly classified if the correct class is among them.

To properly train, evaluate and compare models, several datasets, such as Mnist [85], ImageNet [33], or PASCAL VOC [38] were created.

Before the invention of convolutional neural networks, other classifiers were used. Classifiers, in general, can be divided into parametric and nonparametric methods. The nonparametric ones require no training phase and the decision is based directly on the data. Most common method is the Nearest Neighbor approach [14, 156]. The parametric methods, on the other hand, require a training phase to find the parameters of the model, which can be in form of decision tree [15], AdaBoost [107], or the most common Support Vector Machine (SVM).

The classification pipeline of SVM is such that a set of features is extracted into a vector and a SVM is applied. These features can have many different forms and can also be combined. A histogram [21], Bag of features [82, 106], SIFT features [151, 12] or Haar features [101] can be used.

Convolutional neural networks (CNN) are the state of the art in image classification. They were introduced in 1990's [83], but only only since 2012 had a great success.

In 2012 AlexNet [77] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with the top-5 error being just a 15.4%. This was a huge success compared to the second best with 26.2% top-5 error rate and this is considered to be the beginning of deep learning in computer vision.

The ZFNet [155] in 2103 introduced insides to how CNNs work by introducing deconvolutional network, that could show various feature activations. It also outperformed the AlexNet on ImageNet by the top-5 error rate being 14.8% and winning the ILSVRC in 2013.

GoogLeNet/Inception won the ILSVRC 2014 with the incredible top-5 error rate of 6.67%. The architecture was based on LeNet [84] but introduced an inception module. This module eliminates all full-connected layers, greatly reducing the number of parameters. It is used as a backbone in many object detection networks and was used in this thesis as a base network for Facenet [124] described in the section 6.5

The second best network in ILSVRC 2014 was the VGG Net [128]. The depth of the network was increased, but the number of parameters was kept low thanks to very small 3x3 filters. The architecture is simpler than of Inception and it is widely used as a detection network backbone. This network was used in this thesis as a backbone for the SSD [91] described in the section 6.2.

The Microsoft's ResNet [54] introduced a deeper architecture. They were able to train the network thanks to the introduction of the residual connections. Part of the information passes through each layer unchanged. This helps to solve the vanishing gradient problem [57]. With the top-5 error rate 3.57%, they surpassed the human accuracy winning the ILSVRC 2015.

The ResNet idea was further developed. Wide ResNet [154] reduced the number of layers while widened the network. ResNeXt [150] is furthermore highly modularized and introducing new dimension called cardinality, which is more effective, than simply increasing the number of layers or their width.

DenseNet [59] connects each layer to all previous layers. This furthermore helps with the vanishing gradient problem and reducing the number of parameters. It outperforms ResNet while requiring less memory and computation.

The task of image classification is considered to be solved, but more research is being done. These image classification networks can be used as a backbone for other tasks, such as image detection, localization or segmentation.

2.2 Object Detection

To be able to track vehicles, they need to be first detected. The most common methods use object detection from cameras. This section introduces a general vehicle detection as well as general object detection in images.

2.2.1 Vehicle detection

There are many ways how to detect vehicles, not just with cameras. One can detect changes in magnetic fields [32, 20] or use a laser scanner [43].

Cameras are the most common sensor, but they can be also combined with a laser scanner [144, 112] or a sonar [73, 142]. Sometimes a stereo vision [11, 137] can be used to gain a better model of the environment.

A lot of research was done for detection of vehicles and pedestrians thanks to the recent advancements in autonomous driving. Many datasets were created [62, 97, 96, 18] for detecting vehicles, pedestrians and other objects from the vehicle point of view. There is even a research for detecting vehicles by their shadow [138]. The state of the art in vehicle detection using cameras is detecting each image independently using techniques described in the section 2.3.

2.2.2 Object detection in computer vision

In computer vision, the object detection is a specified task. The goal is to draw a rectangle (bounding box) around each object and classify it. The accuracy is measured in mean Average Precision described in the section 8.1.

Before the introduction of neural networks, various methods were used for object detection. Haar features were used for detecting faces [52, 88, 140] and vehicles [130]. For general object detection, the background subtraction [109, 58] or optical flow [103, 114, 23] described in the sections 4.3 and 4.4. SIFT [93] HOG [46, 143, 159, 39, 31] were also used.

The big advancements came with the introduction of region proposal networks [47]. R-CNN [46] was the first to introduce this concept. It consists of two neural networks, one to propose the regions of interests and the second one to classify them. Their performance was mAP of 53.3% on PASCAL VOC 2012 dataset. This was a huge success compared to the mAP of 43.3 %[19] the year before. However, R-CNN was very slow (47 seconds to detect an image on GPU with the VGG16 [128] network), thus were far from real-time video analysis. It requires a full convolutional network forward pass for each of the around 2000 proposals.

Improved and faster version Fast R-CNN [45] achieved 68.4% on PASCAL VOC 2012 with the VGG16 network while significantly increasing speed over 200 times compared to R-CNN. This was due to sharing computations over proposals and using a single network for the feature extractor, classifier and the regressor in one network. However, the selective search for the region proposals was found to be the bottleneck for the detection process.

The Faster R-CNN focused on exactly that. The feature extractor was also used for the region proposal network making the region proposal almost cost-free. They also increased the learning speed, because only one CNN needed to be trained. Faster R-CNN with VGG16 achieved 75.9% mAP on PASCAL VOC 2012 dataset with just 7 fps on GPU. This is much closer to processing a real-time video.

The YOLO [117] performs 45 fps while achieving 63.4 mAP on VOC 2007. It splits the image in a grid and predicts only two bounding boxes and class probabilities for a square. However, it struggles with detecting more small objects close to each other and would not be a good detector for vehicles from the street camera. However, there were some improvements to this network [118, 119].

	Faster R-CNN	Fast YOLO	YOLO	SSD300	SSD512
fps	7	155	21	59	22
mAP	73.2	52.7	66.4	74.3	76.8

Table 1: Results on PASCAL VOC2007 test.

Region convolutional neural network, which create proposals and then classify them, are still too slow. The Single shot multibox detector (SSD) [91] based on [37] leaves out the region proposals completely and has a fixed number of regions. It was introduced in November 2016 and had an incredible 74.3% mAP at 59 fps on VOC 2007. This network was chosen for object detection in this thesis and was described more in detail the section 6.2.

There were lately many more architectures introduced, such as [89, 87, 30] and many more are coming.

These networks process images, but are used also for detecting video. Most common schema of detection objects in a video is, that the video is decomposed to different frames and each frame is detected independently [129]. This loses a lot of the information encoded in the video. Background subtraction [49] or optical flow [103] as described in the section 4 can be used for extracting more information from the video context.

Optical flow can be connected with a neural network [104], but the optical flow is expensive to compute, even though there is a convolutional network for optical flow estimation [127]. [70] preserves the video information by taking as an input multiple frames from the video, but this makes the model large.

A good trade-off between the network's size and preserving video information was introduced in this thesis by combining the RGB input image with the 4th channel of temporal difference between frames and feeding it to a neural network. There is a research [6, 75] using the temporal difference for detection and segmentation, but to my knowledge was not combined with deep learning. To our best knowledge, there was no published result of designing a detector capable of processing fisheye distorted images.

2.3 Object tracking

The previously described tasks process single images. Now the problem expands to a new discrete-time dimension when processing video, but for now, keeping just one video feed. The goal is to create a trajectory or a sequence of bounding boxes of an object. This task is difficult because of the changes in illumination, partial and full object occlusions and the real-time processing requirements [153]. Almost all trackers assume, that the frame rate of the video is high enough, that the movements of the objects are smooth. The approach can be divided into a dense and a sparse method.

The sparse method scans only pixels nearby the tracked objects and tries to estimate their movement. The input is a position of the object and it is tracked over upcoming frames. This is especially good for tracking one object. This method has not been chosen, since there

is a need for continuous detection of incoming vehicles and the dense method has been used. The object can be represented as a single point [67], a bounding box [27, 111, 152, 36], or a silhouette [63]. Only the changes can be registered [67] or a robust reidentification [139] can be used, which performs better with occlusions, than standard methods. A statistical representation can be connected with a Kalman filter [4] or a particle filter [158]. The movement is often estimated using sparse optical flow [67, 98]. With the recent deep learning advancements, tasks as tracking are also being solved with deep neural networks [10, 55, 48, 42, 86].

The dense methods for tracking receives a video and detections for each frame. This has the advantage, that the tracks can be created without explicitly manually selecting each object we want to track. This method has been chosen, since the vehicle must be detected from each camera. However, these approaches are more computationally complex, since they require object detection. The tracker clusters the bounding boxes into tracks. The main methods use Jaccard overlap [136, 8] and optical flow [23]. This task can be complex because of the crossing tracks as well as false positive and false negative detections [66, 36].

2.4 Reidentification

When an object leaves one camera and appears in another camera, the task is to recognize it. When positions and orientations of the cameras are not known, the location and speed of the detected objects can be used for obtaining the spatial relationships among the cameras [99]. The key to reliable reidentification is to correctly model the relationships among the cameras, as well as to find a similarity metrics of the detected objects. When the cameras overlap, the key is to accurately estimate the position of the tracked object and match them [72, 78, 157]. The detected objects can look very differently on different scenes because of the different scaling, rotations and lighting conditions. The brightness transfer function can be estimated and compensated [64, 110].

When the camera fields of view don't overlap, the task becomes much more challenging. The camera positions can be either known [115] or unknown [99]. For reidentification, mean a posteriori (MAP) is estimated, giving the probability of the detected object being the same [64, 61]. [71] used a probabilistic Bayesian model formulation with previously known transition functions. [68] explored this system for controllable movable cameras.

The state of the art multi-camera vehicle tracking approaches are very domain specific. They are either set for highways with hard-coded lanes [26, 79], rely on license plate recognition [3, 35], have very narrow field of view [100] or even use magnetic sensors [80].

The methods rely on the same schema: object detection and tracking on cameras and matching these tracks using similarity and a prior knowledge about relations between the cameras. The similarity of vehicles is usually decided by the license plate recognition as mentioned before. The cameras used in the thesis are not good enough to recognize a license plate. There is a classification network for identifying a car's model, but they require a good quality image of the front of the car [102]. Since similarity between vehicles was not otherwise explored, a similarity of faces was.

The network Facenet [124] by Google is the state of the art for deciding similarity of

faces. It is built on powerful Google inception [134] network. It achieved 99.63% accuracy on the Labeled Faces in the Wild [60] dataset and 95.12% accuracy on YouTube Faces [147] dataset. This network was selected and retrained in this thesis for vehicle similarity.

3 Fisheye camera model

Figure 3.1: Frame of the provided video.

For correct estimation of position of detected objects, it is crucial to find the relationships between the camera pixel position and the real world positions.

The cameras were provided by the Brazilian party and no technical parameters were available. The model of the camera and its parameters had to be computed. A set of improvised requested calibration images were provided shown in the 3.3.

3.1 Scene localization

Before selecting the model of the cameras, another hardware error of the camera had to be compensated for. As can be seen in the image 3.1, the scene is shifted in the frame to the left down. It is not even circle, but rather an ellipse. This is due to manufacturing uncertainty of sensor placement and this error is different in each camera. Since this project has to be easily scalable when adding new cameras, and it is not convenient to measure and set the parameters manually, an universal algorithm for detecting ellipse was developed in this thesis.

The algorithm, which was designed in this thesis, is based on an iterative optimization. It takes an image as an input and produces parameters of the ellipse. From observation, the ellipse can only be either the horizontal major axis or the vertical major axis ellipse. The equation 1 of the ellipse is rather unusual, but this formulation allows faster cost function evaluation.

$$\frac{(x-s_x)^2}{a}+\frac{(y-s_y)^2}{1}=r^2 \tag{1}$$

Now we need to find the parameters s_x, s_y, a, r.

The original image I of the size H, W, and channels I_1, I_2, I_3 is transformed to a mask M of the same size by thresholding the total sum of channels on the 8-bit scale is greater or equal to 1.

$$M_{x,y}=\begin{cases}1 & if \quad \sum_{i=1}^{3} I_{i,x,y} \geq 1\\ 0 & otherwise\end{cases} \tag{2}$$

The mask M represents the scene by the pixels with the value 1 and the background by the pixels with the value 0.

We create a predicted mask $E(s_x, s_y, a, r)$ of the ellipse as

$$E_{x,y}(s_x, s_y, a, r)=\begin{cases}1 & if \quad \frac{(x-s_x)^2}{a}+\frac{(y-s_y)^2}{1} \leq r^2\\ 0 & otherwise\end{cases} \tag{3}$$

The cost function $C(M, E(s_x, s_y, a, r))$ penalizes the pixels that were masked as the scene and lie outside the ellipse and the pixels, that were masked as background and lie inside the ellipse.

$$C(M, s_x, s_y, a, r)=\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} E_{x,y}(s_x, s_y, a, r)\cdot(1-M_{x,y})+(1-E_{x,y}(s_x, s_y, a, r))\cdot M_{x,y} \tag{4}$$

The algorithm could evaluate all combinations of parameters, but the number of searched parameters can be greatly reduced by searching in a coarse to fine manner.

In each step, a baseline is set and for each parameter, a higher and a lower value by a constant is evaluated. The best value is selected and set as a new baseline for the next step and the constant is divided by two. The main idea is based on a binary search.

The cost function evaluations can be run in parallel, which can speed up the process on multi-core CPU.

3.2 Camera model

To correctly localize object from the camera, we need to know the transformations between real-world coordinates x^w, y^w, z^w and the projection on the captured frame x^f, y^f. After applying the algorithm from 3.1, we know, where in the frame the scene is projected.

First, we will consider the circle model and at the end, we will apply the transformation to the ellipse.

Computing in the cartesian coordinates is not very useful for optics, because the view of a camera is inside a cone. Instead, the world coordinates are chosen to be spherical and the frame coordinates are chosen to be polar. The world coordinates are with respect to the camera. The transformations between the world cartesian coordinates x^w, y^w, x^w and the world spherical coordinates r^w, θ^w, ϕ^w are standard transformation equations for spherical coordinates:

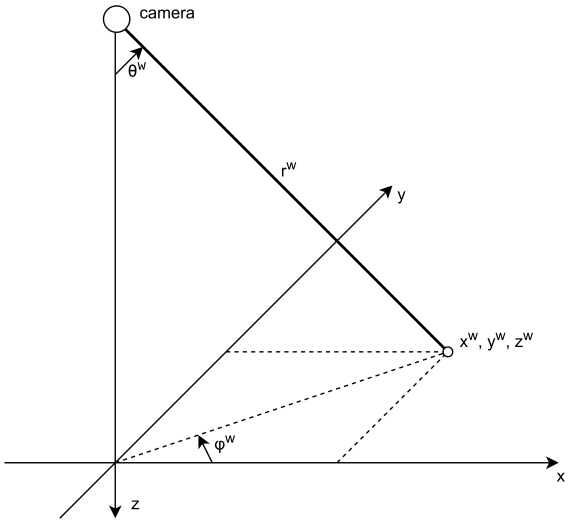


Figure 3.2: The spherical coordinates of the world.

$$\begin{aligned} x^w &= r^w \cdot cos(\theta^w) \cdot cos(\varphi^w) \\ y^w &= r^w \cdot cos(\theta^w) \cdot sin(\varphi^w) \\ z^w &= r^w \cdot sin(\theta^w) \end{aligned} \quad , \quad \begin{aligned} r^w &= \sqrt{(x^w)^2 + (y^w)^2 + (z^w)^2} \\ \theta^w &= arcsin\Big(\frac{z^w}{r^w}\Big) \\ \varphi^w &= arctan\Big(\frac{y^w}{x^w}\Big) \end{aligned} \tag{5}$$

The detected scene circle has the radius of R pixels and the center at pixels s_x, s_y. The transformations between cartesian frame coordinates x^f, y^f and the polar frame coordinates r^f, θ^f are:

$$\begin{aligned} x^f &= s_x + R \cdot r^f \cdot cos(\varphi^f) \\ y^f &= s_y + R \cdot r^f \cdot sin(\varphi^f) \end{aligned} \quad , \quad \begin{aligned} r^f &= \sqrt{(x^f - s_x)^2 + (y^f - s_y)^2} \\ \varphi^f &= arctan\big(\frac{y^f - s_y}{x^f - s_x}\big) \end{aligned} \tag{6}$$

The transformations between the world spherical coordinates r^w, θ^w, ϕ^w and the frame polar coordinates r^f, θ^f need to be found. However, there are some nice properties:

- The φ are the same, i.e. $\varphi^w = \varphi^f$. That results from aligning both coordinate systems.

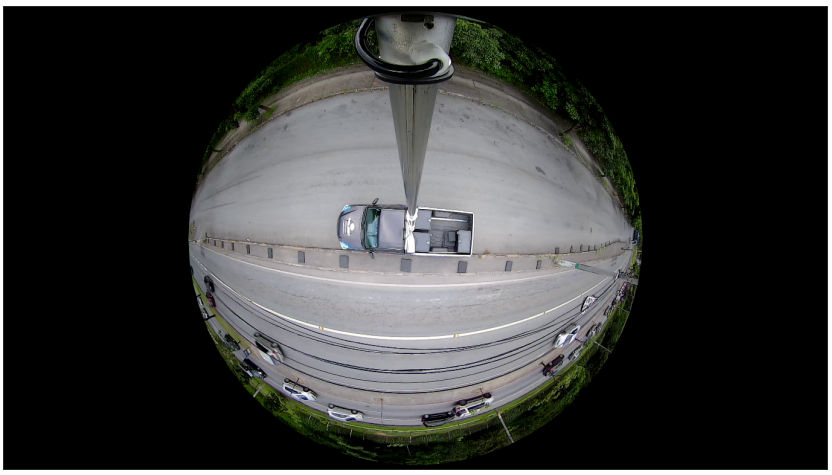


Figure 3.3: The provided calibration data.

- The transformations do not depend on r^w. The projection depends only on the direction, not on the distance from the camera.

With this knowledge, only the he transformation of θ^w and r^f needed to be found by representation of function f, such that

$$\begin{aligned} \theta^w &= f(r^f) \\ r^f &= f^{-1}(\theta^w). \end{aligned} \tag{7}$$

There are many models for finding f [28].

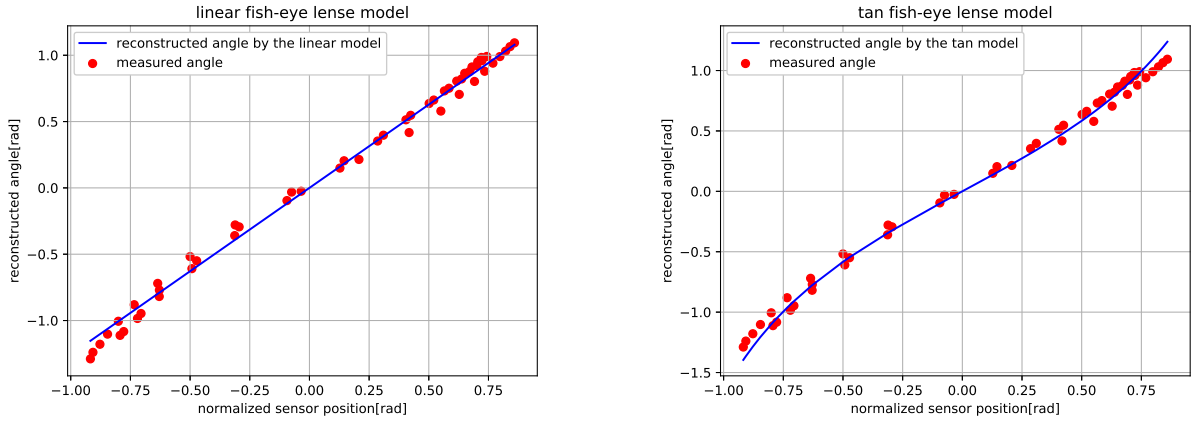
- The linear model: $f(r^f) = FOV \cdot r^f$
- The tangent model: $f(r^f) = FOV \cdot \tan(r^f)$
- The sinus model: $f(r^f) = FOV \cdot \sin(r^f)$

With each model, the only parameter, that had to be found was FOV, which is the field of view of the camera.

There was no access to the cameras, so the standard calibration using mesh could not be used. Instead, a set of marks was provided as shown in the figure 3.3. These marks are exactly 2 meters apart and are enough to estimate the function $f(r^f)$.

3.2.1 Linear model

This is the simplest model. The real world angle is proportional to the distance from the center on the image.



(a) The linear model of the lens.

(b) The tangent model of the lens.

Figure 3.4: Approximation of the calibration data by a linear and tangent model of the lens.

$$\theta^w = f(r^f) = FOV \cdot r^f \tag{8}$$

Fitting of the model is shown in the fig.3.4a

As seen in the figure 3.4a, the linear model estimates the real one well only for small angles.

3.2.2 Tangent model

This is a more complicated model, which is based on the pinhole camera model.

$$\theta^w = f(r^f) = \theta^w \cdot FOV, \tag{9}$$

Fitting of the model is shown in the fig.3.4a. This model represents the camera optics much and was chosen to be the final one.

3.3 The city coordinate system

There are many ways how to represent a vehicle in a city. The most obvious one would represent the position by longitude, latitude, and elevation. This would be correct, but not very practical. Since the distances between same circles of latitude and longitude are different, this would need more complicated transformations and there is a simpler model.

Since we care only about only one city, we will use a city cartesian coordinate system (x^c, y^c). We can choose any position and rotation of the coordinate system. All we need to know is the relative translations and rotations of each camera $(\Delta x, \Delta y, \Delta\phi)$ to the city

coordinate system. For simple transformations, we will use homogeneous coordinates [121], which are in form of $(x, y, 1)^T$. Any transformation of coordinate systems can be represented as a matrix multiplication.

A point from a camera coordinate system (x^w, y^w) is transformed to the city coordinate system (x^c, y^c) by a standard 2D rotation and translation matrix [121] as

$$\begin{bmatrix} x^c \\ y^c \\ 1 \end{bmatrix} = \begin{bmatrix} cos\Delta\phi & -sin\Delta\phi & \Delta x \\ sin\Delta\phi & cos\Delta\phi & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x^w \\ y^w \\ 1 \end{bmatrix}$$

4 Dataset generation

A detection and tracking of vehicles without using deep learning was explored. For several reasons described in this section, it was not used for the final product, but allowed a much faster video annotating, than standard methods and was used to create the dataset for training similarity between vehicles described in the section 7.4

4.1 Need for a custom dataset

There are publicly available datasets for image classification and object detection, such as ImageNet [33], COCO [90]. They also contain images of cars, people, trucks, and other classes. However, The section 7.1 shows, that these datasets cannot be used on their own because of the angle, lens type, and other factors, that differ from our cameras.

(a) Example of COCO photo for object detection.

(b) Example of ImageNet photo for image classification.

Figure 4.1: The car class examples in ImageNet [33] and COCO [90].

In our domain vehicles viewed from directly above or are very distorted. Creating dataset from provided videos is the only way to train object detection network.

4.2 Distributed system

This section, apart from dataset generation, introduces a computationally fast solution capable of running inside lamps, if they were equipped with CPU.

The final system needs to be easily scalable with additional cameras and a particular system architecture was explored. If the detections and tracking were computed on-board of the cameras, that would help greatly. There would be much less communication needed. Instead of transferring whole video streams, only some meta-data would be sent. That would include:

- Time stamps of frames.

- Locations of objects and their classes.
- Some description vector of the detections.
- Detections clustered to tracks.

This system could be greatly distributed sending packets of information among only the cameras that the information is relevant to.

However, this has some downfalls, mainly in the computational manner. Each camera would have to be equipped either with a capable computational unit. The detections, tracking, and similarities would all have to be computed onboard. Since it is not possible to have a GPU in every lamp for many reasons, for example, it is a very wet environment, usage of neural networks would not be possible. This section introduces approach for detection, tracking and classification, that could run on CPU inside lamp.

4.3 Background subtraction detection

Probably the best classical detection methods from static videos, that can be computed in real-time on limited hardware, are based on the background subtraction algorithm [109]. The main idea is creating a model of the scene without the objects that we want to detect and then subtracting the current frame and by thresholding determine, where the vehicles are. This simple approach showed many false positives and some improvements need to be made.

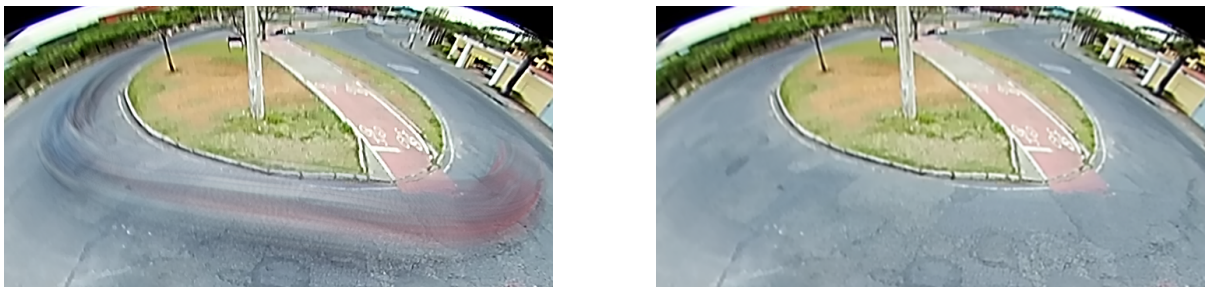
For the background subtraction procedure, a model of the background had to be found. For our purposes, an model needs to be known of how the road looks like without any vehicles and people. This can not be done by simply waiting for such a case, but the traffic is usually high. Instead, the background needs to be acquired from multiple images.

The algorithm was implemented in OpenCV [16]. The background is usually created by the mean over several images called running gaussian average [148]. The idea is to estimate a Gaussian to each pixel independently. Each pixel is updated with each new frame as a weighted sum. The background looks like a photo with a long exposition. In places, where vehicles drive, are colored lines as shown in the figure 4.2a. When computed the difference from a video frame to such a background model, as shown in the figure 4.3a, the places, where usually cars drive, can have high values. This can be bad for creating a mask by thresholding because a higher thresholding constant has to be set.

The background model changes with every new frame. In the first iteration, the background B_0 is the first frame F_0. The background in next iteration is the weighted sum of the current frame and the background model in the previous iteration.

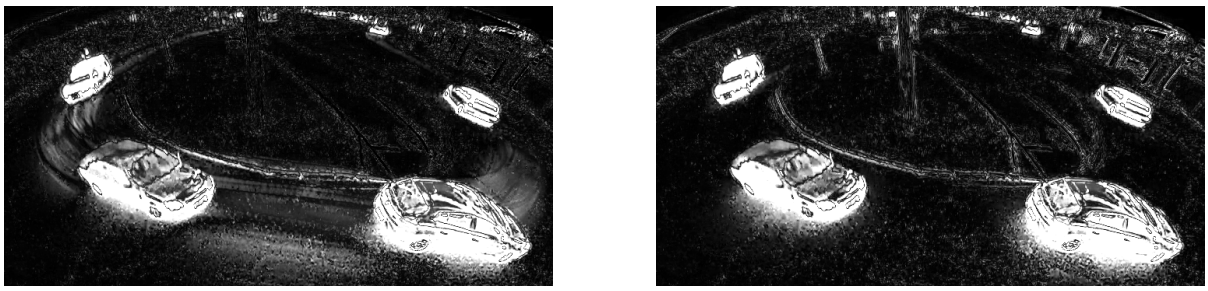
$$B_n = \alpha \cdot F_n + (1 - \alpha) \cdot B_{n-1} \tag{10}$$

This algorithm is simple, fast and can be highly parallelized. The picture having N pixels, the complexity of this standard background subtraction algorithm is $O(N)$.



(a) Mean model. (b) Median model.

Figure 4.2: Background model created by the mean and the median approach.



(a) Mean. (b) Median.

Figure 4.3: The difference between the frame and a background shown in a gray-scale.

An improved way of acquiring background model was introduced, which greatly improves the quality of current background subtraction methods. A simple change of taking the median instead of the mean at each pixel position gives a much better estimation of the background [92, 29]. The algorithm keeps a queue of K images in a memory and with each incoming frame it puts it in the database and for each pixel, it computes a median from the queue. This algorithm can be implemented with the complexity $O(N \cdot log(K))$ if we insert each pixel from an incoming frame to a sorted structure. In reality, for small K this would slow the algorithm because in OpenCV and numpy there is a great support for working with the whole images. This approach simply computes the median over all the images from the queue. The complexity is $O(N \cdot K \cdot log(K))$. The K was set to 35. The histogram of a particular pixel position over the queue is shown in the figure 4.4.

This turns out to work much better, but still has its limits. If the traffic is very high and vehicles occupy on average more than half of the ground, the background model will fail, but mean approach would fail as well.

Some more complicated models based on unsupervised learning, such as clustering or taking the most frequent bin from histogram for each pixel position could be used, but they would be computationally too complex and would not be practical for real-time video.

The difference between background model and the current frame is very noisy and some filtration had to be made. Before subtraction, the background and the frame were filtrated with a Gaussian filter with the size 3x3 for smoothing. This compensates for the camera

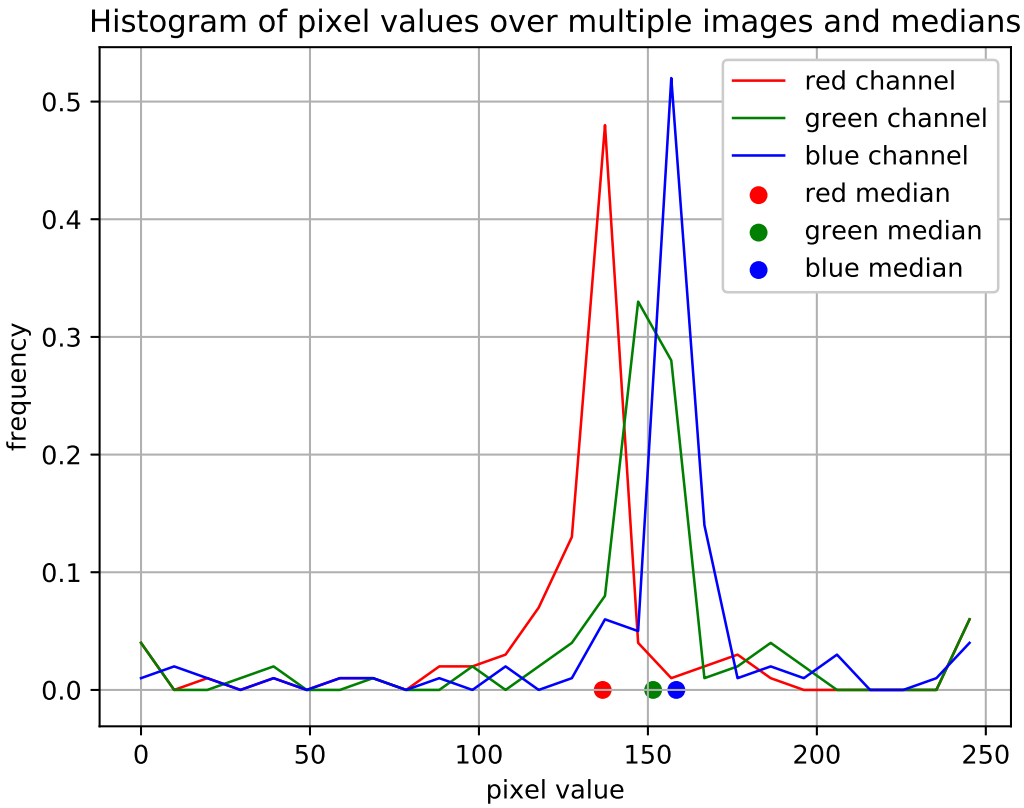


Figure 4.4: The histogram of a particular pixel over 100 images with computed medians from the scene in figure 4.2.

vibrations. Then smoothed again with the filter 11x11. This serves as an apriori probability. The idea is, that if there are big differences in the neighboring area, it is a higher probability of the pixel belonging to the car. This also helps to detect gray and black cars, which have a similar color to the road. Another advantage is, that this greatly reduces noise and helps to detect vehicles as a whole.

This differential image is thresholded and a mask is obtained as shown in the figure 4.5b. Each blob is presented with a contour and they are thresholded once more by the area. The resulted blobs become detections and a bounding box is created.

The background must be continuously adapting to the scene, but the rate of adapting is crucial. If the background is changing too slowly, it will create false positives with changes of lightning from coming clouds, etc. If the background adapts too fast, it will start to contain cars, that stop at the cross section and when the cars leave, this will become a new false detection. From experiments, there is no optimal adapting rate and it depends on the scene, weather and even then these problems will not completely disappear. One small advantage is filtering detections while adapting background.

Background subtraction is a computationally fast detection algorithm and when hav-

(a) A frame for detection.

(b) The detections and areas of contours.

Figure 4.5: The background subtraction detection algorithm.

ing perfect conditions, generated bounding boxes are accurate. Unfortunately, it has many downsides:

- Moving trees and their shadows create false positives.
- Overlapping vehicles are detected as only one.
- It works badly in high traffic because it can't create a correct background model.
- It is very sensitive to changes of lightning, such as moving clouds.
- It is very sensitive to setting of hyper-parameters.

Most of these points relate to changing the background. Especially if the scene is partly cloudy and the lighting changes a lot, the background model needs to adapt quickly. On the other hand, if in the scene is a traffic light, cars spend a lot of time on one spot and could be incorporated to the background model. Not only the car will not be detected, but a false positive will be detected when the car leaves.

For these problems, background subtraction alone can't be used as a good detector, but on perfect scenes, it can be very useful for collecting high-quality training data for neural networks detectors, as described in the section 7.2.

4.4 Optical Flow tracking

The detections were described in section 4.3. Having only the detections for each frame does not give us that much information. We need to connect these detections to a track.

A custom set of algorithms was implemented in opencv[16] for extending the background subtraction algorithm to track. Optical flow [7] is used for a motion estimation in a video. It pairs pixels in two subsequent frames. In other words, it is a discrete 2D vector field, where each vector is a displacement vector showing the movement of points from the first frame to second.

The computation of optical flow over the whole image is usually a very expensive procedure. The method used is Lucas-Kanede method [94].

This computation is not used on the whole image. That would be computationally too expensive and would not be feasible for limited computational resources and real-time system. Instead, each detection is extended for a one optical flow point. In the next frame, this point will move with the object. Each detection is characterized by this point.

This extends the detector for object tracking and partially solves the problem of two overlapping vehicles. If two vehicles drive close to each other, background subtraction would start to treat them as one object. This improved model will detect this situation and try to keep the bounding boxes on the different vehicles.

In first experiments, when a new detection appeared, the position of the optical flow point was selected with the Shi-Tomasi [126] algorithm, which is an improved version of the Harris corner and edge detector [51]. It tries to find some features, that have a high gradient and will be easier to track, rather than selecting just the middle of the bounding box.

In a perfect scenario, the algorithm could be used without further improvements. However, in real-world scenario, false positives, as well as false negatives detections must be dealt with. Furthermore, trees and lamps also complicate the situation greatly. When a vehicle drives behind some sort of a pillar, the optical flow point cannot be matched with a next frame and stays in the same place in the frame. When the vehicle completely passes, the tracking is lost. A feature had to be added, which is an additional centering of the optical flow point to the middle of the bounding box. With each new frame, the optical flow point moves with the vehicle, but is also moved towards the middle of the bounding box. This solves the issues of the pillar obstacles but excludes using the Shi-Tomasi and Harris features.

The tracking algorithm is described by a set of rules. The main ones are:

- If an optical flow point is outside the background subtraction mask, it becomes a 'zombie'.
- If a background subtraction detection is without an optical flow point and there is no 'zombie' in the detection, optical flow point is created in the middle of the bounding box.
- If a zombie is not recovered in 10 frames, it disappears.

- If there are more optical flow points in one background subtraction detection, the bounding boxes continue moving with a low pass filter.
- The optical flow points are forced to the middle of the bounding box. This solves the problem of the optical flow points being created on the front of an incoming vehicle. When the vehicle is seen from a side, the point is on the edge of bounding box and due to noise can move outside.

These improvements work surprisingly well and solve the tracking problem to some extent. If vehicles overlap completely, this approach will fail, but so will most of the other ones.

4.5 Classification

There are many objects detected in a frame and need to be classified. The most important classes are a person and a vehicle. If a classification was accurate, simple scenes could be annotated 100% automatically. That would mean gaining labeled data almost for free and creating huge amounts of datasets for object detection networks.

A simple classifier was introduced using Haar features [52] and the SVM classifier. Each Haar feature is a difference of average brightness level in two rectangular areas in the image. 87 distinct haar features were extracted and a vector of real numbers was created representing the image. This vector was the input into an SVM classifier.

The SVM classifier was trained on 8144 images of cars and 10567 images of people. This dataset was extracted from the fisheye traffic videos provided to us, using the methods described in the section 4.6. The test set split ratio was 0.2.

Accuracy on the test set was 0.842. That is quite a good score without using neural networks, but still too far from deploying on the final project or dataset creation and it was not used.

4.6 Semi-supervised dataset generation

Standard annotation approaches for image detection training need the annotator to draw a rectangle and assign it a class for every detection. This is very time consuming and therefore costly. A video is usually around 25 frames per second, so to annotate a minute of video means annotate 1500 images. This process can be made easier by skipping some frames and moving the bounding boxes around. For the skipped frames linearly approximate the movement of the objects. This can speed up the process, but it still takes a very long time.

The algorithms introduced in this section cannot be used for the final product, mainly because of the problems described in the section 4.3. However, that does not mean, that this can not be used for other purposes. As mentioned before, it works very well on easy scenes. The bounding boxes are precise and this can help the annotator to annotate scenes faster and more precisely, than standard approaches. the annotator is presented with a dialog as shown

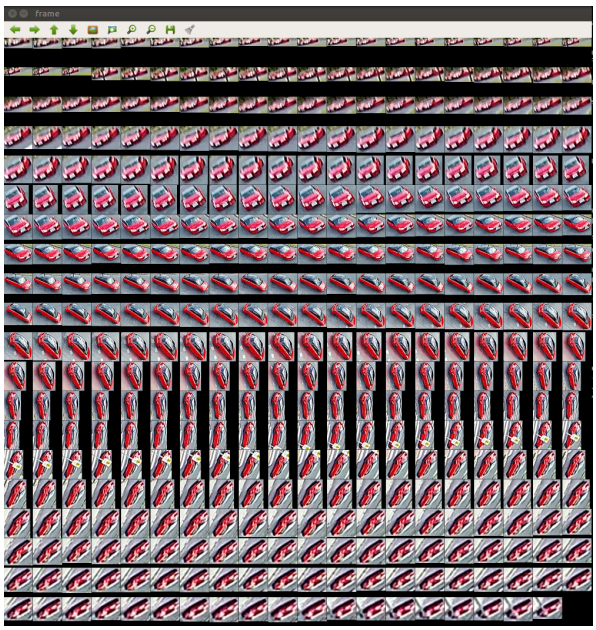


Figure 4.6: The dialog from the annotation tool.

in the figure 4.6. That includes the whole track, which is annotated by a user through one click.

Because of the not perfect classifier, classification is done by the user. The annotator manually selects the class of the detected track.

5 Convolutional Neural Networks

Deep learning is used for artificial intelligence models, which use more non-linear layers of computation. Most common are the artificial neural networks. They are a computational model inspired by a brain, that can be thought various tasks from image classification [132] to speech processing [50]. In this section, we will focus on convolutional neural networks for image processing.

5.1 Inspiration by biology

Brains are incredibly capable of processing images. Half of a human brain is either directly or indirectly devoted to vision [131]. Vision can also be a very parallelized process, which is how a brain is structured.

The basic computational unit in a brain is a neuron. A neuron in a brain has an input and an output. The input is a dendritic tree, which is connected to outputs(axons) of other neurons. Neurons are only unidirectional and their output is binary. They either fire, if the input is strong enough or they don't. Their connections to other neurons can vary from very weak to a very strong one and their size can change by learning.

The basic element of artificial neural networks is also a neuron. It also has several inputs and one output. The most common neuron can be also called a perceptron[122], which is a well-known classifier. The basic function of a neuron can be described as $f(\omega \cdot \mathbf{x} + b)$, where $\mathbf{x}$ is the input of the neuron, ω is the output and f is the activation function and b is the bias. The weights and the biases are the only thing, that changes during the training, where the goal is to find such weights and biases, that the neural networks perform the required task well.

When an architecture is created, the neural network needs to learn how to perform the required task. It figures that out in a training phase from showing the network many input data with labels, for example, many pictures and classes of the objects in the images. This is called a supervised learning. After the training phase, the weights are frozen and the network can perform the task, that is was trained to do.

While the structure of the neural network is inspired by the brain, the training is not. The way it is performed is explained more in detail in the section 5.3.

5.2 Layers

Neurons in artificial neural networks, similarly like in a brain, are organized in layers. These layers are usually connected to each other in a serial way. Neurons in one layer perform the same function. There are more types of layers depending on the neuron function and the way they are connected. The information describing this defines the neural network architecture.

5.2.1 Convolutional layer

(a) An example of a convolutional layer.

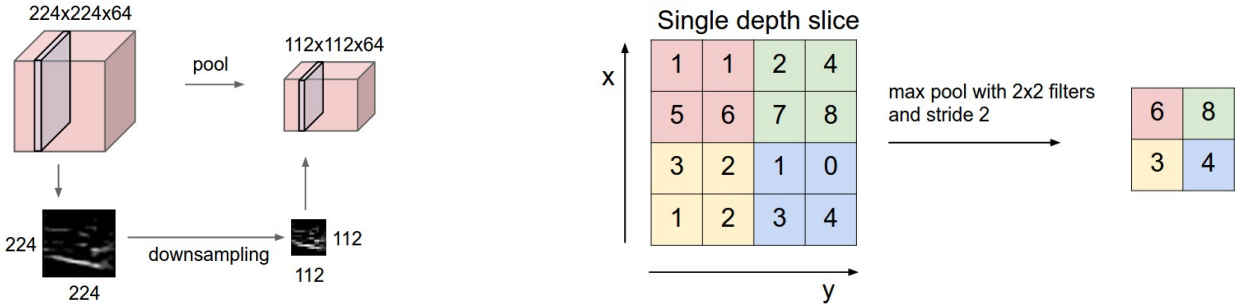
(b) Model of a neuron.

Figure 5.1: Examples of neural networks concepts from [69].

The convolutional layer is a set of neurons placed in a grid of size $m \times n \times k$. Each neuron is connected to a local region in the previous layer. Each neuron performs the function $f(\omega \cdot \mathbf{x} + b)$ shown in the figure 5.2b. Thanks to their regular structure and sharing parameters they perform a convolution function, where they look for different features in the previous layer. The neurons in lower layers can detect edges or lines, while neurons in higher layers can detect eyes or wheels. The neurons can be in a sparser grid relative to the previous layer with gaps called stride, downsampling the previous layer.

A better insight into this phenomena was described in [155].

5.2.2 Pooling layer



(a) Pooling downsamples the previous layer.

(b) Max pooling operation.

Figure 5.2: Examples of pooling concepts from [69].

The image has usually high resolution, but the information gained can be described in the much smaller information. Because at the lower levels we care about the relative positions of different features, such as lines or colors a lot, with the higher level, where the generalization is bigger, the dimension of the network usually becomes smaller. Pooling neuron takes a set of inputs and returns only one output as the highest input(max pool) or the average of the inputs(average pool). The pooling layer[123] is an important part of almost all convolutional neural networks and it can not be trained since it does not have any parameters.

5.2.3 Fully connected layer

In the fully connected layer, each of the neurons is connected to every neuron in the previous layer. Since the layers can have many neurons, this is a very expensive layer and is usually performed in the last layers of the network, where the layers are smaller.

5.2.4 Overfitting and dropout layer

Neural networks have many parameters and high Vapnik-Chervonenkis dimension [13], therefore it is more prone to overfitting on small datasets. The network should figure out from the training set of examples some basic understanding of the problem and use this knowledge to process a sample, that it has never seen before. Overfitting means, that the network performs well on the training set by learning it by heart, but fails on the test set. This is a general phenomenon in machine learning. Usually, the way is to select a simpler classifier, that can still handle the problem, regularization(penalizing high weights) or increase the training data. Neural networks have come up with a solution called dropout, which does not solve the problem completely, but greatly helps.

A neuron in a dropout layer has only one input and during training randomly copies its value to the output or returns a zero. This forces the network to build more robust connections. This layer is active only during training.

5.3 Backpropagation

As mentioned before, the learning of artificial neural networks is very different from learning of a real brain. With each input example, the network performs a forward pass. The information flows from one layer to next layer and the operations are performed until the information reaches the output layer. Information is compared with the ground truth embedding and a backpropagation is performed, that changes the parameters of the network in a way, that next time the output is closer to the required embedding. The difference is called loss and there are different ways to compute it.

The simplest square loss is computed as

$$L(\mathbf{y}) = -\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{11}$$

After each forward pass, backpropagation finds a gradient for each weight and bias ω, that minimizes the loss and changes it in the direction.

$$\omega_{t+1} = \omega_t + \frac{\partial L(\mathbf{y_t})}{\partial \omega} \tag{12}$$

This general process is called a gradient descent. Backpropagation is an efficient way how to compute these partial derivatives from back to the front of the network based on the chain rule.

5.4 Transfer learning

Since filters in lower layers detect just very simple features, in comparison to higher layers, they are not too domain specific. If a network was trained on images of animals and we wanted to retrain it to detect vehicles, the higher level filters for detecting eyes or legs could not be used. In contrast, the low-level features detecting edges or lines could stay unchanged. This is the high-level idea of transfer learning. Training the network on a different domain and retraining it for a different one.

This has many advantages. Training of the lower layers is more difficult because of the vanishing gradient problem [57]. Training of these layers also requires a huge amount of data and computational time. There are many different trained networks available online. Using such a network and only fine tuning it [56] to a specific task requires also less training data. However, this is possible only for the exact same architectures of the networks.

5.5 Frameworks

Artificial neural networks require sometimes billions of very simple operations. Although they have been known for many years[145], only recently they have had a great success. That is due to increased computational power, parallelizing the computations on GPU, which perform these operations much faster than CPU and access to a big amount of data.

There are also some frameworks, that offer very fast computations on GPU such as Google's TensorFlow [2], Theano [9] or CNTK[125] by Microsoft, from which the TensorFlow is the most common and the reidentification part of the thesis described in the section 7.4 was implemented in this framework.

There are also some libraries such as Keras [24] and Caffe [65] that make implementing deep learning in C++ or python much easier. The SSD object detection network described in the section 7.2 was implemented in Keras.

6 Classification, Detection and Reidentification networks

The Neural networks are the state of the art for all kinds of image processing. The easier task is image classification, for which the driving force has been the ILSVRC [33] competition. This task is considered to be solved, but networks for classification are being used as a backbone (base network) for object detection networks. VGG [128] described in the section 6.0.1 was used in this thesis as a base network for SSD [87] described in the section 6.2. The Google Inception [132] described in the section 6.1 was used as a base network for the Facenet [40] described in the section 6.5.

6.0.1 VGG

The VGG[128] architecture from Oxford is one of the simplest ones but is very accurate. It is often used as a base network in object detection networks such as YOLO[117], ARTOS[5], SSD [91] or Faster R-CNN [120]. In this thesis, the VGG network was used as the SSD backbone and was adjusted for this specific domain in the section 7.2.

The VGG is an image classification network. It takes an input image and produces probabilities for each predefined class. It was inspired by [25] and [77].

The input resolution of the network is fixed to 224 $\times$ 224. The idea of the network is to use very small filters 3 $\times$ 3 (which is the smallest size to capture the notion of left/right, up/down, center)[128]. Two of these filters stacked on top of each other create a receptive field of 5 $\times$ 5 and three have the receptive field of 7 $\times$ 7. With sharing parameters, the three layers have almost half the parameters, then the layer with 7 $\times$ 7 filters. This is different from [77] with 11 $\times$ 11 filter in the first layer and [155] with 7 $\times$ 7.

To bring more non-linearity, the VGG16 uses filters with 1 $\times$ 1 filters. This was used also in [83] as a network in a network.

Thanks to padding and the stride 1 of convolutional layers, the layers are down-sampled only by 2 $\times$ 2 max-pool layers with stride 2. The number of channels increases for better processing of higher level features.

Different VGG architectures were designed in [128] and their architectures are shown in the figure 6.1.

6.1 Inception

The Google Inception network [132] was used as the base network for Facenet [124] described in the section 6.5, that was used for computing vehicle similarity. Inception was the winner of the 2014 ILSVRC [33] with the 6.67% top-5 error rate. It has only 4 million parameters compared to AlexNet [77] with 60 million and VGG16 [128] with 138 million parameters. A network with a low number of parameters is easier to deploy on less powerful machines and is less prone to overfitting.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 6.1: The different VGG architectures. The ReLU function is not shown for simplicity. [128]

They achieved this by introducing an inception module shown in the figure 6.2. These are basically small models inside a bigger model. The idea is, that when adding a layer, one must decide which layer to use. The Inception module combines all these possibilities together and performs a 1×1 convolution, 3×3 convolution, 5×5 convolution and 3×3 pooling layer.

Apart from classification training, Inception network was used for face identification [40] and object detection [105, 133].

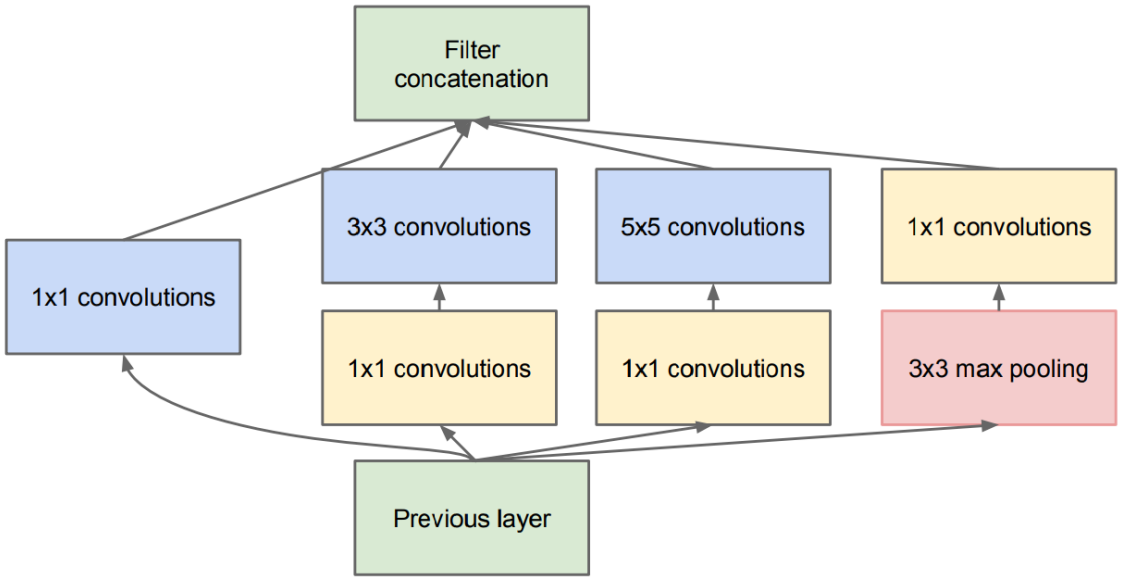


Figure 6.2: Inception module [132] with dimension reductions.

6.2 SSD network for detection

The state of the art in detection is at the time the Single Shot MultiBox detector [91]. This network provides accurate real-time object detections (74.3% mAP at 59fps on VOC2007 on GPU). They achieve much faster speed, compared to previous Faster R-CNN (7 fps on GPU) thanks to eliminating the bounding box proposals. Since the SSD architecture is much simpler and compact than R-CNN architectures, the training is performed end to end on a single model, instead of training multiple networks for region proposal, classification and bounding box regression.

6.2.1 Architecture

The SSD network uses one feed forward flow to produce a fixed number of bounding boxes and their probability for each class. The early layers are sometimes called a base network or a backbone. They are a classical architecture for image classification.

Some additional layers were added on top of the base network shown in fig. 6.3. Multi-scale feature maps for detection are convolutional layers that progressively decrease their size and allow cheap detections of multiple scales. Convolutional predictors for detection can from feature maps of size $m \times n$ with p channels can predict scores for categories or the offsets for each 3×3 element using small kernels of $3 \times 3 \times p$. With decreasing feature maps, the predictions relate to different object scales. The bounding box offsets are measured relative to each feature map location.

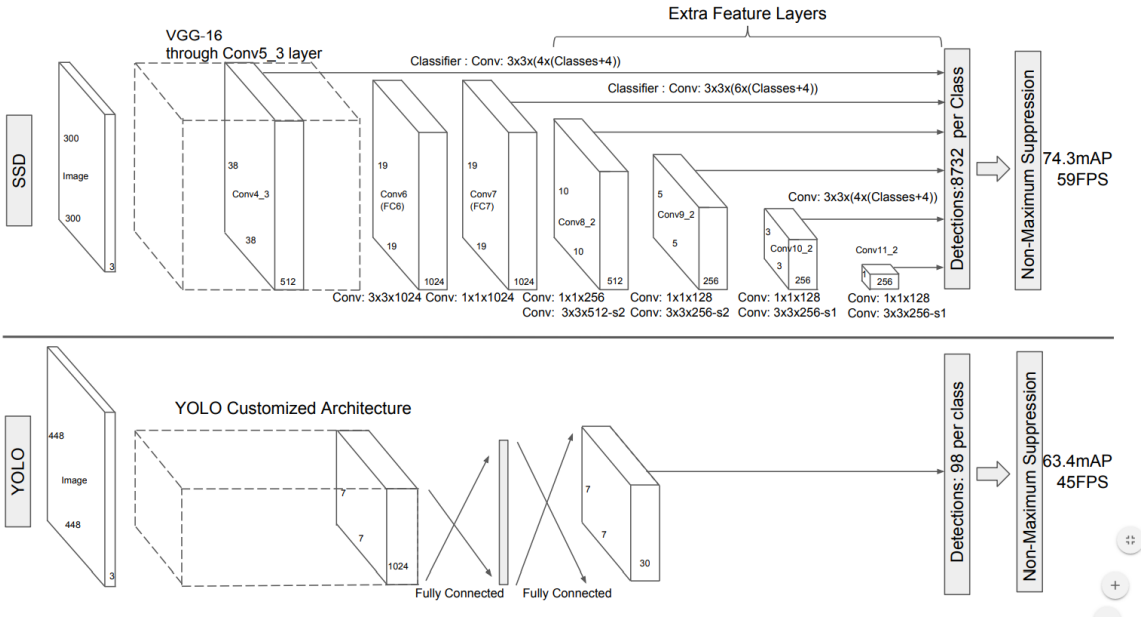
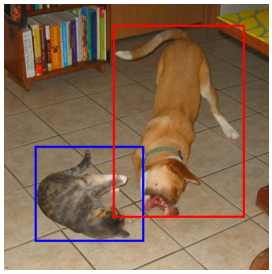
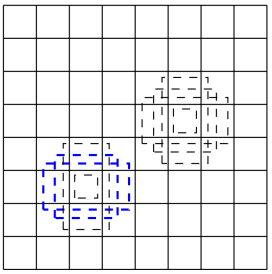


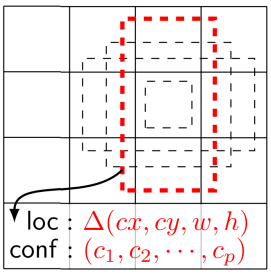
Figure 6.3: Comparison of the SSD[91](300x300) and YOLO[117](448x448) architectures.



(a) SSD prediction.



(b) 8 × 8 feature map.



(c) The predicted offsets.

Figure 6.4: The SSD [91] predictions.

6.2.2 Default boxes and aspect ratios

Each feature map cell, from the feature maps at the end of the network, is associated with a set of bounding boxes. They differ in shape and scale as shown in the figure 6.4c. Since the feature maps are computed from convolutions and max pool layers, the bounding boxes regularly tile the input image as shown in the figure 6.4c with fixed positions.

A prediction of a probability of each class and predicted offset of the bounding box is computed. The offsets are given relative to the associated fixed position of the bounding box as shown in the figure 6.4b. For predicting k shapes and c classes with 4 numbers representing the translations $\Delta(cx, cy, w, h)$, we need $k(c+4)$ filters for each feature map cell and for one $m \times n$ feature map layer we need $mnk(c+4)$ filters. The default boxes are similar to anchor boxes in Faster R-CNN with the difference, that SSD uses multiple feature maps for different object scales.

6.3 Loss

The network is trained with two training objectives. The first one is the classification loss L_{conf} and the second one is the localization L_{loc} loss. These two losses are added to a unified loss L and propagate together. The loss according to [87] is:

$$L(x, c, lg) = \frac{1}{N} L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \tag{13}$$

where N is the number of matched boxes, $x = \{0, 1\}$ is indicator for matching default boxes to ground truth boxes, c is the confidence of each predicted class, l is the predicted bounding box and g is the ground truth box and α is a weighting constant. The L_{conf} is the cross-entropy maximizing the ground truth class confidence and minimizing other classes. L_{loc} maximizes the Jaccard overlap of good predictions using the Smooth L1 loss [45]. Both of these losses are easily differentiable and can be backpropagated.

6.3.1 Training

With each training image, the ground truth boxes need to be assigned to a default bounding box. The ground truth box is matched to all default boxes, which have the Jaccard overlap higher than 0.5. This is more robust than picking only the one box with the highest overlap as in MultiBox [37]. After that, the training is performed end to end by a standard backpropagation.

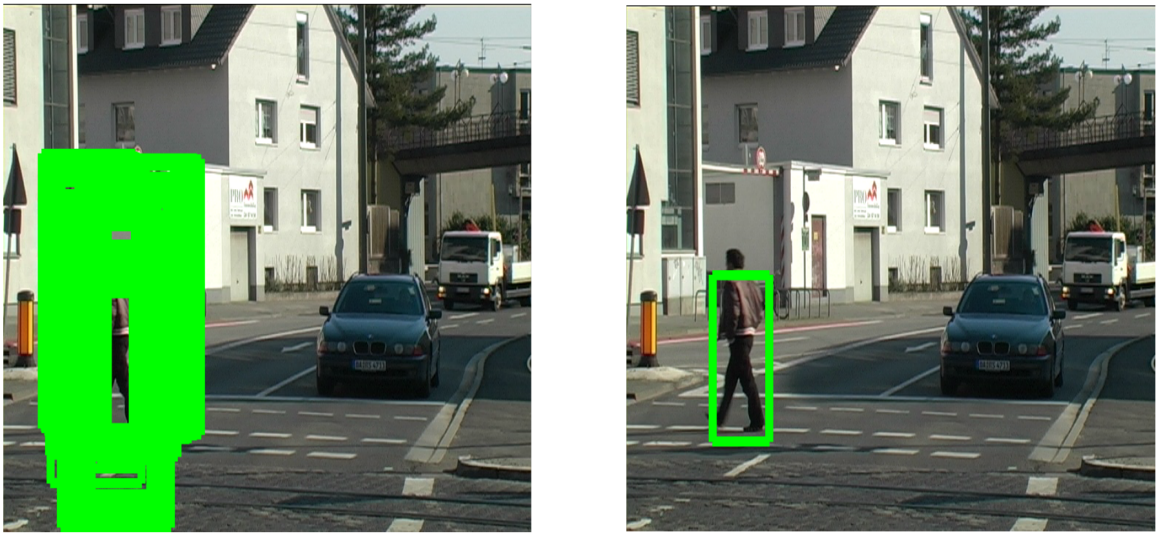
The training phase also includes a positive and a hard negative mining. Not all predicted boxes participate in training. Most of the predicted boxes are usually negatives. The backpropagation is on all the boxes matched with ground truth box as well as the negatives with the lowest score for the background. This is called hard negative mining and it speeds up the convergence. The negatives are picked with the ratio 3:1 to the positives.

6.4 Non-maxima suppression

Detectors usually create many predictions around an object as shown in the figure 6.5a. The goal of non-maxima suppression is to keep only a single bounding box.

Each prediction has a confidence. Non-maxima suppression sorts all predictions by their confidence. It takes a prediction with the higher confidence and finds all predictions with a Jaccard overlap[136] higher than some constant, removes them and moves to next prediction. This is a naive approach and use of k-d trees can speed up the process.

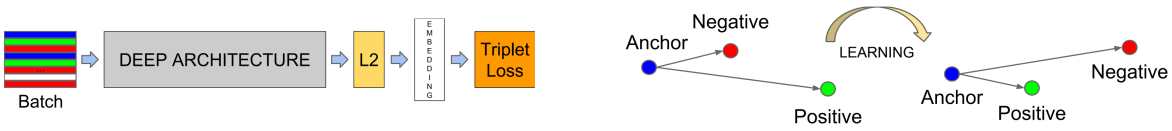
The constant is the only parameter of the algorithm and setting it is a very important thing. If set too high, objects will have multiple detections. If set too small, objects, that are close together, will be detected as one. The rule of thumb is to set the constant between 0.3 and 0.5.



(a) Predictions before applying nms.

(b) Predictions after applying nms.

Figure 6.5: Non maxima suppression[34] keeps a single prediction.



(a) The model structure.

(b) The learning objective.

Figure 6.6: The Facenet [124].

6.5 Facenet for reidentification

Google Facenet [124] is a deep neural network for computing similarities between faces but can be retrained for computing similarities between objects from any domain, such as vehicles. This network receives an input image in RGB with the size of 220×220 and returns an embedding vector with the length of 128. The Euclidean distance between samples directly corresponds to the similarity of objects. Faces of the same person are mapped to the similar place in the Euclidean space. Once we have these metrics, recognition becomes a K-NN and verification is a simple thresholding of the distance. The goal is, that the embeddings will be invariant to pose, illumination and other factors. For computing, the similarity of an image with more images representing the same identity, a mean of the distances is taken.

6.5.1 Architecture

The facenet architecture is mostly an image classification network for 128 categories and a normalization layer. Google has tried various architectures, such as Zeiler&Fergus [155] (with the input resolution 220×220), Google Inception [132] (224×224, 160×160, 96×96) and mini Inception (165×165) and tiny Inception(140×140).

The [132] shows, that the embedding space dimensionality of 128 is enough for recognizing faces.

6.5.2 Training

The training and evaluation data are sets of images of different people. The goal of the network is to project images of the same person close to each other and different people far from each other as shown in the figure 7.4b. This is a different problem, the classification, where a finite number of categories is previously known. There have been attempts [149, 135] at training network for classification and then removing the last classification layers, but with less success.

The introduced approach is different from an image classification training and is called triplet loss. With each training step, three examples are selected. Two of the same class and one from a different one. One of the positives is called anchor, the one from the same class is called a positive and the third one a negative sample. The goal is to change the network parameters in a way, that the positive embedding is moved closer to the anchor and the negative further from the anchor.

The triplet loss is represented as $f(x) \in R^d$, where x is the input image that is transformed into an Euclidean d-dimensional space. The last normalization layer normalizes the embedding to a hypersphere of the size 1, meaning $\|f(x)\|_2 = 1$

To ensure, that the model is consistent, meaning, that there exists a threshold, that will correctly classify every pair of the images, we need to ensure, that for $\alpha = 0$

$$\begin{aligned} &\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \\ &\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \end{aligned} \tag{14}$$

The α is a constant, that can enforce a margin. T is a set of all possible combinations of images in the training set.

The loss, that is being used for training according to [132] is

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+. \tag{15}$$

After a couple of steps of training, most of the images will be classified correctly and the constraint will be met. For fast convergence, it is crucial to select those triplets, that violate the condition. This is called hard positive and hard negative mining.

There were recently some improvements of triplet loss in form of quadruplet loss [22], that enforces greater margins among classes.

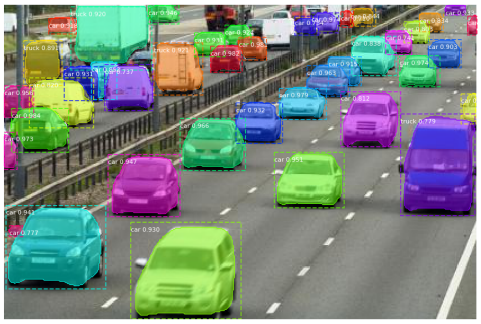
6.6 Multi camera tracking

After solving the task of detection, single camera tracking, and similarity metrics, The task needs to be extended to the whole city. It is important to note, that the city with built cameras in lamps does not exist at the time of writing the thesis, therefore these ideas cannot be tested on the whole city. Only a theoretical approach was explored. It is also important to keep in mind the task of tracking one vehicle in a city, therefore not tracking all vehicles at once, which can

7 Implementation

The deep learning runs in TensorFlow [2] framework. Keras [24] was used for implementation and development of neural networks, opencv [16] for video and image operations and numpy [141] for fast vector operations. The whole project was written in python.

7.1 Mask R-CNN segmentation



(a) Mask R-CNN on a standard street camera.

(b) Mask R-CNN on a fisheye camera has problems recognizing vehicles from the top.

Figure 7.1: Examples of Mask R-CNN network [53].

First, a couple of approaches were explored. One of them is Mask R-CNN[53]. This network builds on detections and provides segmentation. This gives more information since we know exactly what parts of objects belong to which entity, making the tracking simpler. As shown in the figure 7.1a, Mask R-CNN can very well detect standard traffic scenes, however, in the figure 7.1b one can see, that it has problems detecting vehicles from the top. This is due to a very small number of vehicles viewed right from the top in the Microsoft COCO dataset [90].

Since there are no datasets for training and evaluating segmentation from the top-places fisheye street cameras the performance was not measured. Transfer learning could be applied, but creating such a segmentation dataset is very expensive to create. Combining this fact with the 5 fps of this network, it is not feasible for real-time application and this approach was not explored further.

However, it is a valuable insight and when segmentation becomes faster, this is definitely an approach worth looking into in the future work.

(a) A frame from the street camera.

(b) Temporal difference highlights moving objects.

Figure 7.2: Temporal difference helps to detect moving objects.

7.2 SSD detector

The most important part of the thesis is the vehicle detection. This is a very hard problem given the conditions. First of all, since the camera is fisheye, it is not possible to simply use an off the shelf network trained for example on ImageNet or on PASCAL VOC and use it right away.

For the detector was chosen the SSD network [91] described in the section 6.2. It is the state of the art real-time detector outperforming the R-CNN[120] and YOLO[117] in speed and accuracy as described in the section 2.2.2.

The SSD used VGG16 for the base network, which is described in the section 6.0.1. VGG networks are popular among detection networks for their performance and simplicity.

The SSD introduces two versions differing in the input resolution: SSD300 and SSD512 with the resolutions 300×300 and 512×512 respectively. Because the cameras are positioned high and because of the fisheye view, objects, especially bicycles and motorcycles, can appear very small. It is sometimes a problem for humans to detect them on 1024×1024 image, therefore the version with higher resolution was chosen.

A working network implemented in keras[41] was used as a starting point, but some modifications were made.

7.2.1 Temporal difference

As described in the section 2.2.2, the standard pipeline for video detection is detecting each frame independently. This loses much information, that can be gained from the video. Cameras used in this thesis are static and the vehicles, that are to be detected are mostly moving. A cheap segmentation of the scene is a temporal difference.

A difference between two subsequent frames is shown in the figure 7.2b. It is computed as:

$$D(F^T_{i,j}) = max(255, \sum_{c=1}^{3} |F^t_{i,j,c} - F^{t-1}_{i,j,c}|), \tag{16}$$

where $F^t_{i,j}$ is a value of a frame at the time t, i-th row and j-th column. The image is represented in 8-bit unsigned int, therefore to prevent overflow, the value is saturated on 255. The difference between frames can be computed very fast in python using numpy[141] and opencv[16] libraries.

There are more ways, how to use this information in the network. One could feed just this layer and probably achieve good detections but would have problems with object classification by losing the RGB information. The introduced approach keeps both the information by feeding both to the network.

The SSD network was extended for an extra input channel in addition to RGB, that is the temporal difference. The data have size $H \times W \times 4$ instead of $H \times W \times 3$ for standard images. This version of the network was temporarily called RGBD network. The D stands for the difference and should not be confused with commonly used distance.

In some cases, parked cars should not be detected. This is in a case of a road with a lot of parked cars and where are no traffic lights. In this case, the tracking will perform better with less bounding boxes and no occlusions with the parked cars. This can be easily done by computing the average value in the difference channel and thresholding it.

7.2.2 Architecture

The architecture is similar to the original SSD512 architecture, but some changes were made. The input layer was extended to the differential channel as described before.

According to [17], SSD might have a problem with detecting small objects. Therefore one more feature layers were added with a small scaling factor of 0.05, compared to the original 0.1. This additional feature layer is able to detect roughly twice as smaller objects as the original network, such as pedestrians, bicycles or motorcycles.

7.2.3 Dataset

A scene annotating was introduced in the section 4.6. However, this approach could not be used for SSD training and was used only for similarity training described in the section 7.4. Even though it produces no false positives, sometimes it produces false negatives due to the inability of correctly segmenting overlapping objects. It only detects such a situation and does not label anything.

Despite over-all good scene annotation, the SSD's hard negative mining strategy selects the bounding, that the network predicted most likely as a presence of an object and was not labeled and tweaks the weights in such a way, that next time it is predicted as a background. In other words, if a vehicle is not labeled in the image, the network might detect it but will be corrected, that it is a background. This would lead to a bad learning, overfitting and might not even converge.

A standard online annotation tool was provided by the Good Vision company, that allows annotators to draw bounding boxes and attach classes to images. Several annotators have annotated 1764 images.

The Brazilian party provided 15 videos from different scenes, each about 1 hour long. Each video was shot from a slope on a car, as shown in the figure 3.3. Only two cameras in lamps were working and recording video at the time of writing this thesis since the whole project is still in development.

The RGB images and the temporal difference was extracted from the total of 17 scenes a second apart and the images without traffic were deleted. There is a need for many scenes for the diversity of the roads, surroundings and lighting conditions. A detector can easily overfit on a scene by learning its background, therefore the test data were chosen to be one scene, where the detector did not learn.

The dataset contains 1654 training and 110 testing images. Because of the sparsity of the data and not many hyperparameters, the validation set is equal to the test set. There are 7 classes in the data: [person, bicycle, car, bus, motorbike, truck, animal]. The classes person and animal are not important for this project but might be used for adding new features in the future.

7.2.4 Data augmentation

For extending the dataset, many different methods of editing the images are being used. They usually contain horizontal flip, selecting patches or editing brightness.

The nature of the 360-degree camera enables us to rotate the image in an arbitrary angle, not only a horizontal flip, such as on standard datasets.

On a standard frame, not all vehicles are moving. Some are parked and some stay at a traffic or on a red light. It is important to detect these also, therefore 4th layer is set to zero with the probability of 0.1 to provide more data for the RGB part of the network.

Instead of creating a new augmented dataset, augmentation adjustments were applied on each incoming sample during training. All the data augmentation techniques used are:

- Random rotation of the image.
- Random brightness change.
- Random image translation.
- Random scaling of the image.
- Random setting of the 4th channel to zero.

7.2.5 Training

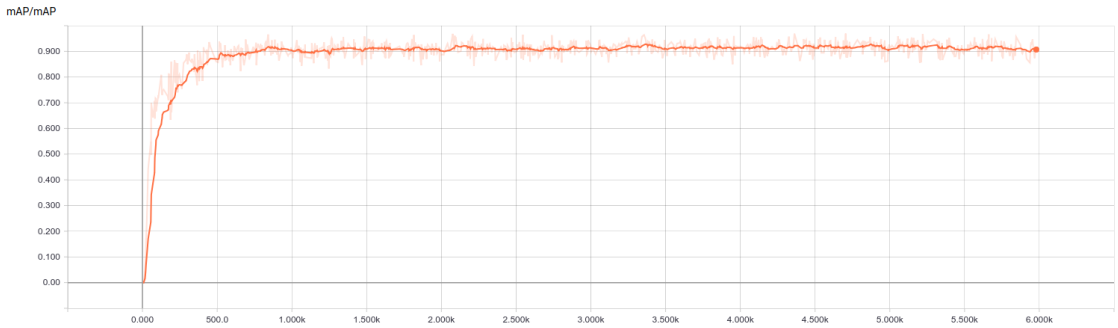
It is not possible to use a pre-trained model on ImageNet[33] or COCO [90], because of the changed architecture of the network. A pre-trained weights could not be used for transfer learning. An attempt was made to pre-train the model on Youtube-bounding boxes dataset[116], but only static videos would have to be selected because of the differential input channel and the scenes are too different from the fisheye traffic camera.

The images were shuffled before each epoch. The minimum overlap between the anchor and a label was set to 0.5, the maximum overlap between anchor and background label was set to 0.2 and the threshold for moving labels from the ground truth during training was set to 0.2. The training was performed on the NVIDIA GeForce GTX 1080 for 4 days.

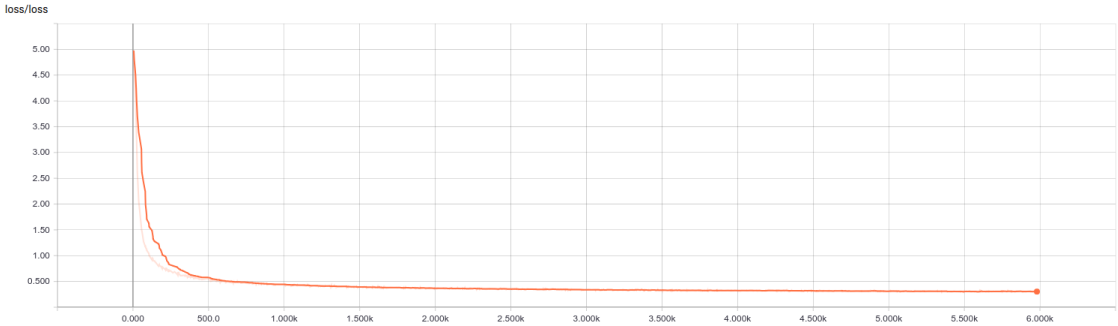
A custom tensorboard callback and mAP computation had to be implemented for observing the training progress and evaluating the model. Other features were added, such as tracking the detection ratio or computing the best threshold from the mAP curve.

The Adam optimizer [74] was used with parameters: The learning rate 10^{-3}, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, decay = 0.0005$. These parameters were selected by [41] and were not changed. The training process is shown in the figure 8.2

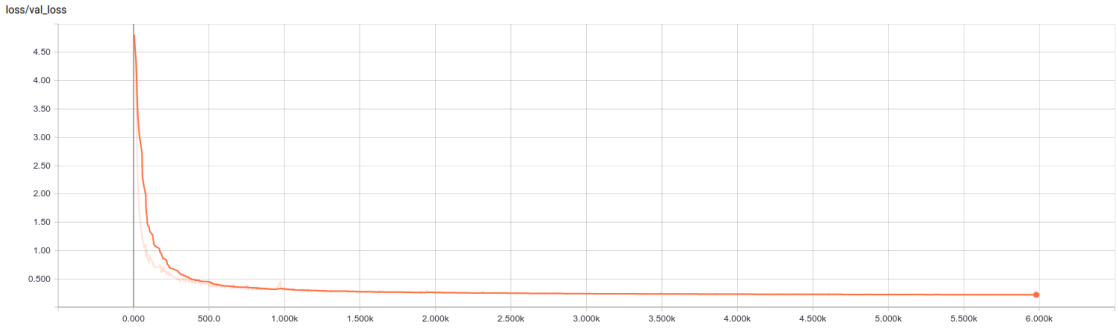
The training converged after estimated 3000 training steps. The final training loss was 0.3035 ant validation loss 0.2201. The mAP on the test set was 91,6 %.



(a) The mean average precision on validation set.



(b) The loss on the training set.



(c) The loss on the validation set.

Figure 7.3: The process of training the 4 channel input SSD network. The training was performed on the NVIDIA GeForce GTX 1080 for 4 days.

7.3 Single camera tracking

Single camera tracking is the only thing, that was not implemented by the author of the thesis. The company Good Vision provided their own state of the art tracker. It is similar to the tracker explained in the section 4.4, but more advanced.

It requires the video and a sequence of detections and connects these detections to tracks, which are sequences of these detections. This allows non-supervised clustering of track in the scene without needing to mark the initial object.

The Good Vision tracker works in three steps called seeding, displacement, and match-

ing.

7.3.1 Seeding

When a new detection appears, a set of features is initialized. These are a set of points inside the bounding box that were selected by the Shi-Tomasi [126] feature detector. The Shi-Tomasi chooses features by finding local maxima in differentiating the image in both axes and choosing the smaller of the two. It usually finds corners or other distinguishing features, that are easier to track by optical flow [7].

Because the detection is a bounding box, the features are initialized not only on the tracked vehicle but also on the background. This is undesirable since we want to track only the moving object. This could be solved by replacing object detection with segmentation but would increase the computational time of the detector and make creating datasets harder.

7.3.2 Displacement

After features were initialized, displacement of each feature is computed with the incoming frame. The movement is estimated by optical flow using Lukas Kanede [94] method. Optical flow is computed only for the sparse local features and not for the whole frame, making it faster. Those features, that were initialized on moving vehicle move with it and those on the background stay in the same place.

7.3.3 Matching

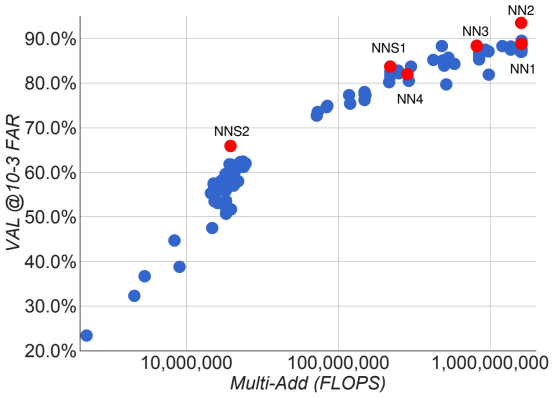
Each feature remembers, which object it originally belonged. After the features were moved in the displacement step, they are matched to the new detection. The features outside detections(those previously seeded on the background) are removed. The detections with features, that belonged to the same previous detection, take its identity and are matched together. This way a track is created.

7.4 Similarity

The neural network Facenet [124] was chosen as the metrics of similarity among vehicles. As described in the section 6.5, it is designed to compute the similarity between faces and is taught to be invariant to various poses and illumination conditions. The network can be retrained for a different domain apart from faces, in our case vehicles. Given an image of a car, the network will produce an embedding in a 128-d Euclidean space in such a way, that the same objects should be close to each other. Google has tried many architectures, that are compared in the figure 7.7d. The Inception network NN3 with the input resolution of 160×160 is a good trade-off between the speed and accuracy. Furthermore increasing input resolution is not practical for our task, since the resolution of the cameras is only full HD and the vehicles inhabit only very small area in the frame.

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	$87.9\% \pm 1.9$
NN2 (Inception 224×224)	$89.4\% \pm 1.6$
NN3 (Inception 160×160)	$88.3\% \pm 1.7$
NN4 (Inception 96×96)	$82.0\% \pm 2.3$
NNS1 (mini Inception 165×165)	$82.4\% \pm 2.4$
NNS2 (tiny Inception 140×116)	$51.9\% \pm 2.9$

(a) Mean VAL at 10^{-3} FAR.



(b) Comparison of different DNN architectures.

Figure 7.4: Comparison of various facenet[132] base networks evaluated on Labeled faces in the wild[60] and YouTube faces[147] datasets.

As a starting point, TensorFlow implementation of the network [40] was used. The network stayed unchanged, but the data handling, tensorboard callbacks, and evaluation had to be implemented.

7.4.1 Dataset

Figure 7.5: An example of a one object in a training set.

The dataset needed for this task is in a form of a set of images of an object for many objects. That means, that we need set of images of the same vehicle for many vehicles. Use of other datasets, such as [76] is not possible, because their images look too different, from those in the fisheye street camera.

The main idea in creating a custom dataset was, that if we have a track - a set of bounding boxes of the same car on one scene, we can use these detections as a set of images of the same vehicle. That is exactly what the background subtraction and optical flow do. This dataset was created by a semi-supervised annotation tool based on background subtraction and optical flow described in the section 4.6.

Different scenes and different weather conditions were used for better diversity. The created dataset contained over 9 000 images.

7.4.2 Problems with the dataset

When the facenet network was trained on this dataset, it did not work at all. There are a couple of big problems that have to do with the way Facenet learns. In each training step, it takes two images, that belong to different classes, but were classified as too similar. If each class contained a different car, this would make no problem. However, vehicles, especially in South America, are very similar. If there is a model of the same car in the database more times, the facenet will be taught, that it is a different car and it needs to change. Especially if the car is in the same scene, it looks almost identical and not even human can tell them apart.

Another problem is, that if the same model of the car appears on a different scene, there is no match with the original car. The network treats these cars as different. This is the opposite of what we want to achieve.

The final problem considers low diversity among one track. The road is usually of the same color and the lightning conditions barely change. The network can then overfit on these features and will not generalize.

7.4.3 Improving the dataset

These problems had to be taken into account when creating a new version of the dataset. It would be very hard for the annotator to remember, which types of vehicles have passed and which did not. Instead, the original dataset had to be clustered. That means putting same vehicles from different scenes into one class. This is a very time demanding process, but the results are worth it.

7.4.4 Training

To extend the dataset, some augmentation techniques were used, such as rotation of the vehicles by 90°. Transfer learning was used and the network pre-trained by Google on faces was fine-tuned on vehicles. The training took 12 hours (150 000 steps) on the NVIDIA GeForce GTX 1080. The training set contained 8800 images and the validation set 884 images from the clustered dataset. These were models of a vehicle, that the network has not seen before. The accuracy was computed by changing the task to a classification task described in

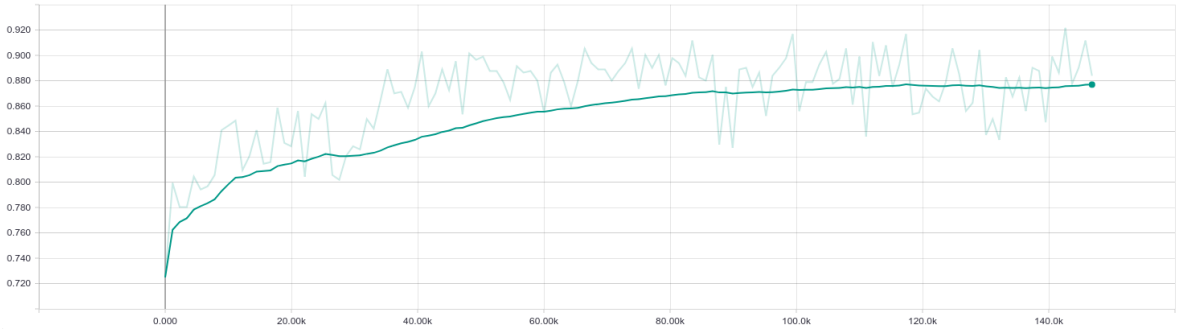
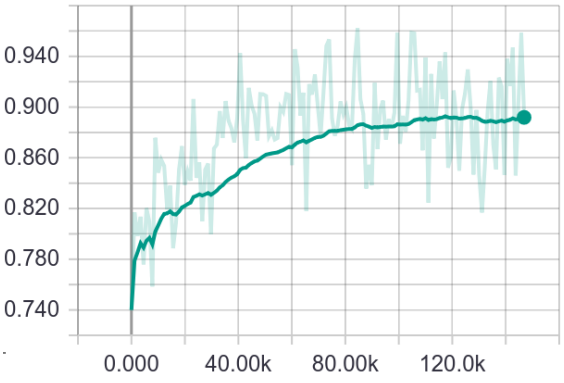
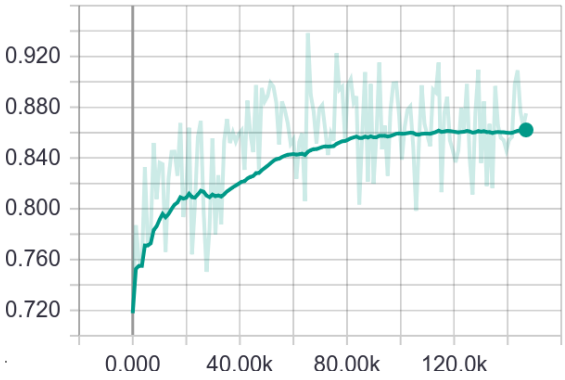


Figure 7.6: Accuracy during the facenet training.

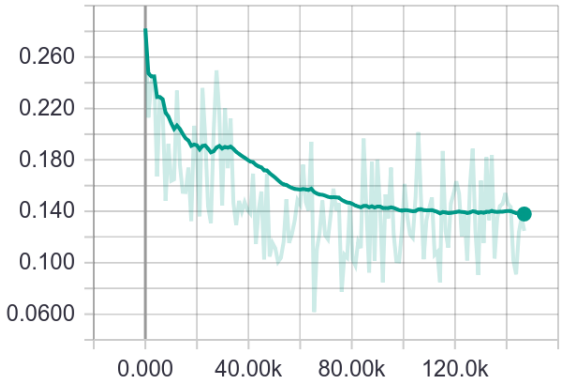
the section 8.3.1



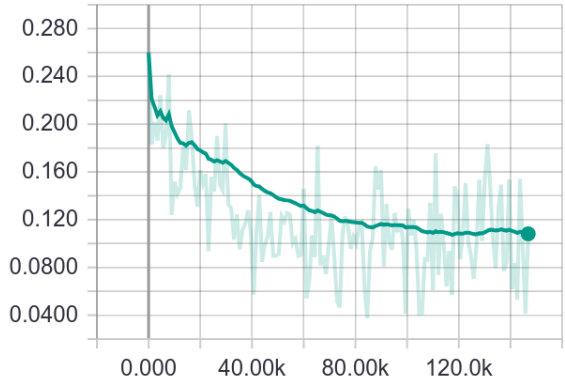
(a) True positive rate.



(b) True negative rate.



(c) False positive rate.



(d) False negative rate.

Figure 7.7: Additional information about the Facenet training. The training was performed on the NVIDIA GeForce GTX 1080 for 12 hours.

The final results are shown in the table 3.

Figure 7.8: Visualization of facenet embeddings using the T-SNE dimensionality reduction.

7.4.5 T-SNE visualization

The facenet transforms the images into a 128-dimensional Euclidean space. It is very hard to understand problems, such as the variance of rotation, lightning and so on. Simple distances between images are not enough to understand better of the embeddings.

Some dimension reduction techniques can be used for the embedding visualization. The most common techniques are PCA[146] and T-SNE[95]. PCA is better to understand the overall relations, while T-SNE can better visualize the local relations. T-SNE was chosen and a small batch of car images was selected. The T-SNE was computed in python using sklearn[108] package.

The T-SNE algorithm The results are shown in the figure 7.8.

It can be seen, that it can distinguish different colors of cars. However being presented two very similar white vehicles, it has some problems distinguishing them. Also, we can see, that the network is not fully rotation invariant, even though it was trained to be. But we need to keep in mind, that a lot of information was lost with the dimensionality reduction, therefore the preview is not very reliable.

7.5 City representation

There is a couple of ways how to represent a city, its cameras, and vehicles. It does not make much sense to represent the city as a 3D model since the city is mostly 2D and an altitude of places would be complicated to precisely measure.

If we had cameras in every lamp, this would be a different task, since we could use

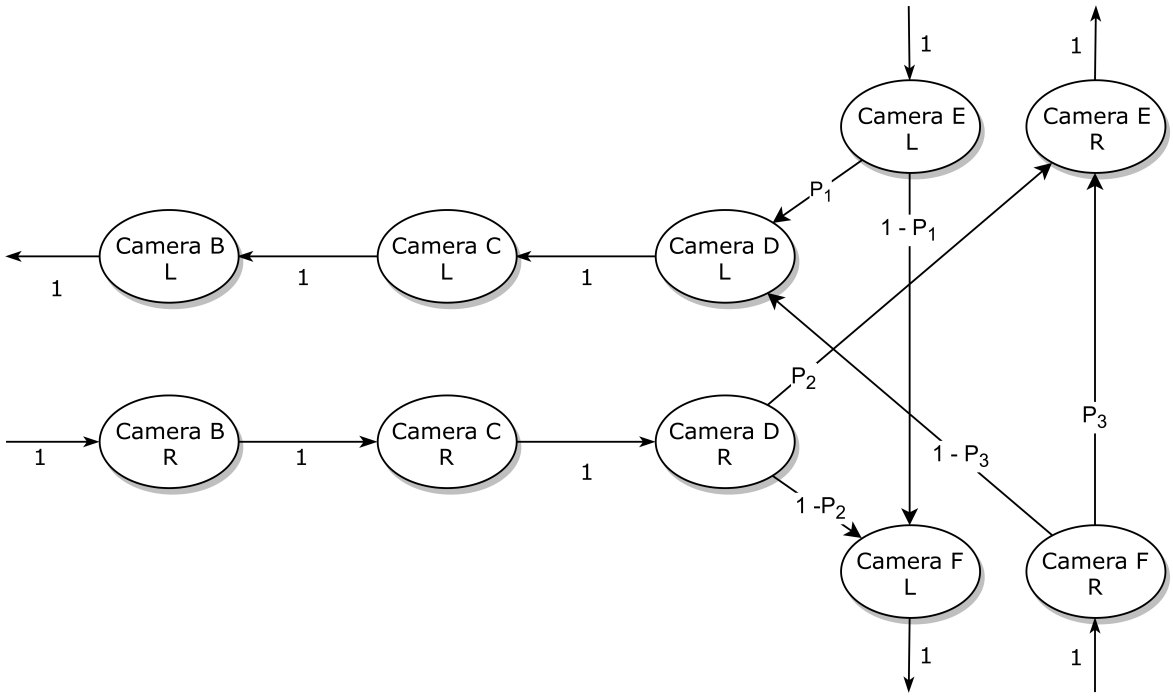


Figure 7.9: An example of a part of city representation with transition probabilities.

techniques for overlapping camera tracking. However, this is not the case. The cameras are only at some and the rest of the system is mostly unknown.

Since vehicles drive only on roads, it would be a good idea to include this knowledge in the model, rather than not constraining the vehicles positions. The model should be also easily scalable.

This leads to representing the city as a graph, rather than a 2D plane. All the cameras will be representing a set of vertices and connections between them are edges. Each vertex corresponds to a vehicle state. The edges don't necessarily represent roads directly since the cameras are not at every intersection.

The city is represented by a Markov chain, where the probabilities of all the outgoing edges sum to 1.

When a vehicle is detected on one camera, the heading is an important piece of information and can be detected from the track. Therefore we create a different state for each vehicle direction as shown in the figure 7.9.

Each edge has a probability(transition function), that corresponds to the vehicle moving in that direction. Some probabilities are deterministic, for example, a long road with multiple cameras and no intersections. Some transitions are probabilistic, as shown in the figure 7.9 as P_i or $1 - P_i$. When there is no camera at an intersection, only a probability can describe the vehicle's movement. In this case, each edge will have a constant probability assigned to it based on observation. This can be a function of for example daytime, but will not be changed

otherwise. In case of a camera at the intersection, the probability will be adjusted based on the measurement of the camera, but still can deal with some uncertainty, especially when the camera is not sure, where the vehicle actually went.

This probabilistic representation allows more advanced modeling of the city with some uncertainty.

Each of the vertices has some more parameters, which are the mean and variance of traveling between the vertices. This is obtained by observing the vehicles. This can also be a function of daytime or the traffic.

7.5.1 Reidentification

When expecting a vehicle in a camera, not only similarity but also a time and the estimated position of the target vehicle plays a role. An algorithm is presented, that takes all the information and combines it with a simple yes/no answer if a detected vehicle is the target one. If the target vehicle was at time t_0 at a state s_0 with the probability p_0 and there is a transition probability of P_0^i to a state s_i and the edge having a mean transition time μ_0^i with variance σ_0^i and the detected similarity distance between the target vehicle and the detected one is m, then the score function is

$$S(i, m, t) = m^{-1} p_0 P_0^i \frac{1}{\sigma_0^i \sqrt{2\pi}} e^{-\left(\frac{t_0 + \mu_0^i - t}{2\sigma_0^i}\right)^2} \tag{17}$$

The score function of the i-th state with the measured similarity distance m at time t is thresholded and that gives us a robust yes or no answer if the detected vehicle is the target one. If more tracks are thresholded, the track with the most matches is considered to be the target vehicle.

This method describes a reidentification of one edge. If a vehicle is detected, the probability of the vehicle being at the time t at state i is 1, and the algorithm follows the vehicle on another edge. The score function expects a normal probability of appearance of the vehicle at another camera.

The μ can also be a function of velocity and position of the car. For this, a precise calibration of the camera needed to be developed as described in the section 3

7.5.2 Decreasing computational demands

Because the detection is a computationally very expensive process and only one vehicle is being tracked, it does not make sense to run the detector on all cameras at once, especially in a big city for cameras being far away. The equation 17 can be rewritten to

$$A(i,t) = \sum_{j \in T} p_0 P_0^i \frac{1}{\sigma_j^i \sqrt{2\pi}} e^{-\left(\frac{t_j + \mu_j^i - t}{2\sigma_0^i}\right)^2} \tag{18}$$

where T is the set of all the states s_j, that have an edge leading from s_j to s_i. The score $A(i,t)$ is the probability of the target vehicle being at the camera. It can be thresholded and giving us the information about whether the camera can be turned off or not. This is generally fast to compute.

7.6 Multi camera tracking

All the different parts of the thesis were put together to be able to detect a vehicle, detect it on another camera and recognize it based on the time of arrival, direction, similarity and the prior knowledge about the cameras and their relations.

The tracking algorithm at one camera detects a track, from which many images of the target vehicle are extracted. This serves as a model for the reidentification task.

Transition times and variances are computed prior to the program execution and remain constant. After a vehicle is detected, another camera starts the detection and tracking. Each detection is thresholded according to the equation 17 and if one detection is classified as being the target identity, the whole track is declared to be the target vehicle and the model of the vehicle is extended.

8 Evaluation

Each of the modules was evaluated separately and the overall method was experimentally evaluated in a real-world scenario.

8.1 Mean average precision.

Mean average precision (mAP) is the most used metrics for object detection problem. The advantage is, that it does not depend on the selected confidence threshold, but only on the IoU threshold. This metrics is not constrained only for object detection problems in vision but can be used for all detection problems.

The algorithm for computing mAP runs for all thresholds. Given an arbitrary threshold, the predicted bounding boxes are those, whose confidence exceeds it. If there is a high IoU of a ground truth box and some predicted bounding boxes having the same class, the predicted box with the highest confidence is matched and considered true positive(TP) and no other box can be matched with the ground truth bounding box. If a predicted bounding box is not matched with any ground truth bounding boxes, it is considered false positive(FP). If a ground truth bounding box is not matched with any predicted bounding box, it is considered a false negative(FN).

Precision(P) corresponds to what portion of ground truth boxes were matched. With lowering the threshold, it can only increase, since more ground truth bounding boxes will be matched.

$$P = \frac{TP}{TP + FP} \tag{19}$$

Recall(R) corresponds to what portion of predicted bounding boxes were matched.

$$R = \frac{TP}{TP + FN} \tag{20}$$

The precision-recall curve in the Fig.8.1 shows the dependency of precision and recall. The higher the precision, the lower the recall. The area below the curve is called average precision(AP) and the mean overall classes are called mean average precision(mAP).

$$mAP = \frac{1}{|C|} \sum_{c \in C} AP_c \tag{21}$$

where C is the set of classes.

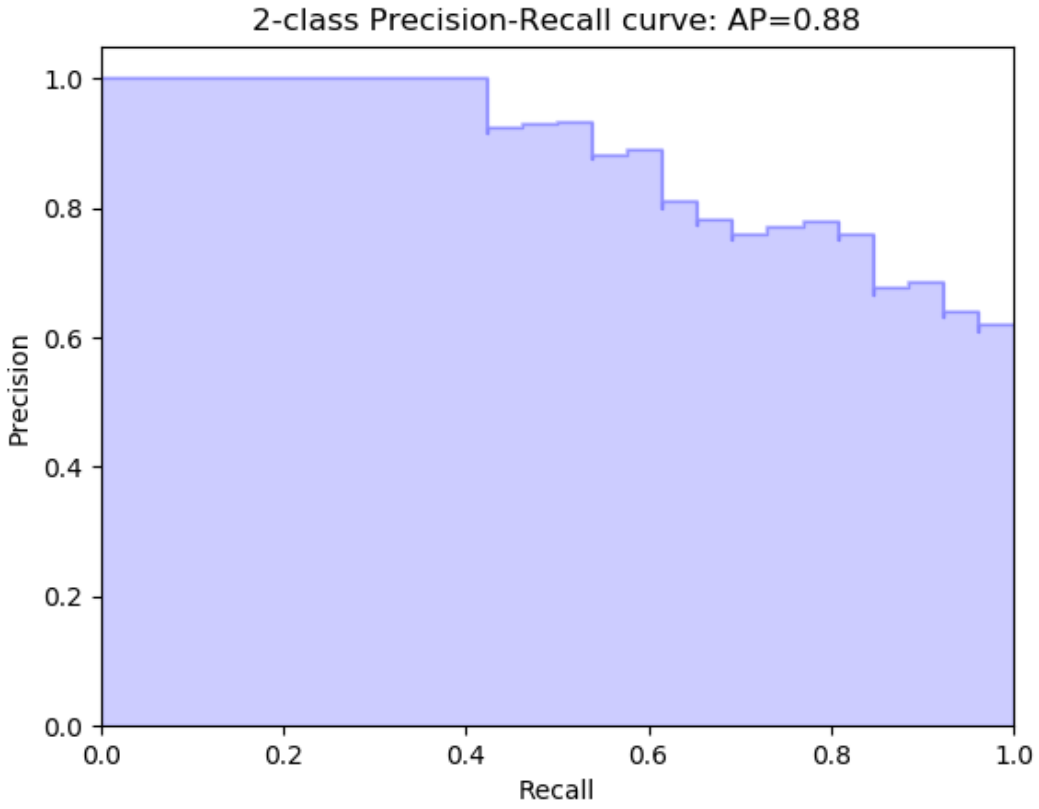
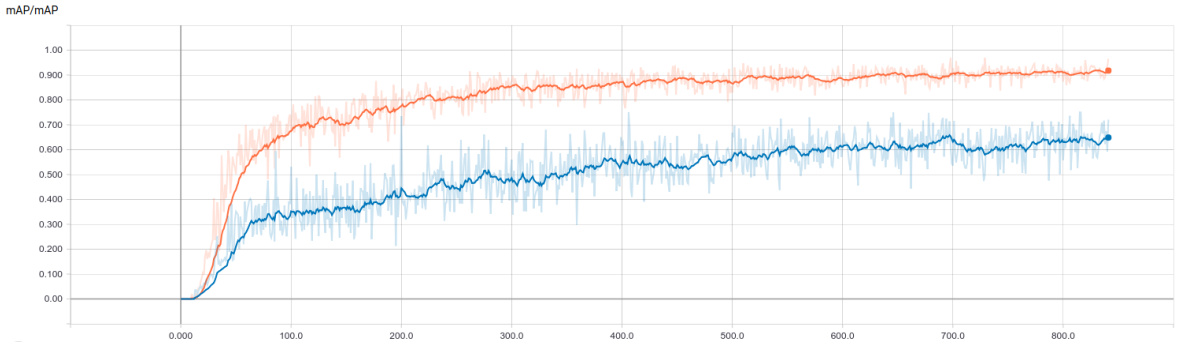


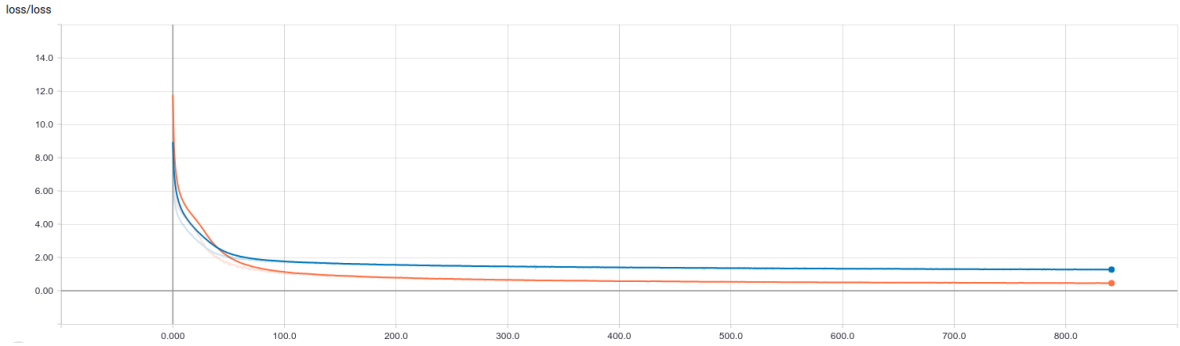
Figure 8.1: Example of arbitrary precision-recall curve.

8.2 SSD object detection

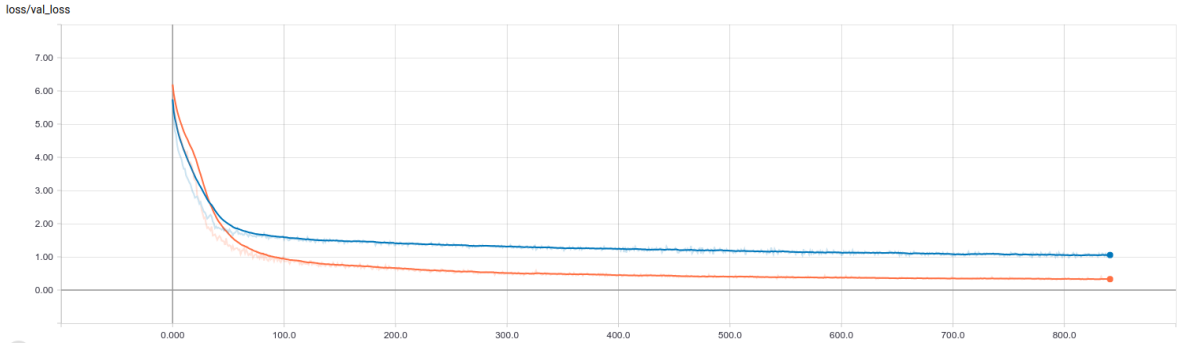
The implemented detector described in the section 7.2 was compared with the state of the art SSD [91]. Because using pre-trained SSD on ImageNet would perform badly because of the domain specifications, the models were trained with the same parameters on the same dataset.



(a) The mean average precision on validation set.



(b) The loss on the training set.



(c) The loss on the validation set.

Figure 8.2: The process of training the introduced 4 channel SSD (orange) and the 3 channel SSD (blue) networks. The training was performed on the NVIDIA GeForce GTX 1080 for 1 day. The graph shows, that the presented solution performs much better than the state of the art SSD.

Because of the hardware limitations, the training of each network took one day. After

models	Training time	Loss	Validation Loss	mAP
SSD512 (RGB)	1 day	1.127	1.055	63,2
SSD512 (RGBD)	1 day	0.4583	0.3349	90,0
SSD512 (RGBD)	4 days	0.3035	0.2201	91,6

Table 2: Results on the custom dataset from the section 7.2.3 show, that the introduced RGBD architecture is more accurate, than the standard SSD.

deciding, that the difference layer greatly helps, the RGBD network was trained for 4 days. The final comparison is shown in the table 2.

The computational overhead given the 4th channel was tested, but the speed of both models are below recognition. That is mainly because the 4th channel is only in the input layer, and increases the number of parameters only in the first hidden layer, the other layers are unchanged.

8.3 Facenet similarity

The facenet was trained on NVIDIA GeForce GTX 1080 for 12 hours on 8800 training images and was tested on different 884 images.

8.3.1 Evaluation metrics

Evaluating the similarity task was transformed to an evaluation a classification task. The task is to classify if a pair is positive (belong to the same class), or negative(does not belong to the same class). Given two images, compute the distance of their embeddings and compare it to a threshold.

The threshold is found by cross-validation on the training set. The validation pairs are selected randomly in the ratio 1:1 of being positive and negative.

8.3.2 Results

The final results are shown in the table 3

Accuracy	0.8105
True positive rate	0.811
True negative rate	0.791
False positive rate	0.209
False negative rate	0.189

Table 3: Final results of the facenet training after 12 hours on NVIDIA GeForce GTX 1080.

The facenet performance with the accuracy 81% was not as good as we expected, but we need to keep in mind, that the task is very hard. The pairs of cars are shown in different lighting conditions and from different angles. It is not easy to say, if a vehicle from a front view is the same, as a different from a back view. Also merging the dataset had a lot of decisions to make, which vehicles are considered to be same and which are not. Since this is not the only parameter to decide reidentification, this performance is enough, but it is definitely something to improve in future work.

8.3.3 Comparison to state of the art

The state of the art methods compare vehicles by their license plates and therefore there is almost no research into vehicle similarity. There is a vehicle model classification [102, 113, 81], but they all require a good quality frontal view, which is not possible with high mounted fisheye cameras.

The original Facenet achieved a much higher performance on a face domain, which is a very loose comparison. Faces have more features than vehicles seen directly from above, and therefore are easier to distinguish. Their dataset was also rotated and centered, while our approach was trained to deal with this variance. All features of the face were therefore seen, but vehicles were seen from different views and therefore some features remained hidden. The last difference is, that even people are often not able to distinguish vehicles, especially with very low resolution and similar models.

8.4 Multi camera tracking experiment

A city with the cameras built in lamps does not exist yet, since the project is still in development. Therefore the representation for the whole city was not tested. However, as was described in the section 7.5.1, from representing one edge, it is a very small step to representing the whole city.

Only two videos of the same street at the same time from two different places was provided by the Brazilian party, that the concept was tested on. The reidentification part was described in the section 7.5.1. The detection and single camera tracking was described in the section 7.2 The score function described in the equation 17 was thresholded with 0.01. It should be stated, that because of the low fps, tracker sometimes created two tracks over similar trajectory. In such a case, only one track was used.

8.4.1 Evaluation measurement

The videos, that were provided, showed estimated 70 vehicles driving on the street during that time. The reidentification was tested each one of them. If the track was classified as the target vehicle correctly, that was measured as a true positive. It was not recognized, that was a false negative. If another track was falsely detected as the target one, that was

considered to be a false positive. The total accuracy was computed as the ratio of true positives to all tests.

8.4.2 Results

Out of 67 vehicles, that drove on the street, 59 were reidentified correctly. That corresponds to 88% accuracy. Out of the 8 badly reidentified vehicles, 6 were false negatives, where the detector did not detect the cars and 2 were false positives, where a different car was marked as the target one.

8.4.3 Comparison to state of the art

As mentioned before, most of the state of the art multi-camera vehicle tracking approaches are very domain specific. Since the problem solved in this thesis contains fisheye cameras, no other method could be directly implemented and compared. They are either only for highways [26, 79], rely on license plate recognition [3, 35] or have very narrow field of view [100].

9 Discussion

Artificial intelligence is said to be 20% developing models and 80% data management. Author of this thesis can only agree. Development of this thesis was very reliant on providing the data by the Brazilian party. However, only small amount of data were provided and very gradually. That is because the smart city project is still in development and only small amount of demo cameras was actually mounted. Combining lack of data with not reliable streaming, the process of development was sometimes frustrating.

The reidentification experiments could be performed only on the data available. The only videos taken at the same time are the two views of the same street. Furthermore, the videos had different lengths and missing frames, which made the process of synchronization tedious.

Because of the lack of data, experiments for the whole city, or even modeling intersections could not be done. However, the author of this thesis is confident, that the introduced models are accurate and robust enough to be able to reliably expand to a complicated graph of a real city.

Methods, such as image unwrapping into several less distorted images and detecting them separately, were explored and implemented but turned out to be a bad trade-off between performance and accuracy. This and several other approaches were not described in this work since they are too far from the assignment of the thesis.

10 Conclusion

This thesis combines solutions of several problems, such as object detection, single camera tracking, similarity metrics of vehicles, city representation and multi-camera tracking. Each of these tasks is very important and if solved poorly, could be a bottleneck of the whole process.

During development, the emphasis was put on computational speed and easy scalability of the project.

A thorough state of the art analysis of multiple fields of computer vision important for multi-camera tracking was described in the section 2. This allowed educated selection of used methods.

The wide angle lens model and it's parameters were found using calibration data and transformations between real-world coordinates, camera coordinates and pixel coordinates in images were found.

An object detection and optical flow tracker were proposed and implemented for faster dataset generation for similarity training. Another dataset was created using standard annotation tools for object detection training.

For object detection was proposed and implemented improved architecture of the neural network SSD [91] utilizing the video information and not just processing each frame independently. With the original SSD have been trained on the presented dataset. Experiments on fisheye camera domain show, that the proposed solution achieved 90.0% mAP compared to 63.2% state of the art SSD[91] after one-day training. After 4 days training the network achieved 91.6% mAP.

Google Facenet [124] was retrained from finding similarities between faces to vehicles achieving 80% classification accuracy. It was trained and tested on a custom dataset.

A single-camera tracker based on optical flow provided by the Good Vision company was used for clustering detections over frames.

Vehicles in the city were represented by a probabilistic state space based on Markov chain. This allows easy expansion of the city with new cameras being installed as well as dealing with probability from not observed places and uncertainty. Reidentification of vehicles was proposed and implemented based on the computed similarity and relationships between cameras.

A real-world experiment was performed achieving the reidentification accuracy of 88%.

10.1 Future work

This thesis is part of a long-term project with the Good Vision company. Author of this thesis will continue to develop and improve this project to the final state. With this being said, there are still areas to be improved and new approaches and models to be tried. Thanks to high modularization of this project, improved detectors, trackers and similarity metrics can be easily evaluated. Areas for further development include:

- Increase the size of both datasets for detection and similarity.
- Deploy the system on a real city and make it accessible for the reinforcement agencies.
- Instead of triplet loss for similarity training, implement and evaluate the recently introduced quadruplet loss [22].
- Train and evaluate newer neural networks for object detection, such as [89, 119].
- With increasing speed of segmentation networks create datasets and train segmentation models, which will improve tracking accuracy.

With newer and better neural networks introduced every year, this project will be gradually improved to keep up with state of the art methods in years to come.

References

[1] Good vision s.r.o. `https://www.goodvisionlive.com/`.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] Clemens Arth, Florian Limberger, and Horst Bischof. Real-time license plate recognition on an embedded dsp-platform. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[4] Salil P Banerjee and Kris Pallipuram. Multi person tracking using kalman filter, 2008.

[5] Björn Barz, Erik Rodner, Christoph Käding, and Joachim Denzler. Fast learning and prediction for object detection using whitened cnn features. *arXiv preprint arXiv:1704.02930*, 2017.

[6] Álvaro Bayona, Juan C SanMiguel, and José M Martínez. Stationary foreground detection using background subtraction and temporal difference in video surveillance. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 4657–4660. IEEE, 2010.

[7] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, September 1995.

[8] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3457–3464. IEEE, 2011.

[9] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3. Citeseer, 2011.

[10] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[11] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, and Stefano Nichele. Stereo vision-based vehicle detection. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 39–44. IEEE, 2000.

[12] Manuele Bicego, Andrea Lagorio, Enrico Grosso, and Massimo Tistarelli. On the use of sift features for face authentication. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 35–35. IEEE, 2006.

[13] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[14] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[15] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[16] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[17] Guimei Cao, Xuemei Xie, Wenzhe Yang, Quan Liao, Guangming Shi, and Jinjian Wu. Feature-fused ssd: fast detection for small objects. In *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*, volume 10615, page 106151E. International Society for Optics and Photonics, 2018.

[18] Nicholas Carlevaris-Bianco, Arash K. Ushani, and Ryan M. Eustice. University of Michigan North Campus long-term vision and lidar dataset. *International Journal of Robotics Research*, 35(9):1023–1035, 2015.

[19] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[20] Michael J Caruso and Lucky S Withanawasam. Vehicle detection and compass applications using amr magnetic sensors. In *Sensors Expo Proceedings*, volume 477, page 39, 1999.

[21] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[22] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proc. CVPR*, volume 2, 2017.

[23] Zhiwen Chen, Jianzhong Cao, Yao Tang, and Linao Tang. Tracking of moving object based on optical flow detection. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 2, pages 1096–1099. IEEE, 2011.

[24] François Chollet et al. Keras, 2015.

[25] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.

[26] Benjamin Coifman and Sivaraman Krishnamurthy. Vehicle reidentification and travel time measurement across freeway junctions using the existing detector infrastructure. *Transportation Research Part C: Emerging Technologies*, 15(3):135–153, 2007.

[27] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5):564–577, 2003.

[28] Jonathan Courbon, Youcef Mezouar, Laurent Eckt, and Philippe Martinet. A generic fisheye camera model for robotic applications. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1683–1688. IEEE, 2007.

[29] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1337–1342, 2003.

[30] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[31] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[32] A Daubaras and M Zilys. Vehicle detection based on magneto-resistive magnetic field sensor. *Elektronika ir Elektrotechnika*, 118(2):27–32, 2012.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[34] Mikel Diez Buil. Non-maxima supression. 2011.

[35] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, 23(2):311–325, 2013.

[36] Ahmed Elgammal, Ramani Duraiswami, David Harwood, and Larry S Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, 2002.

[37] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.

[38] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[39] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[40] Pierluigi Ferrari. Face recognition using tensorflow. `https://github.com/davidsandberg/facenet`, 2015.

[41] Pierluigi Ferrari. A keras port of single shot multibox detector. `https://github.com/pierluigiferrari/ssd_keras`, 2017.

[42] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. *arXiv preprint arXiv:1605.06457*, 2016.

[43] Gwennael Gate and Fawzi Nashashibi. Fast algorithm for pedestrian and group of pedestrians detection using a laser scanner. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 1322–1327. IEEE, 2009.

[44] Global Cloud Index (GCI). 7th annual gci forecast. explore global data center and cloud computing trends (2016 – 2021)., 2016.

[45] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[46] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[47] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[48] Susanna Gladh, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Deep motion features for visual tracking. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 1243–1248. IEEE, 2016.

[49] Rafael C Gonzalez and Richard E Woods. Digital image processing, 2012.

[50] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[51] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[52] Anselm Haselhoff and Anton Kummert. A vehicle detection system based on haar and triangle features. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 261–266. IEEE, 2009.

[53] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[55] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[56] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[57] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[58] Thanarat Horprasert, David Harwood, and Larry S Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Ieee iccv*, volume 99, pages 1–19. Citeseer, 1999.

[59] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[60] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[61] Timothy Huang and Stuart Russell. Object identification in a bayesian context. In *IJCAI*, volume 97, pages 1276–1282, 1997.

[62] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *arXiv preprint arXiv:1803.06184*, 2018.

[63] Michael Isard and John MacCormick. Bramble: A bayesian multiple-blob tracker. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 34–41. IEEE, 2001.

[64] Omar Javed, Khurram Shafique, and Mubarak Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 26–33. IEEE, 2005.

[65] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[66] Kinjal A Joshi and Darshak G Thakore. A survey on moving object detection and tracking in video surveillance system. *International Journal of Soft Computing and Engineering*, 2(3):44–48, 2012.

[67] Kiran Kale, Sushant Pawar, and Pravin Dhulekar. Moving object tracking using optical flow and motion vector estimation. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, pages 1–6. IEEE, 2015.

[68] Jinman Kang, Isaac Cohen, and Gerard Medioni. Continuous tracking within and across camera streams. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.

[69] Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[70] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[71] Vera Kettnaker and Ramin Zabih. Bayesian multi-camera surveillance. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 253–259. IEEE, 1999.

[72] Sohaib Khan and Mubarak Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, 2003.

[73] SamYong Kim, Se-Young Oh, JeongKwan Kang, YoungWoo Ryu, Kwangsoo Kim, Sang-Cheol Park, and KyongHa Park. Front and rear vehicle detection and tracking in the day and night times using vision and sonar sensor fusion. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2173–2178. IEEE, 2005.

[74] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[75] Irena Koprinska and Sergio Carrato. Temporal video segmentation: A survey. *Signal processing: Image communication*, 16(5):477–500, 2001.

[76] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[78] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easyliving. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, pages 3–10. IEEE, 2000.

[79] Reinhart D Kuhne. *Freeway control using a dynamic traffic flow model and vehicle reidentification techniques.* Number 1320. 1991.

[80] Karric Kwong, Robert Kavaler, Ram Rajagopal, and Pravin Varaiya. Arterial travel time estimation based on vehicle re-identification using wireless magnetic sensors. *Transportation Research Part C: Emerging Technologies*, 17(6):586–606, 2009.

[81] Andrew HS Lai, George SK Fung, and Nelson HC Yung. Vehicle type classification from visual-based dimension estimation. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 201–206. IEEE, 2001.

[82] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[83] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[84] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[85] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[86] Jinho Lee, Brian Kenji Iwana, Shouta Ide, and Seiichi Uchida. Globally optimal object tracking with fully convolutional networks. *arXiv preprint arXiv:1612.08274*, 2016.

[87] Zuoxin Li and Fuqiang Zhou. Fssd: Feature fusion single shot multibox detector. *arXiv preprint arXiv:1712.00960*, 2017.

[88] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–I. IEEE, 2002.

[89] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[90] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[91] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[92] BPL Lo and SA Velastin. Automatic congestion detection system for underground platforms. In *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, pages 158–161. IEEE, 2001.

[93] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[94] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[95] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[96] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[97] Vashisht Madhavan and Trevor Darrell. The bdd-nexar collective: A large-scale, crowsourced, dataset of driving scenes. 2017.

[98] Yasushi Mae, Yoshiaki Shirai, Jun Miura, and Yoshinori Kuno. Object tracking in cluttered background based on optical flow and edges. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 196–200. IEEE, 1996.

[99] Dimitrios Makris, Tim Ellis, and James Black. Bridging the gaps between cameras. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.

[100] Bogdan C Matei, Harpreet S Sawhney, and Supun Samarasekera. Vehicle tracking across nonoverlapping cameras using joint kinematic and appearance features. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3465–3472. IEEE, 2011.

[101] Stefan Munder and Dariu M Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

[102] Daniel T Munroe and Michael G Madden. Multi-class and single-class classification approaches to vehicle model recognition from images. *Proceedings of IEEE AICS*, page 50, 2005.

[103] OHTA Naoya. Optical flow detection by color images. *NEC Research and Development*, 97:78–84, 1990.

[104] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4694–4702. IEEE, 2015.

[105] Chengcheng Ning, Huajun Zhou, Yan Song, and Jinhui Tang. Inception single shot multibox detector for object detection. In *Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on*, pages 549–554. IEEE, 2017.

[106] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *European conference on computer vision*, pages 490–503. Springer, 2006.

[107] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak hypotheses and boosting for generic object detection and recognition. In *European conference on computer vision*, pages 71–84. Springer, 2004.

[108] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[109] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[110] Fatih Porikli. Inter-camera color calibration by correlation model function. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II–133. IEEE, 2003.

[111] Fatih Porikli and Oncel Tuzel. Multi-kernel object tracking. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1234–1237. IEEE, 2005.

[112] Cristiano Premebida, Gonçalo Monteiro, Urbano Nunes, and Paulo Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044–1049. IEEE, 2007.

[113] A Psyllos, Christos-Nikolaos Anagnostopoulos, and Eleftherios Kayafas. Vehicle model recognition from frontal view image measurements. *Computer Standards & Interfaces*, 33(2):142–151, 2011.

[114] Georges M Quénot. The'orthogonal algorithm'for optical flow detection using dynamic programming. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 3, pages 249–252. IEEE, 1992.

[115] Ali Rahimi, Brian Dunagan, and Trevor Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[116] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7464–7473. IEEE, 2017.

[117] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[118] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[119] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[120] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[121] Richard F Riesenfeld. Homogeneous coordinates and projective planes in computer graphics. *IEEE Computer Graphics and Applications*, (1):50–55, 1981.

[122] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[123] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[124] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[125] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135. ACM, 2016.

[126] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[127] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[128] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[129] Patrick Sudowe and Bastian Leibe. Efficient use of geometric constraints for sliding-window object detection in video. In *International Conference on Computer Vision Systems*, pages 11–20. Springer, 2011.

[130] Zehang Sun, Ronald Miller, George Bebis, and David DiMeo. A real-time precrash vehicle detection system. In *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 171–176. IEEE, 2002.

[131] Mriganka Sur. Brain processing of visual information. `http://news.mit.edu/1996/visualprocessing`, 1996.

[132] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[133] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[134] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[135] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[136] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to data mining. 1st, 2005.

[137] Gwenaëlle Toulminet, Massimo Bertozzi, Stéphane Mousset, Abdelaziz Bensrhair, and Alberto Broggi. Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis. *IEEE transactions on Image Processing*, 15(8):2364–2375, 2006.

[138] Christos Tzomakas and Werner von Seelen. Vehicle detection in traffic scenes using shadows. In *Ir-Ini, Institut fur Nueroinformatik, Ruhr-Universitat*. Citeseer, 1998.

[139] Cor J Veenman, Emile A Hendriks, and Marcel JT Reinders. A fast and robust point tracking algorithm. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 653–657. IEEE, 1998.

[140] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[141] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[142] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 842–849. IEEE, 2003.

[143] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.

[144] Stefan Wender and Klaus Dietmayer. 3d vehicle detection using a laser scanner and a video camera. *IET Intelligent Transport Systems*, 2(2):105–112, 2008.

[145] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[146] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[147] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.

[148] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):780–785, 1997.

[149] Web-Scale Training WST. Deeply learned face representations are sparse, selective, and robust. *perception*, 31:411–438, 2008.

[150] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

[151] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.

[152] Alper Yilmaz. Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[153] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[154] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[155] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[156] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126–2136. IEEE, 2006.

[157] Tao Zhao, Manoj Aggarwal, Rakesh Kumar, and Harpreet Sawhney. Real-time wide area multi-camera stereo tracking. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 976–983. IEEE, 2005.

[158] Qu Zhong, Zhang Qingqing, and Gao Tengfei. Moving object tracking based on codebook and particle filter. *Procedia Engineering*, 29:174–178, 2012.

[159] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.