1 Introduction

Estimated 80% of the human data is in form of images or videos. ***citace*** There is a lot of useful information hidden, but it is still very complicated to gain it. Most of the information is still labeled manually by people, who make mistakes and are expensive. There is an incredible need for automated video processing in many branches of the industry. 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.

*** It is shown in this thesis, that the introduced object detection approach is better on presented domain, than the standard approaches ***

1.1 Problem statement

This thesis has been implemented for the company Good Vision s.r.o to be used in many South American cities for local reinforcement agencies. The cameras will be mounted directly to street lamps and will provide surveillance data to the police. The cameras have 360 degrees view thanks to their very short focal distances.

If there was a crime committed and the person is driving away in a vehicle, an police officer marks the car and a set of algorithms introduced in this thesis will track the location of the vehicle in the city.

1.2 Overview of methodology

The whole thesis has been divided to subproblems and solved more or less separately. The model of the camera, it's parameters and transformations to the real world coordinates were be found as described int the section 3. The possibility of distributed computing directly in the cameras have been explored as described int the section 4. That included fast non deep learning set of algorithms for detection [91], tracking [5] and classification [45] running on CPU. This approach could not be used directly because of some problems mentioned in the section. However, it was used for semi-supervised dataset generation. The approach for frame decomposition into multiple non distorted images have been explored in section 7.6, but for computational reasons not used in the final product.

An annotation tool has been used as described in the section 7.7 for creating training and validation dataset. SSD[79] neural network architecture has been used for object detection. This network has been extended for an

additional input of temporal difference to better recognize moving objects and has been trained on NVIDIA GeForce GTX 1080.

Google Facenet [104], which was trained on a custom dataset from section 4 has been used as a metrics of similarity between detections in section 6.5. An overall model

1.3 Contribution

***probrat osobne s vedoucim ***

The author of this thesis implemented several modules for this project.

- The calibration of the camera and estimation of the vehicle position.
- Detector and tracker of vehicles using classical methods
- Semi-supervised data generator
- Improving, extending and training deep learning detector and connecting it with a provided tracker.
- Training a neural network for vehicle similarity
- Testing and comparing results.

implementing of classical model detections. using the facenet model for similarities. Improving the ssd model.

2 Related work

The computer vision field has made an incredible leap forward in 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 from previously known set of classes. This is a hard task because of the variance in lightning, pose, rotation, scale, as well as intra class variation. The detection task described in the section 2.2 is linked to the classification problem and all deep learning detectors use image classification networks.

Accuracy is usually 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 [73], ImageNet [28], or PASCAL VOC [31] have been created.

Before the invention of convolutional neural networks, other classifiers were used. Classifiers in general can be divided into parametric and non parametric methods. The non-parametric ones require no training phase and the decision is based directly on the data. Most common method is the Nearest Neighbor approach [11, 131]. 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 [12], adaboost [90], or the most common Support vector machine (SVM).

The classification pipeline of SVM is such that a set of features is extracted and an Support Vector Machine (SVM) is applied. These features can have many different forms and can also be combined. A histogram[18], Bag of features [70, 89], SIFT features [126, 10] or Haar features [87] can be used.

Convolutional neural networks (CNNs) are the state of the art in image classification. They have been introduced in 1990's[71], but only recently

In 2012 AlexNet [68] 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 is considered to be the beginning of deep learning in computer vision.

The paper ZFNet [130] in 2103 introduced some inside to how CNNs work by introducing De Deconvolutional Network 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 2013.

GoogLeNet/Inception won the ILSVRC 2014 with the incredible top-5 error rate of 6.67%. the architecture was based on LeNet [72], but introduced an inception module. this module eliminates all full-connected layers, greatly reducing the number of parameters. This network is used as a backbone in many object detection networks and has been used in this thesis as a

backbone for Facenet [104] described in the section 6.5

The second best network in ILSVRC 2014 was the VGG Net [107]. The depth of the network has been increased, but the number of parameters kept low thanks to very small 3x3 filters. The architecture is simpler than the Inception and it is widely used in transfered learning and as a detection backbones. This network has been used in this thesis as a backbone for the SSD detector[79] described in the section 6.2.

The Microsoft's ResNet[47] introduced a 152 layers deep architecture. they are able to train the network thanks to the introduction of the residual connections. Part of the information passes through the layers unchanged. This helps to solve the vanishing gradient problem. With the top-5 error rate 3.57% They surpassed the human accuracy winning the ILSVRC 2015.

The ResNet idea has been further developed. Wide ResNet[129] reduced the number of layers while widened the network. ResNeXt [125] 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 [50] connects each layer to every other layers in front of it. 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 Detection

2.2.1 Vehicle detection

There are many ways to detect vehicles, not just with cameras. One can detect changes in magnetic fields [27, 17] or a laser scanner [36].

Cameras are the most common sensor, but they can be also combined with a laser scanner [120, 94] or a sonar [64, 118]. Sometimes a stereo vision [9, 113] can be used to gain a better model of the environment.

Lot of research has been done for detection of vehicles and people thanks to the recent advancements in autonomous driving. Many datasets have been created [53, 84, 83, 15] for detecting vehicles, pedestrians and other objects from the vehicle's point of view. There is even a research for detecting vehicles by their shadow [114]. The state of the art in vehicle detection using cameras is detecting each image independently using techniques described in

the next section.

2.2.2 Object detection

In computer vision, the object detection is a specified task. The goal is to draw a bounding box around each object and assign it one class selected from a previously known set of classes. The accuracy is measured in mean Average Precision described in the section 8.0.1.

Before the rapid using of neural networks, various methods have been used for object detection. Haar features were used for detecting faces [45, 76, 116] and vehicles [109]. For general object detection, the background subtraction [91, 49] or optical flow [88, 95, 20] described in the sections 4.1 and 4.2. SIFT [81] HOG [38, 119, 134, 32, 26] are also being used.

The big advancements came with the introduction of region proposal networks [39]. The R-CNN [40] were the first to introduce this concept. R-CNN 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 %[16] the year before. However, R-CNN were very slow (47 seconds on GPU with the VGG16 [107] network), thus were far from realtime video analysis. It requires a full AlexNet forward pass for each of the around 2000 proposals.

Improved and faster version Fast R-CNN [37] 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 VGG-16 achieved 75.9% mAP on PASCAL VOC 2012 dataset with just 7 fps on GPU. This is much closer to processing a realtime video.

The YOLO[98] performs 45 fps, mAP 63.4 on VOC 2007. It splits the image in a grid and predicts only two bounding boxes and class probabilities for a grid. However it struggles with detecting more small objects close to each other and would not be a good detector for vehicles from street camera. However,here have been some improvements to this network [99, 100].

Region convolutional neural networks, which create proposals and then classify them, are still too slow. The Single shot multibox detector (SSD)

	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.

[79] based on [30] leaves out the region proposals with the 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 has been chosen for object detection in this thesis and will be described more in the section 6.2.

There has been lately many more architectures introduced, such as [77, 75, 25] and many more are coming.

Video is a sequence of frames. Most of the time for detection objects in a video, it is decomposed to different frames and each frame is detected independently [108]. This looses much of the information encoded in the video. background subtraction [42] or optical flow [88] as described in the section 4 can be used for detection.

Some newer work combines an optical flow information with neural networks **??**, but optical flow is expensive to compute, even though there is a convolutional network for optical flow estimation **??**. [61] 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 has been introduced in this thesis by combining the RGB input image with a 4th channel of temporal difference between frames and feeding it to a neural network. There is a research [4, 66] using temporal difference for detection and segmentation, but to my knowledge has not been combined with deep learning.

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 illumination changes, partial and full object occlusions and the realtime processing requirements [128]. Almost all trackers assume, that the fps 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 near by the tracked objects and tries

to estimate their movement. This is especially good for tracking low number of objects. The object can be represented as a single point [58], a bounding box [23, 93, 127, 29], or a silhouette [54]. Only the changes can be registered [58] or a robust reidentification [115] can be used, which is more robust to occlusions. A statistical representation can be connected with a Kalman filter [2] or a particle filter [133]. The movement is often estimated using sparse optical flow [58, 85]. With the recent deep learning advancements, tasks as tracking are also being solved with deep neural networks [8, 48, 41, 35, 74].

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. However, these approaches are more computationally complex, since they require object detection. The try to connect the bounding boxes into a sequence of tracks. The main methods use Jaccard overlap [112, 6] and optical flow [20]. This task can be complex because of the crossing tracks as well as false positive and false negative detections [57, 29].

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 [86]. 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 [63, 69, 132]. 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 [55, 92].

When the camera's fields of view don't overlap, the task becomes much more challenging. The camera positions can be either known [96] or unknown [86]. For reidentification, mean a posteriori (MAP) is estimated, giving the probability of the detected object being the same [55, 52]. [62] used a probabilistic Baesian model formulation with previously known transition functions. [59] explored this system for controllable movable cameras.

similarity - facenet...

Figure 3.1: Frame of the provided video

3 Fish-eye camera model

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

The cameras have been provided by the Brazilian party and no technical parameters are available. The model of the camera and it's parameters must be found. A set of improvised requested calibration images have been provided.

3.1 Scene localization

Before we find the model of the optics, we need to compensate for another hardware error of the camera. As can be seen in the image 3.1, the scene is shifted to the left down. It is not even circle, but rather an ellipse. This is due to manufacturing uncertainty and this error is different on each camera. Since this project will be easily scalable, and it is not convenient to measure and set the parameters manually, an universal algorithm for detecting ellipse has been introduced.

The algorithm is based on 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 it 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 8 bit scale is grater or equal to 1. [**?**]

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

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 an additional mask $E(s_x, s_y, a, r)$ of the ellipse as

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

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

$$C(M, s_x, s_y, a, r)=\sum_{x=0}^{W-1}\sum_{y=0}^{H-1} E_{x,y}(s_x, s_y, a, r)\cdot(1-M_{x,y})+(1-E_{x,y}(s_x, s_y, a, r))\cdot M_{x,y} \tag{2}$$

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

Computing in the cartesian coordinates is not very useful for optics. Instead, the world coordinates are chosen to be spherical and the frame coordinates are chosen to be polar. The world coordinates are in 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 as follows:

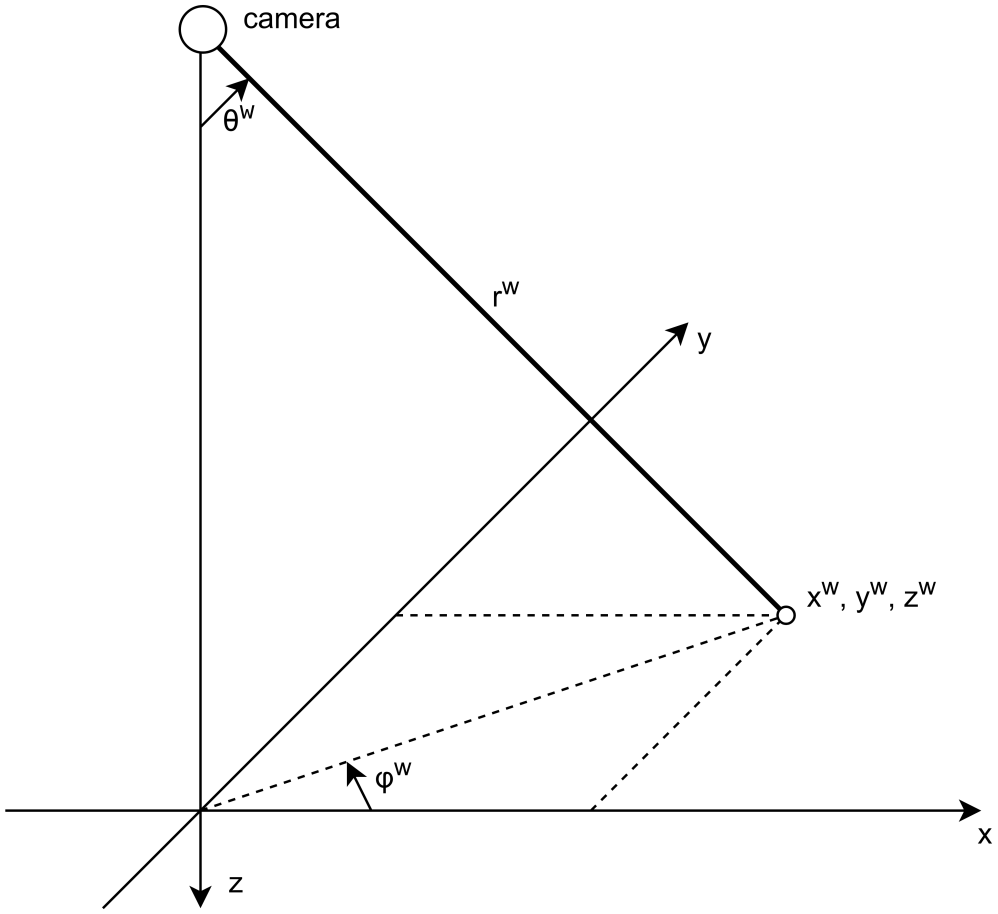


Figure 3.2: The spherical coordinates of the world

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

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:

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

Next, we need to find the transformations between the world spherical coordinates r^w, θ^w, ϕ^w and the frame polar coordinates r^f, θ^f, but there are some nice properties:

- The φ are the same, i.e. $\varphi^w = \varphi^f$.
- The transformations do not depend on r^w. The projection depends only on the direction, not on the distance from camera.

With this knowledge, we need only to find the transformation of θ^w and r^f. We need to find a function f, such as

$$\begin{aligned} \theta^w &= f(r^f) \\ r^f &= f^{-1}(\theta^w). \end{aligned} \tag{3}$$

There are many models for finding f.

- 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, we need to find the one parameter 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 picture. These marks are exactly 2 meters apart and are enough to estimate the function $f(r^f)$.

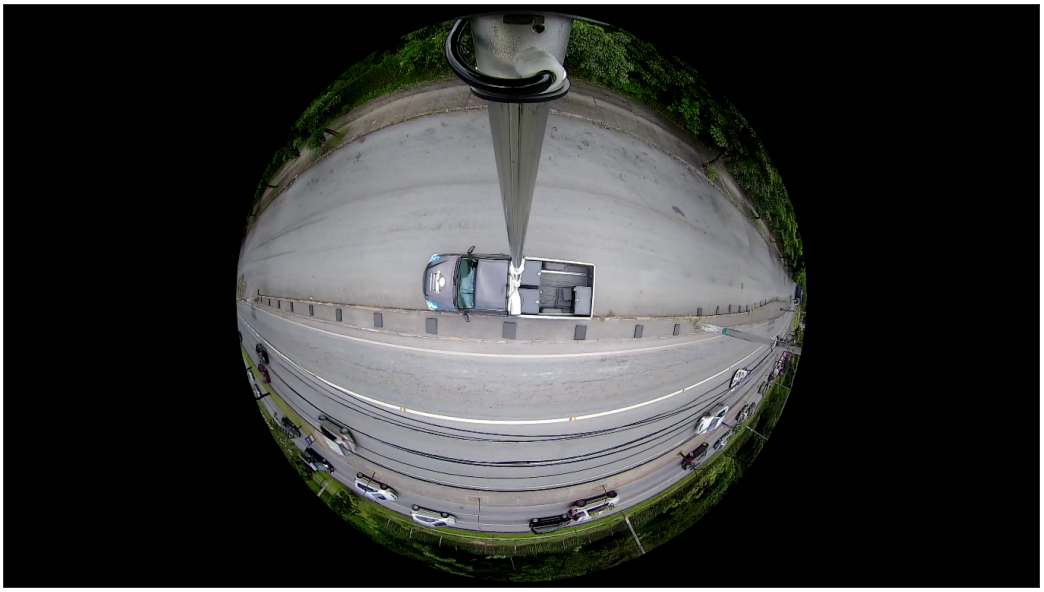


Figure 3.3: The provided calibration data

3.2.1 Linear model

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

$$\theta^w = f(r^f) = FOV \cdot r^f \tag{4}$$

Fitting of the model is shown in the fig.3.5
The linear model somehow estimates the real one, but not that well.

3.2.2 Tangent model

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

$$\theta^w = f(r^f) = \theta^w \cdot FOV, \tag{5}$$

Fitting of the model is shown in the fig.3.5
This model represents the camera optics much better and was chosen to be the final one.

3.3 The city coordinate system

There is many ways how to represent real world for representing the car in the city. The most obvious one would be represent the position by longitude,

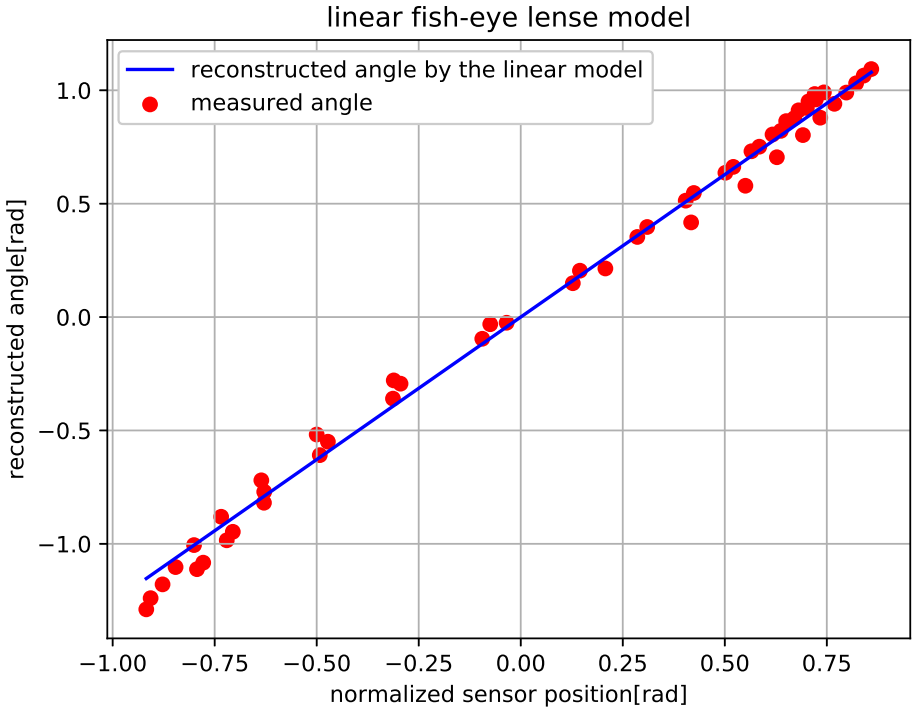


Figure 3.4: The linear model of the lens

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 center. 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, which are in form of $(x, y, 1)^T$.

A point from a camera coordinate system (x^w, y^w) is transformed to the city coordinate system (x^c, y^c) as

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

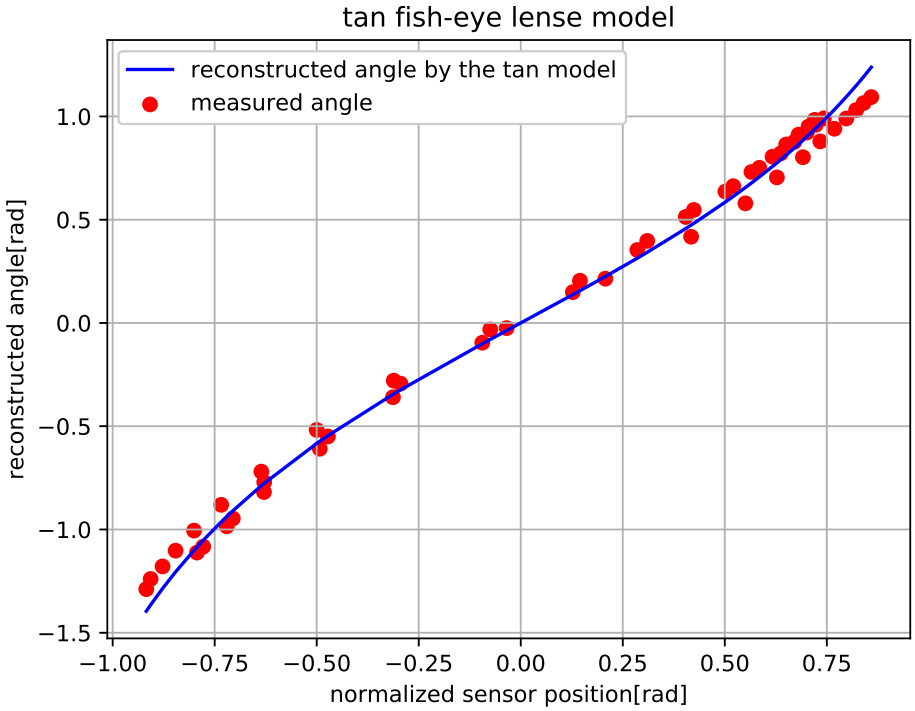


Figure 3.5: The linear model of the lens

4 Dataset generation

A vehicles detection and tracking without using deep learning has been explored. For several reasons described in this section it has not been used for the final product, but allowed a much faster video annotating, than standard methods and was used to create dataset for training similarity between vehicles described in the section 7.5

The final system needs be easily scalable and a particular system architecture has been 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 a 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 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 non deep learning approach for detection, tracking and classification, that could run on CPU.

4.1 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 [91]. 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 does not work very well and some improvements need to be made.

For the background subtraction procedure, a model of the background has to be found. For our purposes we need to know, how the road looks like without any vehicles and people. This can't be done by simply waiting for such a case, because the traffic is usually quite high. Instead, we need to figure out the background from multiple frames.

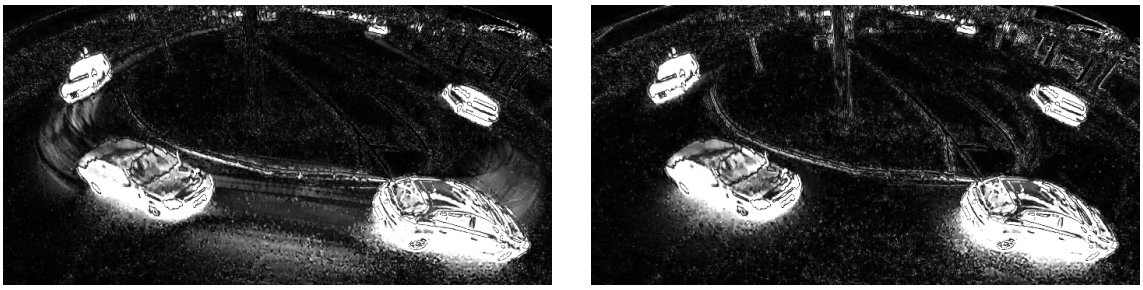
[49, 135]

The algorithm has been implemented in opencv [13]. The background is usually created by the mean over several images called running gaussian average [123]. 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 and the lane. In places, where vehicles drive, are colored lines as shown in the figure 4.1a. When computed the difference from a video frame to such a background model, as shown in the figure 4.2a, 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 just the first frame F_0. The background in next iteration is just the weighted sum of the current frame and the background model in the previous iteration.

(a) Mean model (b) Median model

Figure 4.1: Background model created by the mean and the median approach.



(a) Mean (b) Median

Figure 4.2: The difference between the frame and a background shown in a gray-scale.

$$B_n = \alpha \cdot F_n + (1 - \alpha) \cdot B_{n-1} \tag{6}$$

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

An improved way of acquiring background model has been used, 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 much better estimation of the background [80, 24]. 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 is simply

compute the median over all the images from the queue. The complexity is $O(N \cdot K \cdot log(k))$. The K has been set to 35. The histogram of a particular pixel position over the queue is show in the figure 4.3.

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

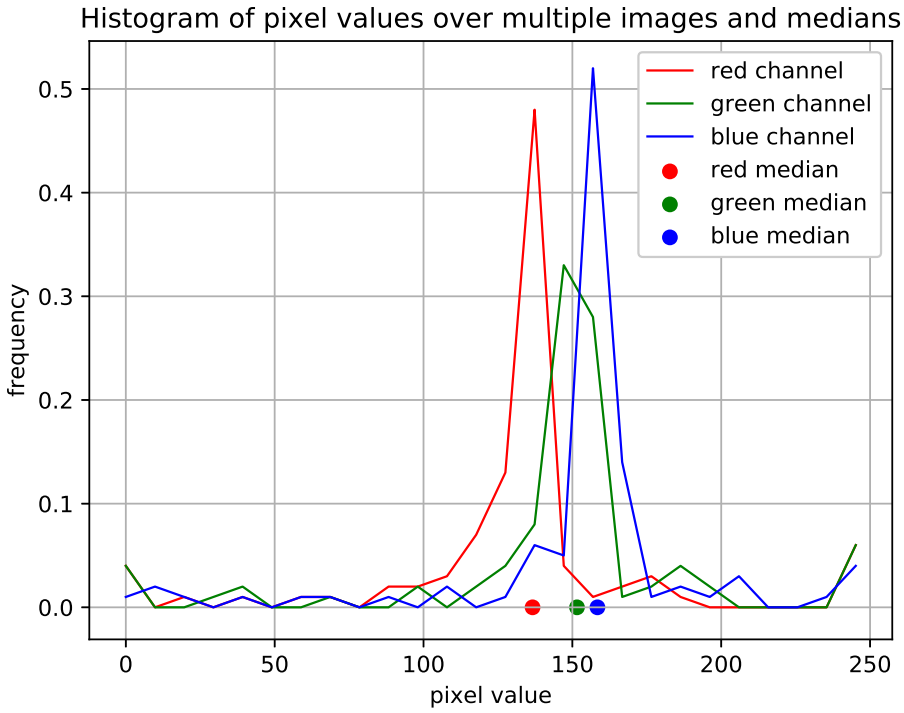


Figure 4.3: The histogram of a particular pixel over 100 images with computed medians

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 realtime video.

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

(a) A frame for detection (b) The detections and areas of contours.

Figure 4.4: The background subtraction detection algorithm.

probability of the pixel belonging to the car. This also helps to detect gray and black cars, which have similar color to the road. Another advantage is, that this greatly reduces noise and helps to detect vehicles as whole.

This differential image is thresholded and a mask is obtained as shown in the figure 4.4b. 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 not work very well 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 very fast detection algorithm and when having perfect conditions, it is very accurate and The bounding boxes are more precise than most deep learning approaches.

Unfortunately it has many downsides.

- Moving trees and their shadows create false positives.
- Overlapping vehicles are detected as only one.
- It works bad 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 correct setting of hyper-parameters.

Most of these points relate to changing background. Especially if the scene is partially cloudy and the lightning 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 **??**.

4.2 Optical Flow tracking

The detections have been described in section 4.1. 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 has been implemented in opencv[13] for extending the background subtraction algorithm to tracking. Optical flow [5] 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 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 [82].

***desctiption, equations, dense/sparse? ***

This algorithm is not used on whole image. That would be computationally too expensive and would not be feasible for limited computational resources and realtime 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 through 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 [106] algorithm, which is an improved version of the Harris corner and edge detector [44]. It tries to find some features, that have high gradient and will be easier to track, rather than selecting just the muddle 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 can not be matched with a next frame and stays on 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 has been 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 movement with a low pass filter.
- The optical flow points are forced to the middle of the bounding box. This solves the problem of the detection being put on the front of the car while coming and due to noise later being outside the bounding box, when the vehicle is viewed from a side.

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.

4.3 classification

In the frames, there are many objects detected and need to be classified. The most important distinction is between a person and a vehicle.

A simple classifier has been introduced using 87 distinct Haar features. Each Haar feature is a difference of average brightness level in two different rectangles in the image. The SVM classifier has been trained on 8144 images of cars and 10567 images of people. The test set split ratio was 0.2 reaching the accuracy 0.842. That is quite a good score without using neural networks, but not good enough for the final project.

The training and testing data have been acquired by developed semi-supervised data annotator based on the detector and tracker introduced in the section 4.4.

4.4 Semi-supervised data 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. Video is usually around 25 frames per second, so to annotate a minute of video means annotate a 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 can not be used for the final product, mainly because of the problems described in the section 4.1. 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 a dialog as shown in the figure 4.5. That includes the whole track, which is annotated through one click.

Because of the not perfect classifier, it is not used. Instead, the annotator selects the class or to throw the whole track away.

and can alswhich detections will be used

5 Deep learning

*** teorie k obecnym neuronovym sitim, popis vrstev. Transfered learning. Popis VGG-16 a Google Inception. Hodne citaci***

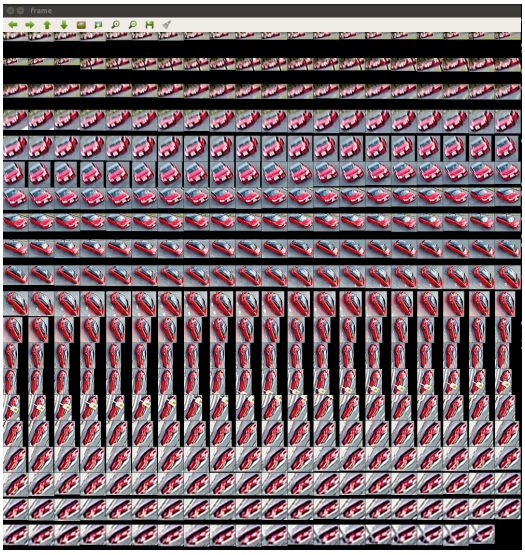


Figure 4.5: The dialog from the annotation tool.

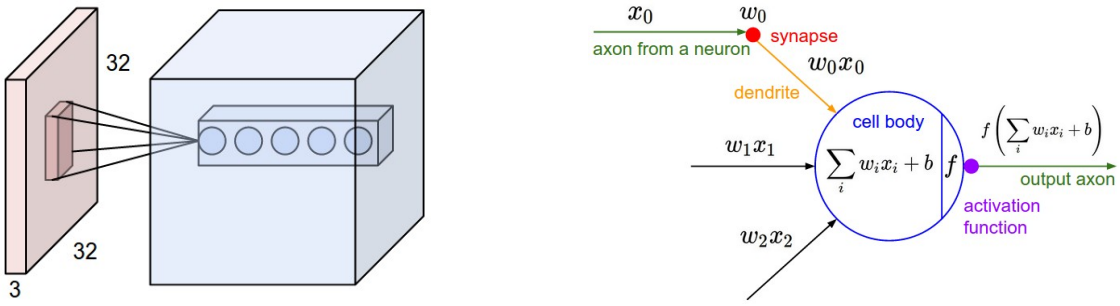
Deep learning is used for artificial intelligence models, which use more non-linear layers of computation. Most common are the Artificial neural networks. They are a computational model inspired by a brain, that can be thought various tasks from image classification [110] to speech processing [43]. In this section we will focus on neural networks for image processing.

5.1 Inspiration by biology

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 another 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 ones and their size can change by learning.

The basic element 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[102], which is a well known classifier. The basic function of a neuron can be described as $f(\overrightarrow{\omega} \cdot \overrightarrow{x} + b)$, where $\overrightarrow{x}$ is the input of the neuron, $\overrightarrow{\omega}$ 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 a weights and biases, that the neural networks performs the required task well.

When an architecture is created, the neural network needs to learn how



(a) An example of a convolutional layer

(b) Model of a neuron.

Figure 5.1: Examples of neural networks concepts from [60].

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 5.4.

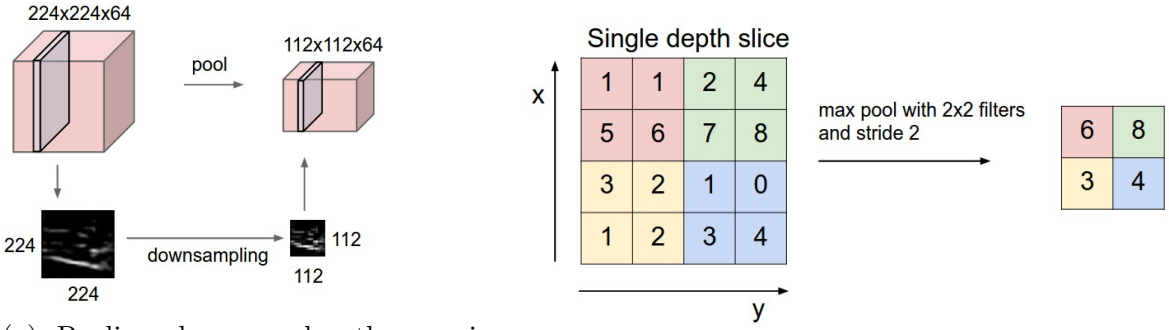
5.2 Layers

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 is more types of layers depending on the neuron's function and the way they are connected. The information describing this defines the neural network architecture.

5.3 Convolutional layer

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 $f(\overrightarrow{\omega} \cdot \overrightarrow{x} + b)$ shown in the figure 5.2b. Thanks to their regular structure and sharing parameters they perform a convolution function, where they look for a 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 to this phenomena has been described in [130].



(a) Pooling downsamples the previous layer.

(b) Max pooling operation.

Figure 5.2: Examples of pooling concepts from [60].

5.3.1 Pooling layer

The image has usually big resolution, but the information gained can be described in 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[103] is an important part of almost all convolutional neural networks and it can not be trained since it does not have any parameters.

5.3.2 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 operation and is usually performed at the last layers of the network, where the layers are smaller. They are used as the final classifier, each neuron representing one class.

5.3.3 Overfitting and dropout layer

The neural network has many parameters and high Vapnik-Chervonenkis dimension **??**, therefore it is more prone to overfitting. The network should from the training set of examples figure out 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 phenomena

in machine learning. Usually the way is to select a simpler classifier, that can still handle the problem 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 it's value to the output or returns a zero. This forces the network to build more robust connections. This layer is active only during training.

5.4 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 informations 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

$$L(\overrightarrow{y}) = -\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{7}$$

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

$$\omega_{t+1} = \omega_t + \frac{\partial L(\overrightarrow{y_t})}{\partial \omega} \tag{8}$$

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

5.5 Frameworks

Artificial neural networks require sometimes billions of very simple operations. Although they have been known for many years[121], only recently they have had a great success. That is due 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 [1], Theano [7] or CNTK[105] by Microsoft, from which the TensorFlow is the most common and the reidentification part of the thesis described in the section 7.5 has been implemented in this framework.

There are also some libraries such as Keras [21] and Caffe [56] that make implementing deep learning in C++ or python much easier. The SSD object detection network described in the section 7.3 has been implemented in Keras.

6 Classification, Detection and Reidentification networks

6.1 Neural networks for classification

The Neural networks have been the state of the art for the classification. Furthermore they are used as the backbone for detection networks.

6.1.1 *** conv layers, ..., ?

6.1.2 VGG

The VGG[107] 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[98], ARTOS[3], SSD [79] or Faster R-CNN [101]. In this thesis, the VGG network has been used as the SSD backbone and has been adjusted for the detection task in the section 7.3.

The VGG is an image classification network. It takes an input image and produces a probabilities for each predefined class. It has been inspired by [22] and [68].

The input resolution of the network is fixed to 224×224. The idea of the network is to use a very small filters 3×3 (which is the smallest size to capture the notion of left/right, up/down, center)[107]. Two of these filters stacked on top of each other create a receptive field of 5×5 and three have the receptive field of 7×7. With sharing parameters, the three layers has almost half the parameters, than the layer with 7×7 filters. This is different from [68] with 11×11 filter in the first layer and [130] with 7×7.

To bring more non-linearity, the VGG16 uses a filters with 1×1 filters. This has been used also in [71] as a network in a network.

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

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. [107]

Different VGG architectures have been evaluated designed in [107] and shown in the figure 6.1.

6.2 SSD network for detection

The state of the art in detection is at the time the Single Shot MultiBox detector [79]. This network provides accurate realtime 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

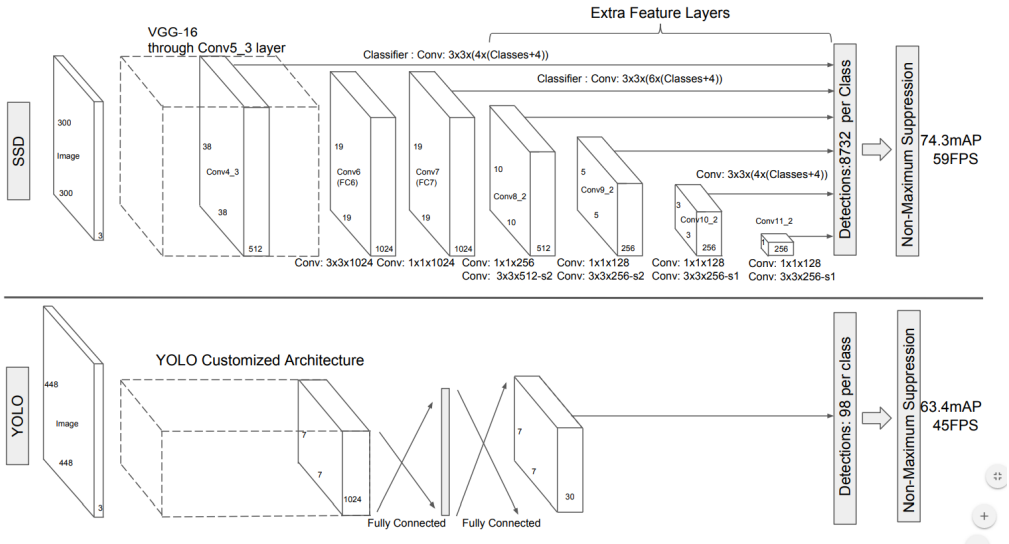
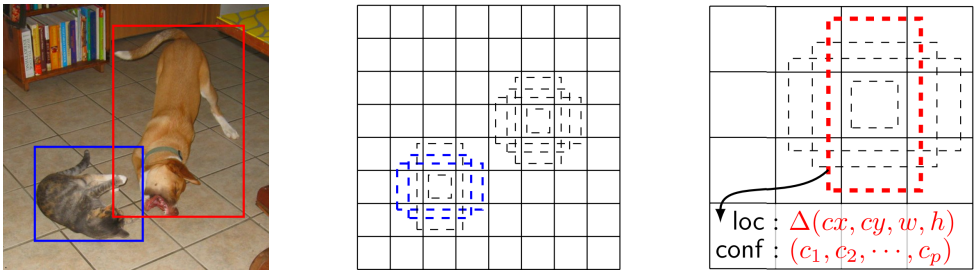


Figure 6.2: Comparison of the SSD[79](300x300) and YOLO[98](448x448) architectures.

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 have been added on top of the base network show in fig. 6.2. 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 boxes offsets are measured relative to the each feature map location.



(a) SSD prediction (b) 8×8 feature map. (c) The predicted offsets.

Figure 6.3: The background subtraction detection algorithm.

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 bounding boxes. They differ in shape and scale as shown in the figure 6.3c. 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 6.3c with fixed positions.

A prediction of a probability of each class and predicted offset of the the bounding box is computed. The offsets are given relative to the associated fixed position of the bounding box as shown in the figure 6.3b. 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.

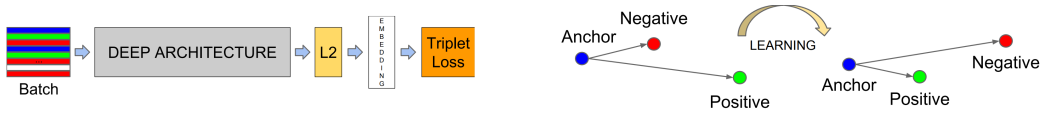
6.3 Loss

*** todo ***

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 [30]. 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 on training. Most of the predicted boxes are usually negatives. The backpropagation is on all the boxes matched with



(a) The model structure

(b) The learning objective

Figure 6.4: The Facenet [104].

ground truth box as well as the negatives with lowest score for 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

*** todo ***

6.5 Facenet for reidentification

Google Facenet [104] 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 a 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 this 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.

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 [130] (with the input resolution 220 × 220), Google Inception [110] (224 × 224, 160 × 160, 96 × 96) and mini Inception (165 × 165) and tiny Inception(140 × 140).

The [110] 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, than classification, where a finite number of categories is previously known. There have been been attempts [124, 111] for 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 $\alpha = 0$

$$\begin{aligned} &\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \\ &\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \end{aligned} \tag{9}$$

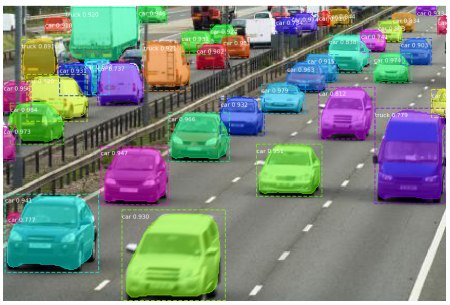
The α is a constant, that can enforce a margin. T is a set of all possible combinations of images in the training set.

The loss, that is being used for training according to [110] is

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+. \tag{10}$$

After 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 have been recently some improvements of triplet loss in form of quadruplet loss [19], that enforces grater margins among classes.



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

(b) Mask R-CNN on a fish-eye has problems recognizing vehicles from the top

Figure 7.1: Examples of Mask R-CNN network [46]

7 Implementation

7.1 Detection and tracking on CPU

7.2 Mask R-CNN segmentation

First, couple of approaches have been explored, but not to a great detail. One of them is Mask R-CNN[46]. 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 **??**.

Since there are no datasets for training and evaluating segmentation from the top-places fish-eye street cameras the performance has not been measured. Transfered 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 realtime application and this approach has not been 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.

7.3 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 fish-eye, it is not possible to simply use a off the shelf network trained for example on ImageNet or on PASCAL VOC and use it right away.

For the detector has been chosen the SSD network [79] described in the section 6.2. It is the state of the art realtime detector outperforming the R-CNN[101] and YOLO[98] in speed and accuracy as described in the section 2.2.2.

the SSD uses VGG-16 for the base network, which has been described in the section 6.1.2. 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 fish-eye view, objects, especially bicycles and motorcycles, can appear very small. It is sometimes problem for humans to detect them on 1024×1024 image, therefore the version with higher resolution has been chosen.

A working network implemented in keras[34] has been used as a starting point, but some modifications have been made.

7.3.1 Temporal difference

As described in the section 2.2.2, the standard pipeline for video detection is detecting each frame independently. This looses 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:

$$D(F^T_{i,j}) = max(255, \sum_{c=1}^{3} |F^t_{i,j,c} - F^{t-1}_{i,j,c}|) \tag{11}$$

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[117] and opencv[13] 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 loosing the rgb information. The

(a) A frame from the street camera.

(b) Temporal difference highlights moving objects.

Figure 7.2: Temporal difference helps to detect moving objects.

introduced approach keeps both the information by feeding both to the network.

The SSD network has been 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 has been temporarily called RGBD network. The D stands for the difference and should not be confused with commonly used distance.

In some cases, we don't want to detect parked cars. This is in a case of a road with 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.

7.3.2 Architecture

The architecture is similar to the original SSD512 architecture, but some changes have been made.

The input layer has been extended for the differential channel as described before.

According to some literature [14], SSD might have problem with detecting small objects. Therefore one more feature layers has been added with a small scaling factor 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.

7.3.3 Dataset

A scene annotating has been introduced in the section 4.4. However this approach could not be used for SSD training and was used only for similarity training described in the section 7.5. Even though it produces no false positives, sometime 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 were 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 has been provided by the goodvision 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 has been shot from a slope on a car, as shown in the figure 3.3. Only two cameras in lamps were working and recording video in the time of writing this thesis, since the whole project is still in development.

The rgb images and the temporal difference have been extracted from the total of 17 scenes a second apart and the images without traffic have been deleted. There is need for many scenes for the diversity of the roads, surroundings and lightning conditions.

A detector can easily overfit on a scene by learning it's background, therefore the test data have been 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.

7.3.4 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.

All the data augmentation techniques used during training are:

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

7.3.5 Training

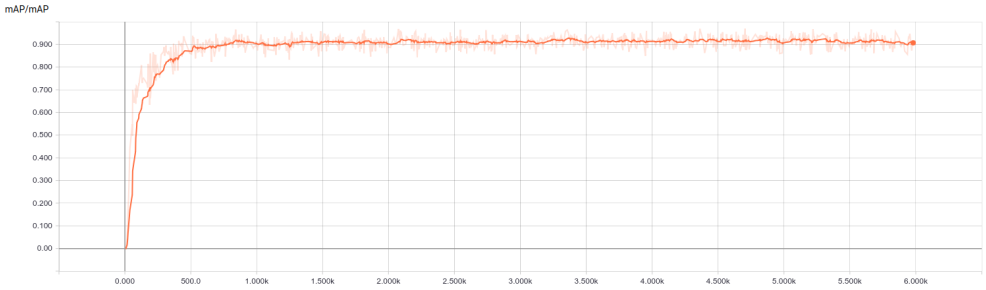
It is not possible to use pretrained model on ImageNet[28] or COCO [78], because of the changed architecture of the network. A pretrained weights could not be used for transfered learning. An attempt has been made to pretrain the model on Youtube-boundingboxes dataset[97], but only static videos would have to be selected because of the differential input channel and the scenes are too different from the fish-eye traffic camera.

The images were shuffled before each epoch. The minimum overlap between anchor and a label has been set to 0.5, the maximum overlap between anchor and background label has been 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.

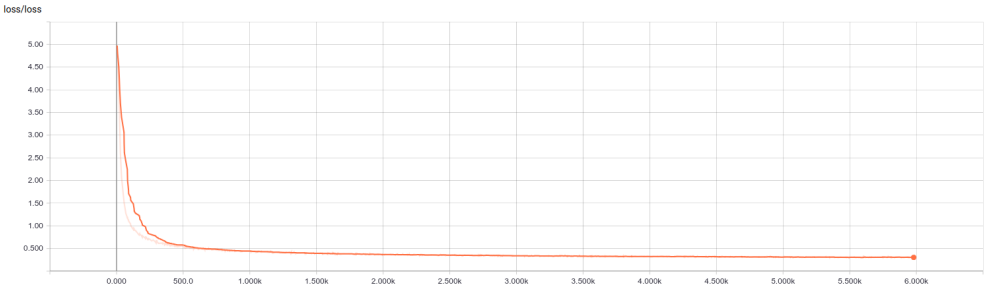
A custom tensorboard callback and mAP computation had to be implemented for observing the training progress and evaluating the model. Another features were added, such as tracking the detection ratio or computing the best threshold from the mAP curve.

The Adam optimizer [65] has been used with parameters: The learning rate 10^{-3}, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, decay = 0.0005$. These parameters

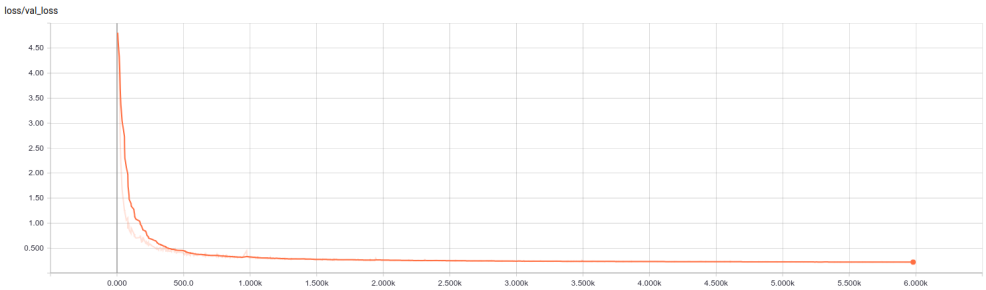
have been selected by [34] and were not changed. The training process is shown in the figure 8.2



(a) The mean average precision on validation set.



(b) The loss on the training set.



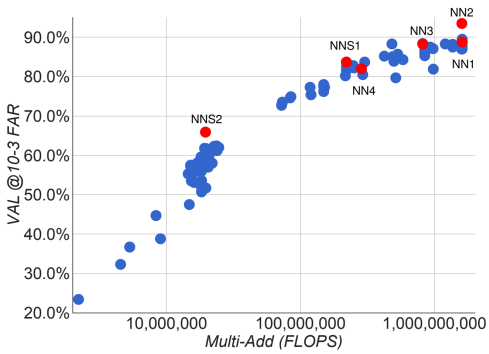
(c) The loss on the validation set.

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.

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

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

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



(b) Comparison of different DNN architectures.

Figure 7.4: Comparison of various facenet[110] base networks evaluated on Labeled faces in the wild[51] and YouTube faces[122] datasets.

7.4 Tracking

7.5 Similarity

The neural network Facenet [104] was chosen as the metrics of similarity among vehicles. As described in the section 6.5, it is designed to compute 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 an 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.

As a starting point, TensorFlow implementation of the network [33] has been used. The network stayed unchanged, but the data handling, tensorboard callbacks and evaluation had to be implemented.

7.5.1 Dataset

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 **??** is not possible, because their images look too different, from those in the fish-eye street camera.

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

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 does. This dataset has been created by a semi-supervised annotation tool based on background subtraction and optical flow described in the section 4.4.

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

7.5.2 Problems with the dataset

When the facenet network was trained on this dataset, it did not work at all. There are 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 have been 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.

7.5.3 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 a different scenes into one class. This is a very time demanding process, but the results are worth it.

7.5.4 Evaluation

Evaluating the similarity task has been transformed to a 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.

7.5.5 Training

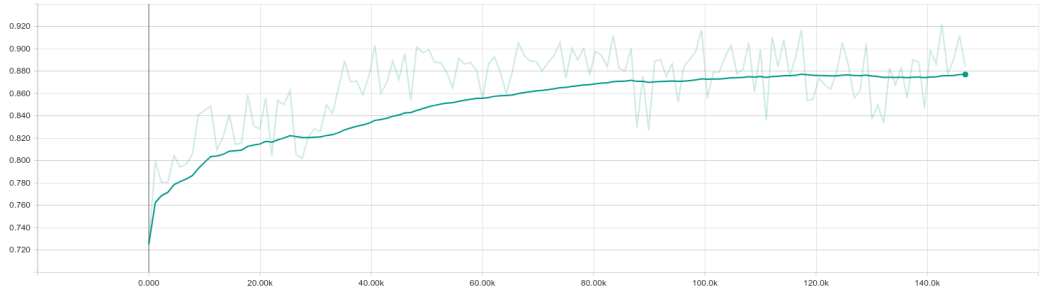
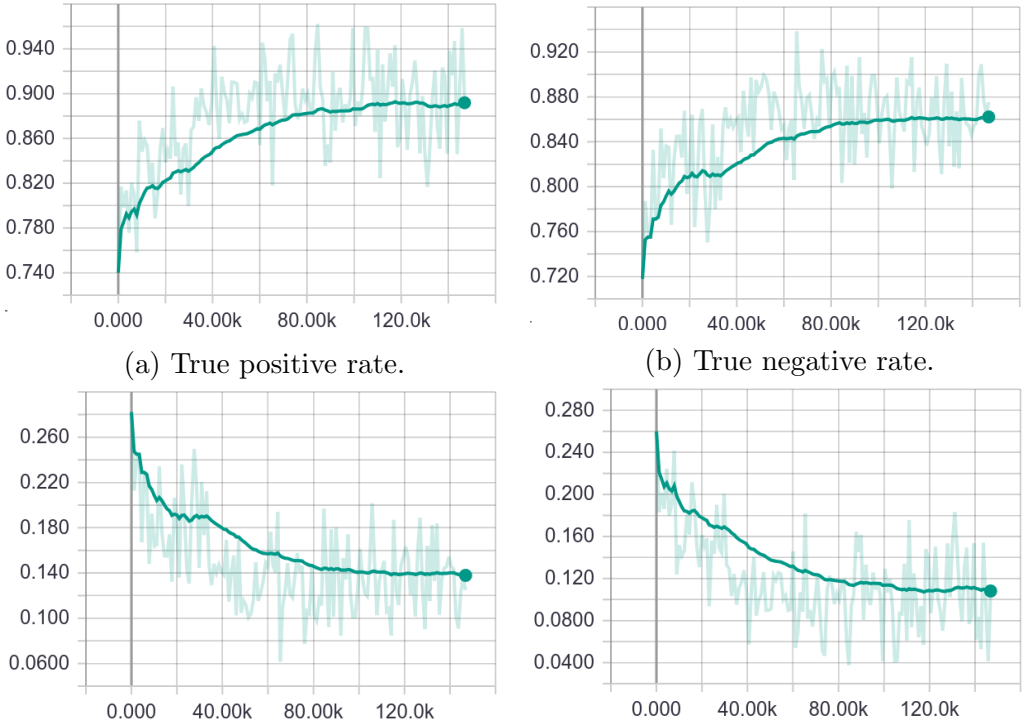


Figure 7.6: Accuracy during the facenet training.

To extend the dataset, some augmentation techniques have been used, such as rotation of the vehicles by 90°. Transfered learning has been used and the network pretrained 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 vehicles, that the network has not seen before.

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

Table 2: Final results of the facenet training.



(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

7.6 Image decomposition

7.7 Dataset

standford [67] cars dataset

8 Evaluation

8.0.1 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 have been matched. With lowering the threshold, it can only increase, since more ground truth bounding boxes will be matched.

$$P = \frac{TP}{TP + FP} \tag{12}$$

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

$$R = \frac{TP}{TP + FN} \tag{13}$$

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 over all classes is called mean average precision(mAP).

$$mAP = \frac{1}{|C|} \sum_{c \in C} AP_c \tag{14}$$

where C is the set of classes.

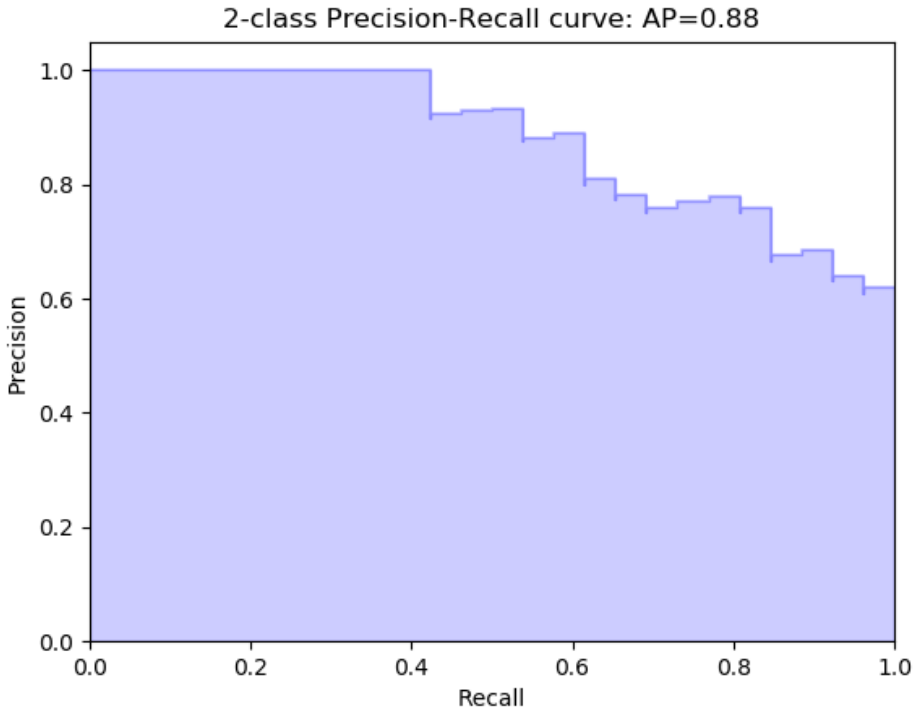
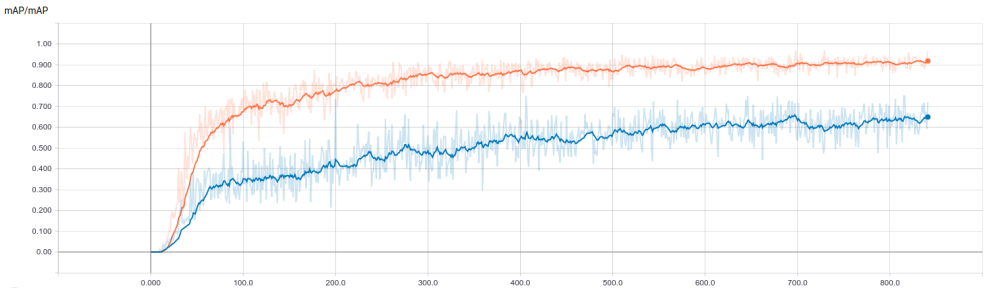


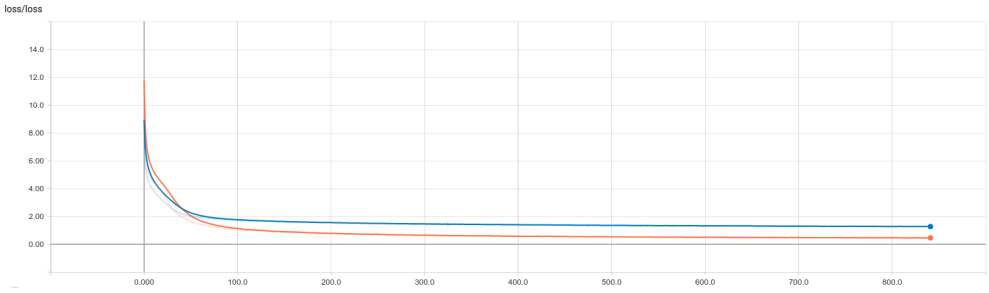
Figure 8.1: Example of precision-recall curve

8.1 Object detection

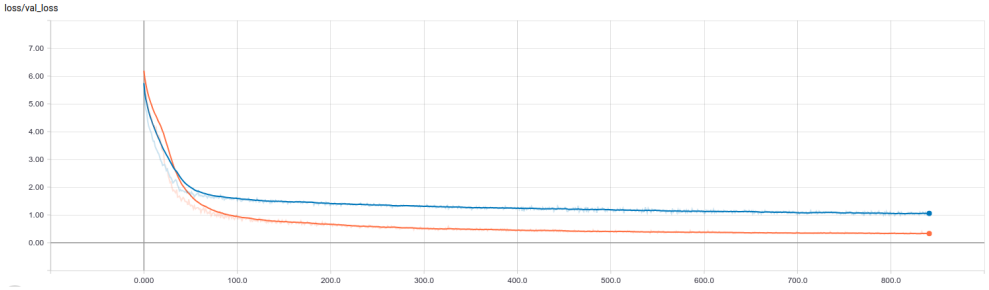
The implemented detector described in the section 7.3 has been compared with the state of the art SSD [79]. Because using pretrained SSD on ImageNet would perform badly because of the domain specifications, the models have been trained with the same parameters on the same dataset.



(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 deciding, that the difference layer greatly helps, the rgbd network was trained for 4 days. The final comparison is shown in the table 3.

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 3: Results on the custom dataset from the section 7.7.

References

[1] 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.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

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

[36] 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.

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

[38] 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.

[39] 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.

[40] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[41] 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.

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

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. *CoRR*, abs/1803.06184, 2018.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

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

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

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

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

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

[76] 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.

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

[78] 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.

[79] 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.

[80] 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.

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

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

[83] 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.

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

[85] 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.

[86] 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.

[87] 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.

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

[89] 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.

[90] 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.

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

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

[98] 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.

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

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

[101] 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.

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

[103] 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.

[104] 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.

[105] 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.

[106] 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.

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

[108] 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.

[109] 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.

[110] 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.

[111] 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.

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

[113] 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.

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

[115] 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.

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

[117] 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.

[118] 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.

[119] 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.

[120] 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.

[121] 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.

[122] 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.

[123] 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.

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

[125] 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.

[126] 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.

[127] 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.

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

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

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

[131] 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.

[132] 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.

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

[134] 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.

[135] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.