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.

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 [46], tracking [1] and classification [23] 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 5.1, but for computational reasons not used in the final product.

An annotation tool has been used as described in the section 5.2 for creating training and validation dataset. SSD[35] neural network architecture has been used for object detection. This network has been extended for an additional input of frame difference to better recognize moving objects and trained on NVIDIA GeForce GTX 1080.

Google Facenet [50], which was trained on custom dataset from section 4 has been used as a metrics of similarity between detections in section 6.2.

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. The ImageNet [14] competition has been for many years the

The biggest driving force has been he use of convolutional neural networks. ****

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.

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 [33], ImageNet [14], or PASCAL VOC [15] 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 [4, 63]. 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 [5], adaboost [45], 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[10], Bag of features [30, 44], SIFT features [61, 3] or Haar features [42] can be used. Each Haar feature is a difference of average brightness level in two different rectangles in the image.

In 2012 AlexNet [29] 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 [62] 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 [32], but introduced an inception module. this module eliminates all full-connected layers, greatly reducing the number of parameters.

The invention of deep convolutional neural networks [29, 31] has improved the classification accuracy incredibly.

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 [13, 9] or a laser scanner [17].

Cameras are the most common sensor, but they can be also combined with a laser scanner [59, 47] or a sonar [27, 57]. Sometimes a stereo vision [2, 54] 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 [25, 40, 39, 7] 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 [55].

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 5.0.1.

Before the rapid using of neural networks, various methods have been used for object detection. Haar features were used for detecting faces [23, 34, 56] and vehicles [53]. For general object detection, the background subtraction [46, 24] or optical flow [43, 48] described in the sections 4.1 and 4.2. SIFT [37] HOG [19, 58, 64, 16, 12] are also being used.

The big advancements came with the invention of region proposal networks [20]. The R-CNN [21] were the first to introduce this concept. It consists of two neural networks, one to propose the regions of interests and the second one to classify them. Their performance was mAP of 53.3% on PASCAL VOC 2012 dataset. This was a huge success compared to the mAP of 43.3 %[8] the year before. However, R-CNN were very slow (47 seconds on GPU with the VGG16 [52] 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 [18] 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 VOC 2012 dataset with just 198 ms on GPU. This is much

...

2.3 tracking

...

2.4 reidentification

When positions and orientations of the cameras are not known, the location and speed of the detected objects is used for obtaining the spatial relationships among the cameras [41, 49]. The key to reliable reidentification is to correctly model the relationships among cameras, as well as to find a similarity metrics of the detected objects. The detected objects can look very differently on different scenes thanks to different scaling, rotations and lighting conditions. The lighting can be estimated and compensated [26]...

3 Fish-eye camera model

Figure 3.1: Frame of the provided video

For correct estimation of the position of detected objects, it is crucial to understand the camera optics and sensor. We need to find the transformations from the camera pixel position to the real world position.

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:

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

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:

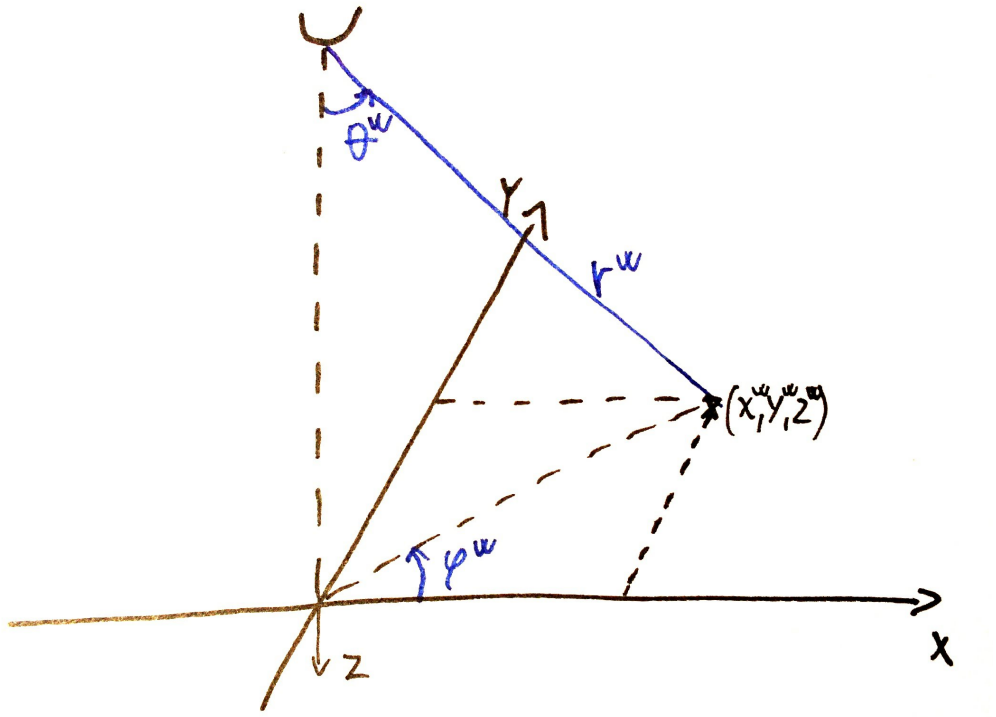


Figure 3.2: The spherical coordinates of the world

- 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.

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

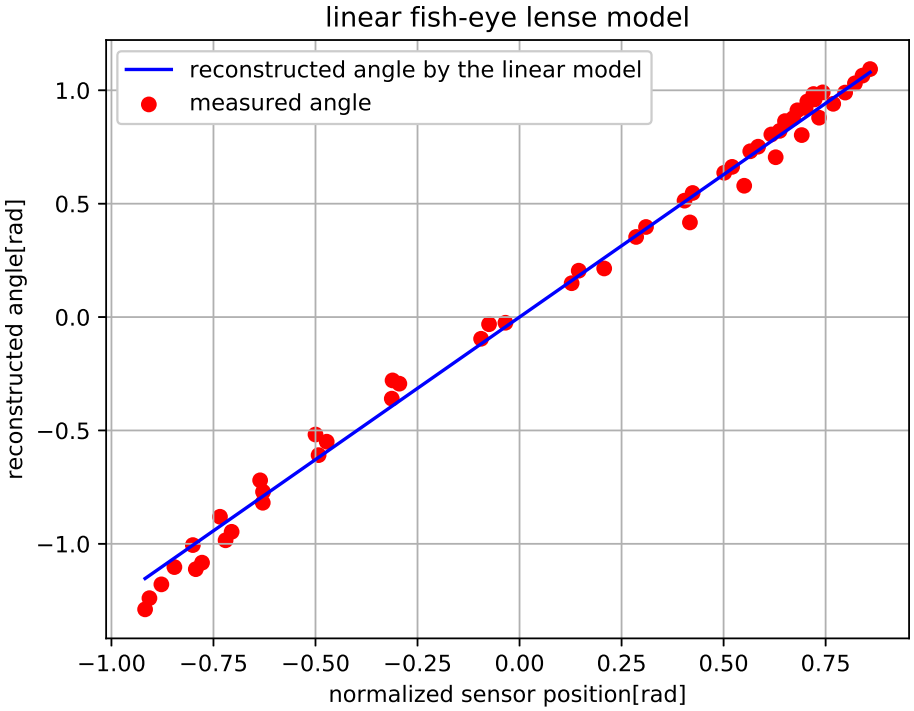


Figure 3.4: The linear model of the lens

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, 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 $(x, y, 1)^T$.

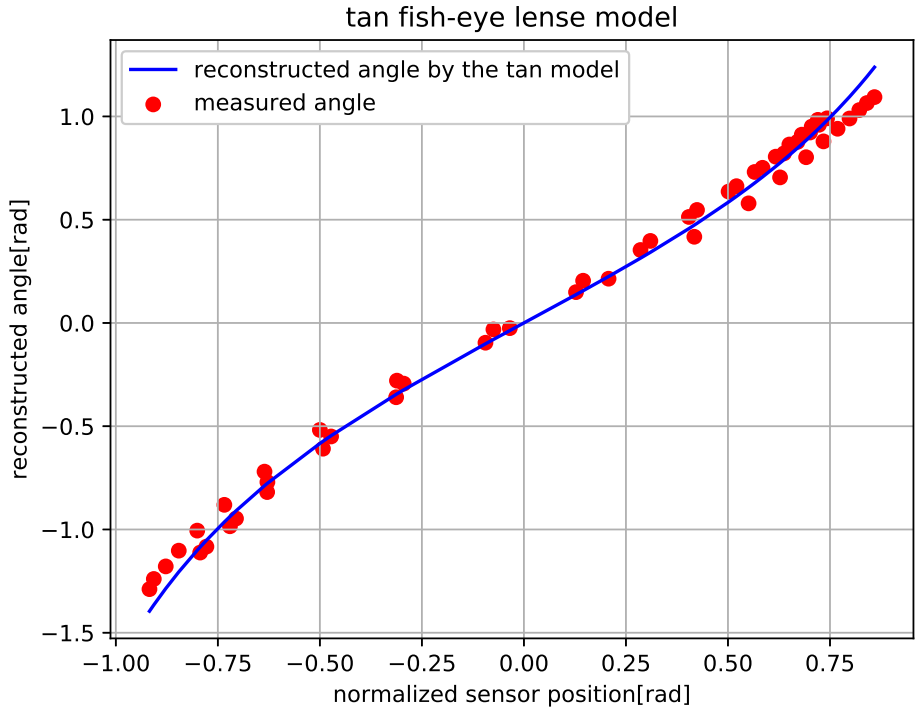


Figure 3.5: The linear model of the lens

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

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

4 Detection and tracking on CPU

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

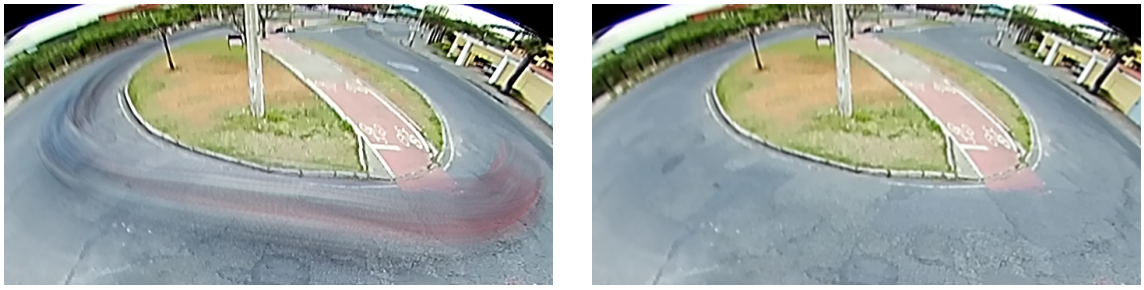
Probably the best classical detection methods from static videos, that can be computed in real time on limited hardware, are based on the background subtraction algorithm [46]. The main idea is creating a model of the scene without the objects that we want to detect and then subtracting the current frame 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.

[24, 65]

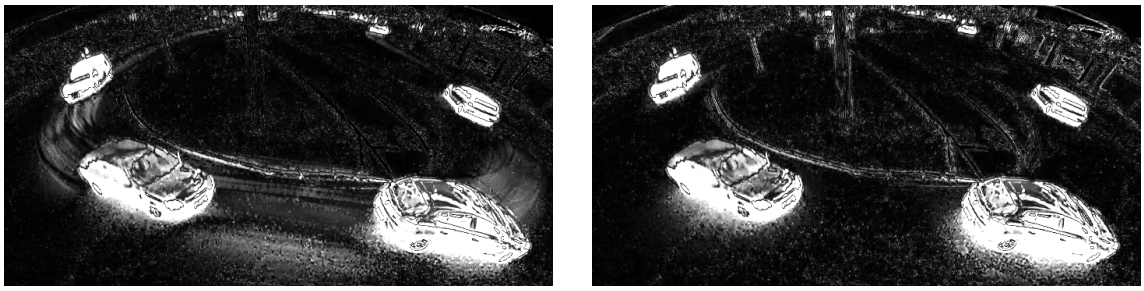
The algorithm has been implemented in opencv [6]. The background is usually created by the mean over several images called running gaussian average [60]. The idea is to estimate a gaussian to each pixel independently. Each pixel is updated with each new frame as a weighted sum. The background looks like 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 [36, 11]. 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.

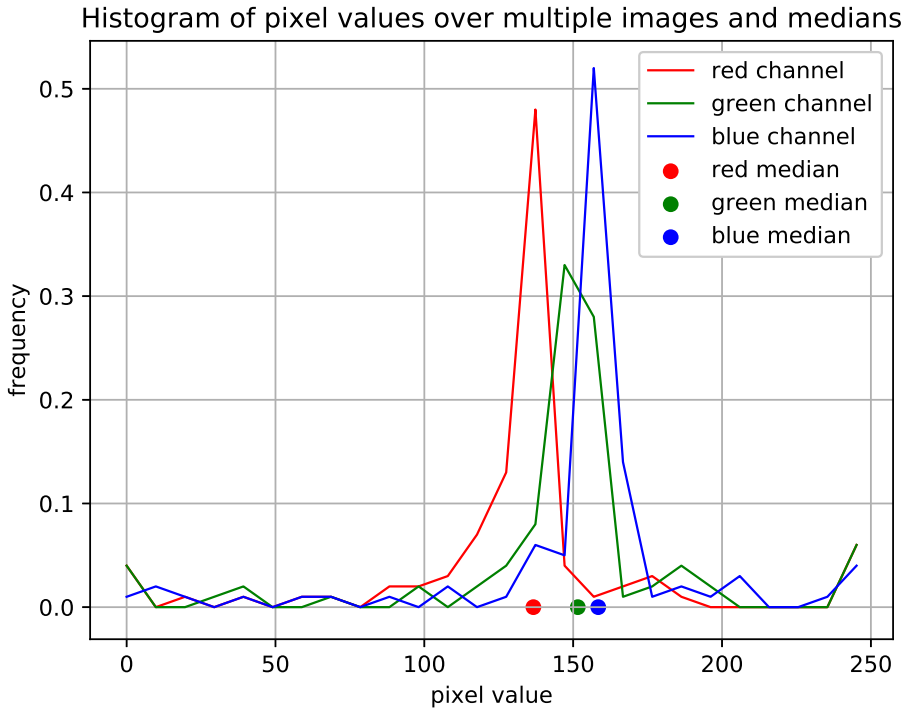


Figure 4.3: The histogram of a particular pixel over 100 images with computed medians

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[6] for extending the background subtraction algorithm to tracking. Optical flow [1] 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 [38].

***desctiption, equations?

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 [51] algorithm, which is an improved version of the Harris corner and edge detector [22]. 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.

********** ...bude smazano. Presunuto do Relaed work/classification Classifiers 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 [4, 63]. 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 [5], adaboost [45], 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[10], Bag of features [30, 44], SIFT features [61, 3] or Haar features [42] can be used. Each Haar feature is a difference of average brightness level in two different rectangles in the image. **********

A simple classifier has been introduced using 87 distinct Haar features. The SVM has been trained on 8144 images of cars and 10567 images of people. The test set split ratio was 0.2. The final accuracy 0.842. That is quite a good score without using neural networks, but not good enough for the final product.

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.

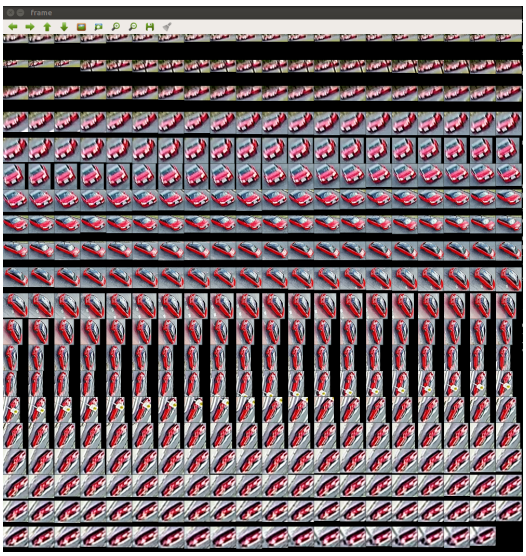


Figure 4.5: The dialog from the annotation tool.

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 Detection and tracking

tried inception rcnn, but not working directly.

***A processing of a single frame needs to be perfected. That includes detection and classification. Detection is in form of a bounding box. Each detection in a single frame needs to be classified into several classes. When we have a detections and classifications over a sequence of frames, a tracking needs to be implemented. That means connecting the bounding boxes into sequences, where each sequence of bounding boxes represents a trajectory of a vehicle. ***

5.0.1 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.

$$P = \frac{TP}{TP + FP} \tag{7}$$

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

$$R = \frac{TP}{TP + FN} \tag{8}$$

The precision-recall curve in the Fig.5.1 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).

$$mAP = \frac{1}{|C|} \sum_{c \in C} AP_c \tag{9}$$

where C is the set of classes.

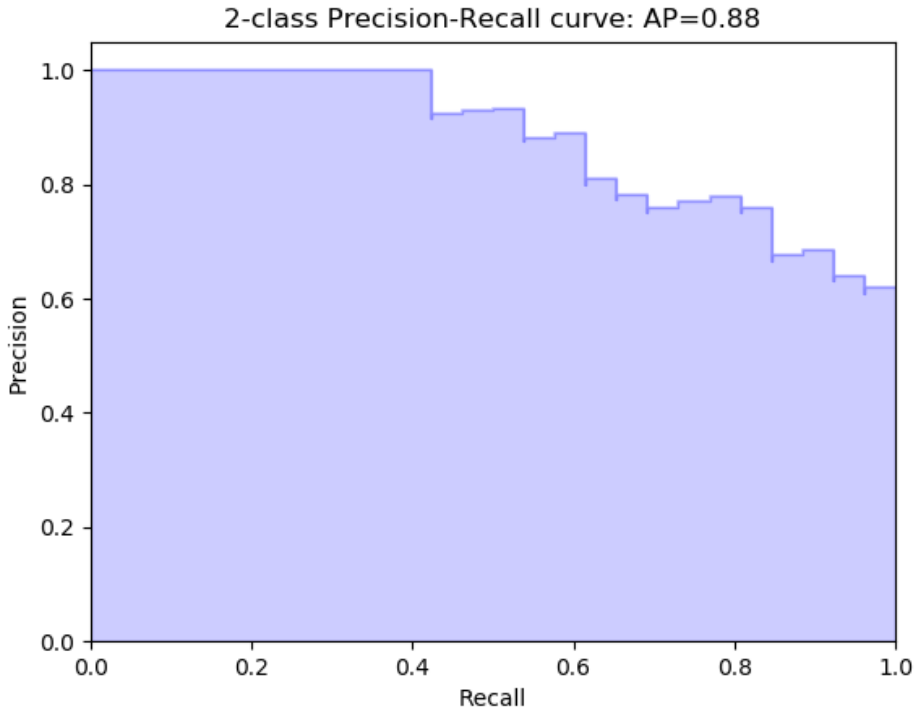


Figure 5.1: Example of precision-recall curve

5.1 Image decomposition

5.2 Dataset

5.3 SSD

6 Reidentification

***When processing a video in one camera has been solved, the problem expands to multiple cameras. Tracking over multiple cameras is deciding, which tracks represent the same vehicle. That is based on a metrics of similarity between two tracks. Finally some representation of the city and the car's position needs to be developed. ***

6.1 dataset

standford [28] cars dataset

6.2 facenet

7 Evaluation

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

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

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

[39] 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.

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

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

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

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

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

[63] 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.

[64] 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.

[65] 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.