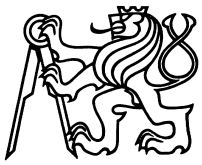
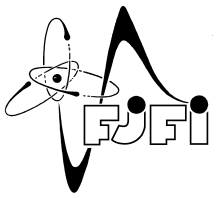


Czech technical university in Prague
Faculty of Nuclear Sciences and Physical Engineering



Automatizovaný systém kontroly motorového prostoru aut

Automated visual inspection system for a car engine space

Master's Thesis

Author: **Bc. Dominik Vít**

Supervisor: **Ing. Adam Novozámský, Ph.D.**

Academic year: 2019/2020

Acknowledgement:
I would like to thank my supervisor Adam Novozámský, Ph.D. for his patient and professional guidance. His knowledge of the topic, approach and advice helped me a lot not only to write this thesis but also to gain numerous skills that I might find very useful in my future work. I would also like to express my gratitude to doc. Milan Krbálek for allowing me to join the project and contribute to its solution. Last but not least, let me thank my family and closest friends for their so needed support.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

Nemám závažný důvod proti použití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 6. January 2020 Dominik Vít

Název práce:

Automatizovaný systém kontroly motorového prostoru aut

Autor: Bc. Dominik Vít

Obor: Aplikované Matematicko-stochastické Metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Adam Novozámský, PhD., Ústav teorie informace a automatizace AV ČR, v.v.i.

Abstrakt: Ve většině automobilových továren je kvalita kontrolována vizuálně lidmi, což může být často nedostatečné a nespolehlivé. Umělá inteligence může tento kontrolní proces v některých aspektech zlepšit. Cílem naší práce bylo v rámci pilotního projektu vytvořit částečné řešení automatizovaného systému kontroly. Konkrétně jsme se zaměřili na hladinu chladicí kapaliny. Naše práce zahrnuje kompletní proces od získání dat z reálného provozu, jejich anotace, předzpracování, nalezení příznaků až po klasifikaci pomocí neuronové sítě.

Klíčová slova: rozpoznání obrazu, Speeded-up Robust Features, konvoluční neuronové sítě

Title: **Automated visual inspection system for a car engine space**

Author: Bc. Dominik Vít

Abstract: In most automobile factories, the quality inspection process is mainly based on vision control, which is often insufficient and unstable. The artificial intelligence can improve some parts of this quality process. The aim of our project was to develop the "Proof-of-concept" of such an automated visual inspection system. We focused on the level of cooling liquid. The analysis includes data collection and labeling, pre-processing, feature extraction and classification.

Key words: Automated inspection, Speeded-Up Robust Features, convolutional neural networks

Contents

Chapter 1

Introduction

For every produced car there is a quality control process at the end of the assembly line. Most of modern car making companies still rely on human vision, which can be limited and sometimes inaccurate. With mathematical development of robust image features and especially with growing popularity and increasing accuracy of neural networks, it is possible to use computer vision for detection of faulty parts.

In this work we try to develop a "Proof-of-concept" of automated visual car inspection to assist employees, as they only have limited time to inspect the whole car. We focus only on detecting the level of cooling liquid and develop this solution for the needs of large Czech car making company, Škoda Auto. We were the first to introduce automated visual inspection solution to the Control block KB8 in this company. At the moment, no other camera hardware is used there for a similar purpose.

In Chapter 2 we describe the project setup and which equipment we used to acquire the necessary amount of video material. Because of a limited budget and operational conditions, we opted for two web cameras to be attached above the conveyor strand to capture real traffic in high definition. In Figure 1.2 it can be seen, that many types of Octavia and Karoq models were passing on the same line - different motorizations and versions of Scout and RS may have different front bumper or engine cover.

In Chapter 3 we give an overview of mathematical methods used for image recognition, from simple local features using second order derivative on the image function to more advanced techniques as SIFT [21] and SURF [3], which can be more robust to image transformations such as change of scale or rotation. We discuss the two methods in depth and show why we have chosen the latter to be used in our work.

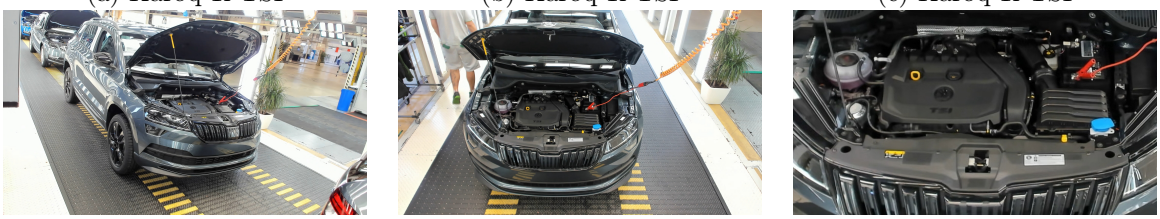
After collecting sufficient amount of data of real factory traffic over several weeks, we then proceeded to processing the data and creating a database of images of cars, which could then be used for training the model. In Chapter 4 we describe how Speeded-Up Robust Features (SURF) was used to detect the exact car position in each video frame and how we decided which parts of the car should be used as its training dataset. Our goal was to separate each car and extract only one image of it in a proper location with respect to the camera. Although the company uses RFID (radio-frequency identification) system for online monitoring of location of every vehicle in the plant (Figure 1.1), we did not have access to it and had to develop an independent solution for estimating car position. We had to overcome several obstacles, such as irregularities in car placement on the line, employees passing between a car and a camera and hence blocking the view and several instances of workers directly hitting our equipment and shifting it from the desired position. HSV color space also played vital role as it allowed us to isolate the distinctive

Figure 1.1: RFID (Radio Frequency Identification) module (yellow box) attached to the roof of every vehicle is used to provide the car maker with exact localization of the car.

color of the cooling liquid. Chapter 5 then contains our solution for determining the level of the cooling liquid.

This by itself would be a sufficient way and a complete solution for the cooling liquid level detection. However, we decided to try and implement machine learning for higher speed and accuracy. Hence in Chapter 6 we first shortly explain basic concept of neural networks and summarize several different designs and then describe how we retrained a convolutional neural network built on U-net architecture to detect and localize the cooling liquid tank. For the training process we use results from previous chapters to feed the segmentation network with labeled images of cars and the location of liquid. With this approach we wanted to gain several thousands of labeled images for training and for validation. We intended to prove, that the combination of SURF image features and convolutional neural network can result in higher accuracy and more importantly, remove the process of having to label each individual training image manually.

(a) Karoq 1l TSI (b) Karoq 1l TSI (c) Karoq 1l TSI



(d) Karoq 1.5l TSI (e) Karoq 1.5l TSI (f) Karoq 1.5l TSI

(g) Karoq TDI (h) Karoq TDI (i) Karoq TDI

(j) Octavia 1l TSI (k) Octavia 1l TSI (l) Octavia 1l TSI

(m) Octavia RS TSI (n) Octavia RS TSI (o) Octavia RS TSI

(p) Octavia Scout TDI (q) Octavia Scout TDI (r) Octavia Scout TDI

Figure 1.2: Karoq and Octavia with different engines and specifications were passing on the same line.

Chapter 2

Data acquisition

Our data was collected on a control block KB8 in Škoda Auto factory in Mladá Boleslav. Before installing necessary devices for data acquisition, we first visited the facility to familiarize with the environment and get further details on the requested output of the project, which was to monitor the level of cooling liquid in a tank inside the engine space. On the control block the cars are parked manually by employees on the conveyor strand, the bonnet is lifted and the whole car is thoroughly visually inspected for defects such as misaligned interior padding, not functioning indicators on a dashboard, discharged battery or other problems that might have occurred during assembly and been missed during previous inspections. One of such problems is the mentioned lack of cooling liquid, which frequently occurs in cars that stayed in the factory longer than expected, e.g. for a change of tyres or for replacement of faulty parts.

During our visit we located possible spots to mount our cameras so that it would be as close to the car as possible while not obstructing any activity of the workers. We decided to mount the first camera to the tv screen rig hanging above the strand. This was the lowest possible placement of a camera above the cars that would not be hit by an opened trunk door. For the placement of the secondary camera we firstly chose the right-side panel, as the cooling liquid tank is always located on the right hand side of the vehicle, but then we noticed that the tank sunk deeper into the engine space on all Karoq models and hence cannot be seen from the right side.

During our second visit we brought our gear and started assembling the setup. Both cameras we used were *Logitech BRIO 4K Stream Edition* webcams. We mounted the *camera A* on the side panel at approximately eye height (Figure 2.1a) and pointed it slightly forwards so that the inside of the engine space could be visible for about 20 seconds. Then we mounted the *camera B* on the TV screen above the conveyor strand. The camera pointed downwards and slightly forward and the engine space of a car could be visible for about 30 seconds. After mounting both cameras with a duct tape we attached two 5 m USB cables to the railings above the conveyor and led them to a side panel on which we have chosen to attach the computer for storing data. The computer was a low profile case Dell machine with a 1TB hard drive. We laid it and attached it with zip ties on the railing at about 2.50 m high so that it would not be directly visible to a naked eye (Figure 2.2).

This setup was by no means robust but since it was meant only as a pilot project, we decided not to invest more resources. The first problem was the position of *camera A*, which was too far for detecting the cooling liquid tank. Because we decided to only narrow our focus to Octavia model we could ignore the Karoqs and thus mount the camera to right hand side, where it was significantly closer to the liquid tank. The second problem was that a worker hit the camera

(a) (b)

Figure 2.1: Mounting of a *camera A* and *camera B* to the railings above the conveyor.

while standing by and then returned it to a shifted position. This actually happened several times and we had to manually modify the script in Chapter 4 to account for the changes.

Both cameras were set to only FullHD resolution, which would be sufficient for purposes of the project but also would keep the size of files reasonably low. We reduced the file size also by setting the compression standard to HEVC h265, which is about quarter the size of a regular h264 video. This was important because we needed to record several days of continuous traffic. The second reason not to record in 4K was the use of passive USB 3.0 cable extensions. With the length needed to connect both cameras to the computer it would not be possible to transfer videos in 4K resolution.

The script for recording the video files was written in C++ and ran ceaselessly on the computer. We wrote it as an endless cycle, that launched in two parallel threads *ffmpeg.exe* on both cameras simultaneously. The frame rate was set at only 1 fps and the total duration was set to 10 minutes per video. During this time all frames were recorded in motion JPEG (M-JPEG) and after that encoded in h265. We also wanted to avoid recording unnecessary frames, when the conveyor was stopped because of a break or a shift change. Fortunately, all working breaks are scheduled regularly as are shift changes and happen at exactly the same time every day. This allowed us to modify the script to stop recording at those time periods. There were a few instances when the line was unexpectedly stopped and one car remained under our cameras even for half an hour, while people were also passing in front of it. We had to account for these unusual cases and build our systems robust enough.

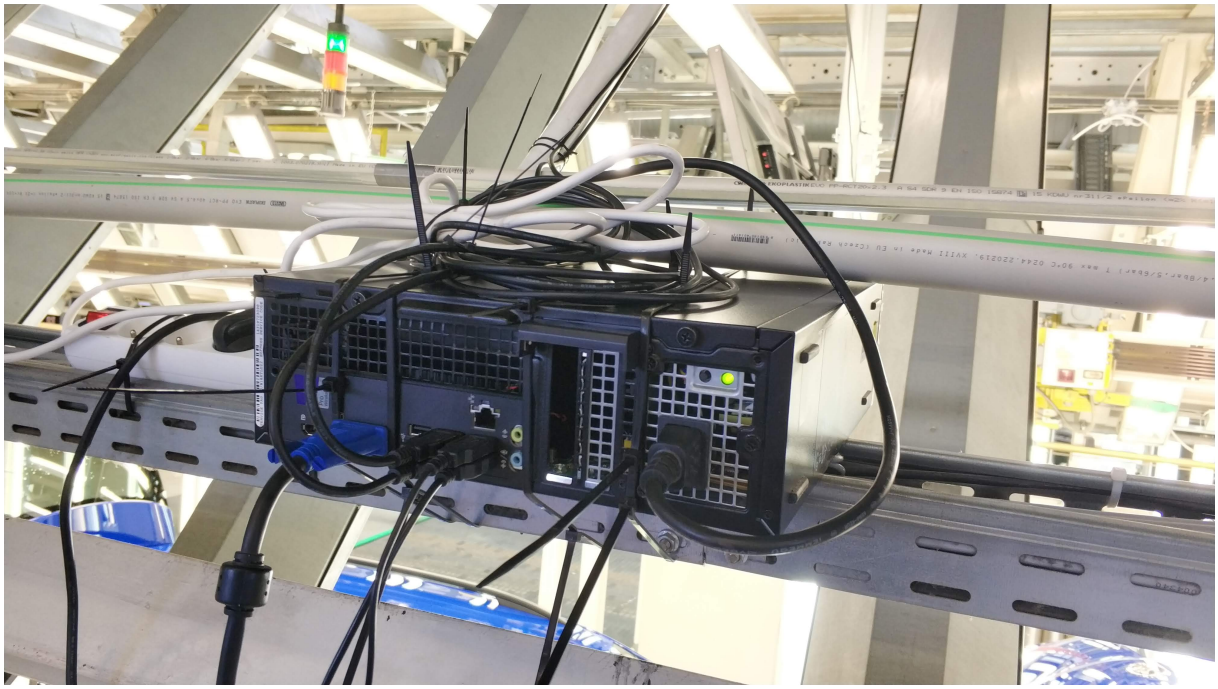


Figure 2.2: Computer was also attached to the construction above the conveyor.

Chapter 3

Feature space

In the previous chapter we described our process of collecting image data on the control point in Skoda Auto factory. To understand the methods of processing the data in Chapter 4 an overview of image recognition techniques is given in this chapter. From a brief historical introduction we move to a deeper description of Scale Invariant Feature Transform (SIFT) in Section 3.2 which marked an important milestone in approach to feature detection and description.

SIFTs are important in understanding the Speeded-Up Robust Features (SURF) introduced several years later that improved the original algorithm by using methods described in Section 3.3. We used SURF for its robustness and speed and because they are well implemented in MATLAB. After explaining the SURF more in depth we give a short overview of methods following and improving the before mentioned.

3.1 Computer vision

Attempts in computer vision date back to late 1960s when it was considered to be a simple problem that would be solved in a few months [35]. However, only a slow progress was made and most focus was on reconstructing a full 3D structure of the image in order to fully understand the scene. In the 1980s, image pyramids gained in popularity and *scale-space* was developed, while there was an increased activity in using projective invariants in the next decade. In the 2000s, interest point features started to dominate in most research before the boom of neural networks in recent years. In the following text we dive deeper in the theory of interest based features as we exploited them in our work. We stem from the work of Tinne Tuytelaars and Krystian Mikolajczyk as they gave an overview of invariant interest point detectors, how they evolved over time, how they work and what their respective strengths and weaknesses are [36].

3.1.1 Local features

While simple global features, like color histograms, work surprisingly well, there are several important instances where they fail. Namely it is their low ability to find correspondences in image after changes in viewing conditions or partial occlusion. Image segmentation was supposed to address this issue by segmenting the image in a limited number of regions, with each such region corresponding to a single object or part thereof, such as a color or texture. However, this creates a new problem - how to select such regions. Local features tend to solve these issues and that has brought them massive popularity up until recently.

It was first noted by F. Attneave in 1954 [1] that information on shape is concentrated at dominant points having high curvature.

A good local feature should meet the following conditions - *invariance*, *discriminability*, *robustness*, *completeness* and *independency*. The invariant feature, or simply the invariant, is a functional I that maps the image space such that $I(f)$ depends on the class f belongs to but does not depend on particular appearance of f [9]. This means that $I(f) = I(D(f))$ for any f and any instance of D. Discriminability on the other hand means that features from different classes should be significantly different (far apart). These two conditions go against each other, as the most invariant feature would have zero discriminability. For instance digits 6 and 9 would have the same shape features. Although this would be invariant to rotation it fails to differentiate between them. After adding a feature of center of mass, the feature would become complete, which means it could fully recover the original image.

To measure which of all newly proposed detectors is the best (in terms of quality of the feature for tasks like image matching, object recognition and 3D reconstruction), *repeatability* score has been proposed by Schmid, Mohr and Bauckhage (2000). They state, that repeatability explicitly compares the geometrical stability of the detected interest points between different images of a given scene taken under varying viewing conditions [31]. While previous methods evaluated detectors for individual images only, they rather observe if an interest point is "repeated", that is if the scene point detected in the first image is also accurately detected in the second one. The final repeatability rate is the percentage of total observed points that are detected in both images. The changes between their observed pictures were after applying on planar images transformation like rotation, scale changes, illumination changes, viewpoint changes and adding noise. This repeatability was used to measure, how much SURF detectors outperform SIFTs. We will discuss it more in the following sections.

3.1.2 Corner vs blob detectors

Two most intuitive local features are corners and blob-like structures. Both methods have their strengths but in its basis they are complementary to each other. The former builds on edge detection where an edge is detected in the point of highest gradient of the image function. As all images are comprised of pixels, the derivative is approximated in the discrete domain. Since most images contain noise, which by its nature also has high gradients, the image first has to be smoothed with a function, most commonly a Gaussian. With use of the convolution theorem

$$\frac{\partial}{\partial x}(f * g) = (\frac{\partial f}{\partial x}) * g \tag{3.1}$$

it is only needed to compute first the derivative of Gaussian and then convolve it with the image which is more simple and faster than finding derivatives on the smoothed image. This plays a vital role in all derivation based detectors and we will expand on it in following sections. The most famous first-order edge detector is a Canny detector [7] that uses approximations by derivatives of Gaussian in conjunction with non-maximum suppression and thresholding with hysteresis.

Corner detectors build on the previous and on assumption that a corner is a point with two strong and different edges in its neighborhood. This means that the detector looks for regions that have more than one dominant gradient directions. The corners detected in a 2D image are points with high curvature and do not necessarily have to correspond to projections of 3D corners [36]. Probably the most popular corner detector is the Harris detector, proposed by

Harris and Stephens [11]. It uses second order matrix (auto-correlation matrix), which describes the gradient distribution in a local neighborhood of a point:

$$M = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(x,\sigma_D) & I_x(x,\sigma_D)I_y(x,\sigma_D) \\ I_x(x,\sigma_D)I_y(x,\sigma_D) & I_y^2(x,\sigma_D) \end{bmatrix} \tag{3.2}$$

with

$$I_x(x,\sigma_D) = \frac{\partial}{\partial x} g(\sigma_D) * I(x), \tag{3.3}$$

$$g(\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.4}$$

The σ_D parameter is the scale of Gaussian kernel for the image derivatives, while σ_I is the scale of Gaussian window used for smoothing the image.

The eigenvalues of this matrix represent the principal signal changes in two orthogonal directions in a neighborhood around the point defined by σ_I. Hence, a corner can be found at locations for which both eigenvalues are large. To reduce the high computational time of computing eigenvalues, Harris and Stephens proposed a *cornerness* measure M_c with a tunable parameter λ:

$$M_c = \det(M) - \lambda \operatorname{Trace}(M). \tag{3.5}$$

After extracting local maxima of this function, using non-maximum suppression, the interest points are translation and rotation invariant.

Although the original Harris detector is widely used, it is not scale invariant. Detection of blob-like structures was exploited for automatic scale selection. Introduced by Lindeberg [20], interest points are detected at their own characteristic scale. Improving this method, Mikolajczyk and Schmid [23] created robust and scale-invariant feature detectors called Harris-Laplace and Hessian-Laplace. They used the determinant of the Hessian matrix to select location and Laplacian to select scale. Together with approximating the Laplacian of Gaussian (LoG) by a Difference of Gaussian (DoG) filter (Lowe [21]), Hessian-based detectors are more stable, repeatable and faster.

3.2 Scale Invariant Feature Transform (SIFT)

A. Lowe introduced in 1999 his set of scale invariant feature detectors and descriptors - SIFT. This was an essential step in this field of research, so although for our work we were using SURFs, which were introduced in 2008, we give a thorough explanation of this method as the SURF build on and expand the research by Lowe. SIFT uses the following structure to generate a feature set [21]:

- **Scale-space extrema detection**
- **Keypoint localization**
- **Orientation assignment**
- **Keypoint descriptor**

Scale space was introduced in 1983 by A. P. Witkin [38]. He drew from previous observations that local features are detected with derivatives taken over certain neighborhood. However, the

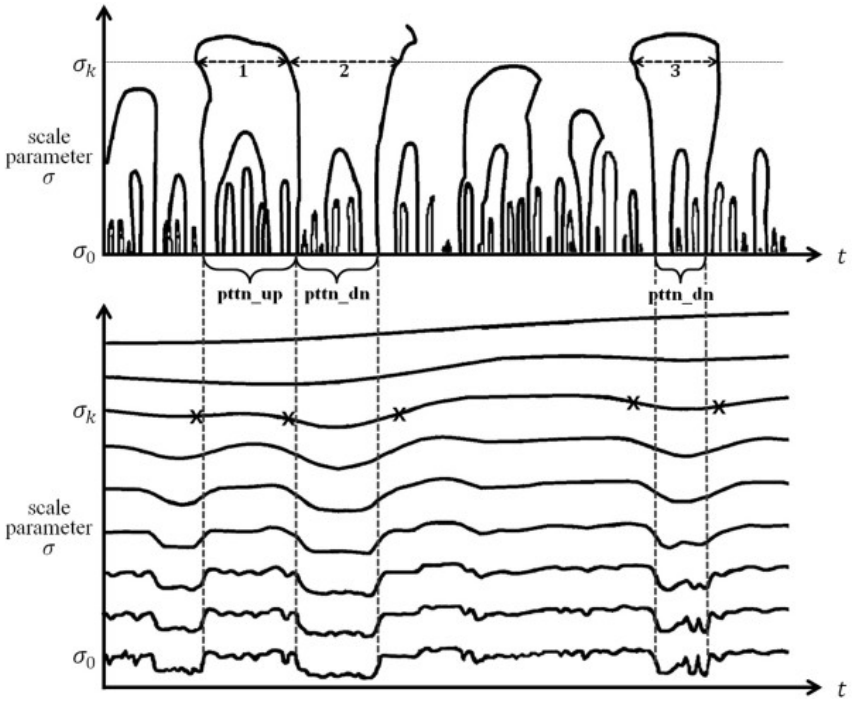


Figure 3.1: A typical 1 dimensional scale-space image from [17] with increasing σ (on the bottom) and the contours of zero-crossing points of second derivatives of the image (on the top). The vertical dotted lines mark inflection points for the lowest scale σ_0. The horizontal dotted lines and their corresponding crosses show distances of inflexion points after smoothing with kernel size of σ_k.

specific size of the neighborhood is unknown. Lindeberg observed, that Gaussians are optimal for scale-space analysis [19]. To create the scale-space, Witkin first applied whole spectrum of scales on the image. The scales are σ values of a Gaussian. After taking the second derivative of a smoothed image, inflection points can be found at zero-crossings.

A simple 1D signal example with zero-crossings plotted against the scale can be seen at Figure 3.1. The zero-crossing curves of two neighboring inflection points always connect at certain scale σ. Smaller changes close quickly while more significant jumps (in image processing corresponding to more dominant edges) close as arches at much higher scales (σ_k in the figure). Witkin then introduced the *interval tree* which reduces the scale space image to a simple tree, concisely but completely describing the qualitative structure of the signal over all scales of observation. This representation shows how much stable are zero-crossings over different scales. A top level description can be achieved by iteratively removing nodes from the tree, splicing out nodes that are less stable than any of their parents and off springs.

For 2D images, finding local maxima in scale-space of LoG gives interest points. The scale-space of an image is defined [22] as a function $L(x, y, \sigma)$, that is produced from the convolution over x and y of a variable-scale Gaussian $G(x, y, \sigma)$ (Equation 3.4), with an input image $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.6}$$

To find keypoint locations in scale-space, Lowe used [21] approximating Laplacian of Gaussian (LoG) with Difference of Gaussian (DoG) which can be computed from the difference of two

nearby scales separated by a constant multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, \sigma) - L(x, y, \sigma). \end{aligned} \tag{3.7}$$

The approximation of LoG using DoG is based on the heat diffusion equation:

$$\frac{\partial G}{\partial \sigma} = \sigma \Delta G. \tag{3.8}$$

From the finite difference approximation using the difference of nearby scales at $k\sigma$ and σ we get:

$$\sigma \Delta G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{3.9}$$

and therefore,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \Delta G. \tag{3.10}$$

This means that the Laplacian of Gaussian can be computed as a subtraction of applied Gaussian filters on the image, with parameter $k\sigma$ and σ respectively. The factor $(k - 1)$ in Equation 3.10 is a constant over all scales and does not influence extrema location. Parameter σ^2 on the right hand side of the Equation 3.10now is the scale normalization required for the scale invariant Laplacian. An efficient way to compute DoG can be seen in Figure 3.2. At first, a Gaussian is applied to the image at different scales, which forms an octave. Each two adjacent images in an octave are subtracted, creating a Laplacian. After completing a new octave, these images are subsampled by taking only every second pixel. By repeating this process, a *scale-space pyramid* is created.

To find the local maxima or minima of $D(x, y, \sigma)$, each sample point is compared to its 8 nearest neighbors as well as to the 9 points in the scale above and the scale below (see Figure 3.3). If it is larger or smaller than all 26 neighboring points, then it is a potential interest point. Then a closest point is found in the next level of the pyramid and compared to its neighbors in the same manner.

After choosing candidates from the extrema detection, outliers are rejected. Those are selected based on the information of location, scale and ratio of principal curvatures. This allows points to be rejected if they have low contrast or are poorly localized along an edge. An improved method for keypoint localization has been developed by Brown and Lowe in 2002 [5] for better matching and stability. Instead of simply locating keypoints at location and scale of the central sample point, they suggested fitting a 3D quadratic function to the local sample point. It uses the Taylor expansion series of the scale-space function:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x. \tag{3.11}$$

The sub-pixel interest point location is taken as the extremum of this function

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}. \tag{3.12}$$

Evaluating the function $D(\hat{x})$ at the extremum is useful for rejecting unstable extrema with low contrast. Usually, all extrema points with $|D(\hat{x})| \leq 0.03$ are discarded.

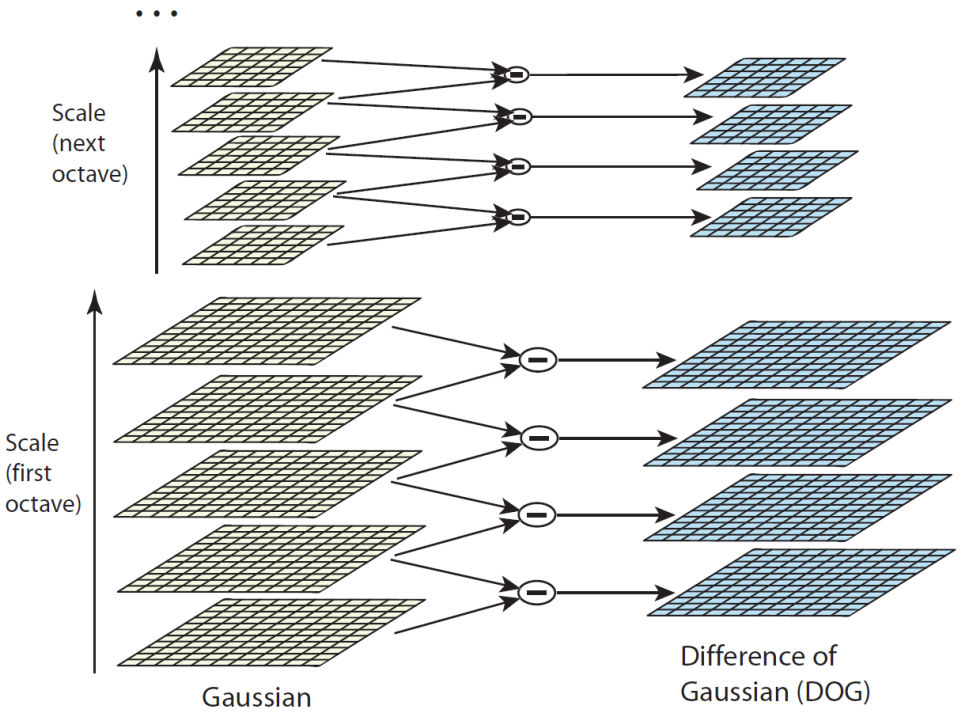


Figure 3.2: [22] For each octave of scale space, the image is repeatedly convolved with Gaussian. This produces a set of scale space images (on the left). To produce the Difference of Gaussian, two neighboring Gaussian images are subtracted. Then each image is downsampled and the process is repeated.

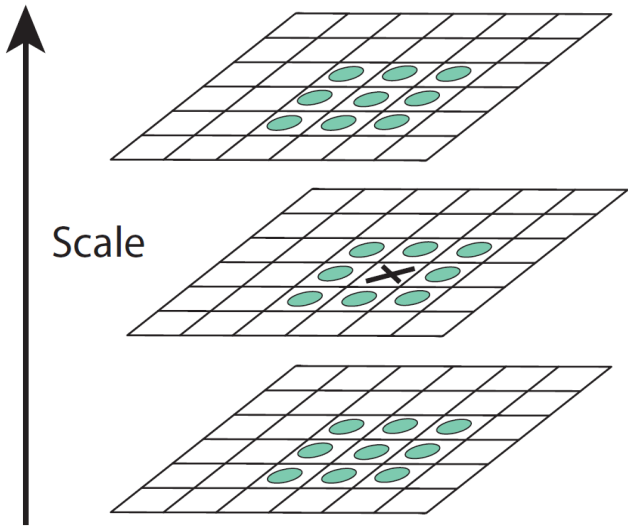


Figure 3.3: Maxima and minima of the Difference of Gaussian images are detected by comparing a pixel (marked with x) to its 26 neighbors in 3x3 regions at the current and adjacent scales [22].

Further keypoints rejection is necessary for detector stability. The Difference of Gaussian has a strong response along edges. If we assume DoG as a surface, then a poorly defined peak would have a large principal curvature (PC) along one edge while a small one in the perpendicular region. Principal curvature can be computed from Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}, \tag{3.13}$$

where the differences are estimated by taking differences of neighboring sample points.

By using what Harris and Stephens proposed [11] (as we discussed in the previous section), the exact value of eigenvalues does not have to be computed if we are interested only in their ratio. Let α be the eigenvalue with the largest magnitude and β with the smallest. Then the sum of eigenvalues corresponds to the trace of $\mathbf{H}$ and their product to the determinant:

$$\text{Trace}\,(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \tag{3.14}$$

$$\det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \tag{3.15}$$

Now the outliers can be simply removed by comparing the ratio of the trace and determinant to a fraction:

$$\frac{\text{Trace}\,(\mathbf{H})^2}{\det(\mathbf{H})} < \frac{(r+1)^2}{r}, \tag{3.16}$$

where $r = \frac{\alpha}{\beta}$. The fraction on the right hand side of the equation is at a minimum when the two eigenvalues are equal and it increases with r. From experiment by Lowe [22], keypoints are eliminated if they have a ratio between the principal curvature greater than 10 ($r > 10$).

The next step in SIFT description is orientation assignment. Each key location is assigned a canonical orientation so that the image descriptors are invariant to rotation [21]. First, the scale of each point is used to select a corresponding Gaussian to create a smoothed image $L(x, y)$ on which a gradient magnitude and direction are computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{3.17}$$

$$\Theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))). \tag{3.18}$$

Then within a region around each interest point, an orientation histogram of 36 bins is formed to cover the 360 degree range of rotations. Each histogram sample is also weighted by both its gradient magnitude and a Gaussian-wighted window of 1.5 times the size of scale of the point. This, together with previous operations assigns to each keypoint an image location, scale and orientation. The objective, however, is to create a local image descriptor, that would be highly distinctive yet as invariant as possible.

The resulting SIFT descriptor, as proposed by Lowe [22], can be now simply described with Figure 3.4. For each detected keypoint a gradient magnitude and orientation is computed in a 16x16 neighborhood and weighted with a Gaussian at corresponding scale (left-hand side of the figure). The circular window illustrates weighting of gradients with a Gaussian at 1.5 times the scale, which of course stresses more the gradients around the keypoint while giving less emphasis on those further away. The area is then divided into 4x4 blocks and for each block an 8 bin histogram of gradient orientation is calculated (right-hand side). Hence, the SIFT descriptor is a 128 dimensional vector. This vector is also normalized to reduce effects of illumination changes.

Final step in image recognition is matching a new image to the database of training images. SIFT finds the nearest neighbor using Euclidean distance. To prevent false positives and to make the matching more efficient, the ratio of distances of the best and 2nd best match is observed. For larger values, close to 1, a possible match is found but smaller values are discarded as ambiguous.

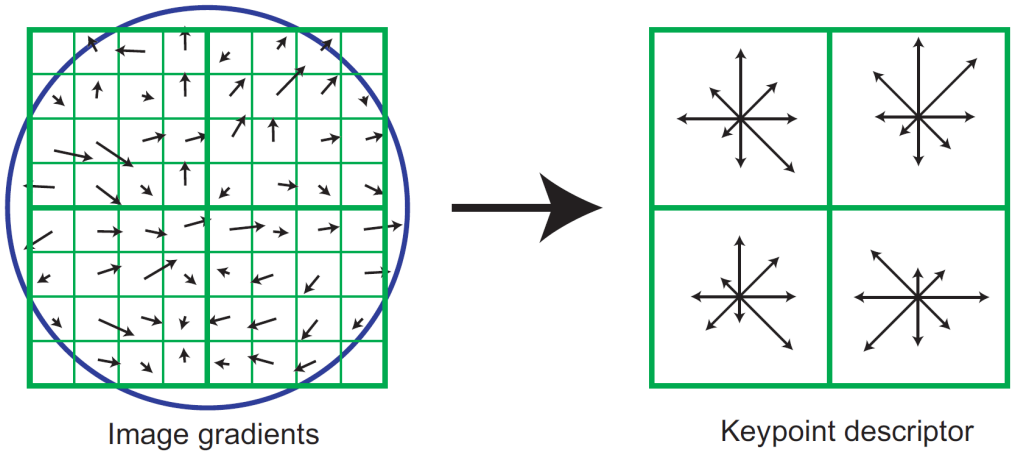


Figure 3.4: The SIFT descriptor analyses a neighborhood of a keypoint by computing relative orientation and magnitude of gradients. The area is then divided into 4x4 regions, with each region being assigned an 8 bin histogram. These histograms concatenated form a 128D descriptor. In this figure, only 8x8 area is combined to a 2x2 descriptor, whereas in reality a 16x16 area is used for 4x4 descriptors [3].

3.3 Speeded-Up Robust Features (SURF)

Partially building on SIFT, H. Bay, A. Ess, T. Tuytelaars and L. Van Gool first presented in 2006 their scale and rotation invariant feature detector and descriptor called Speeded-Up Robust Features (SURF) [3]. It also uses Hessian matrix-based measure for detectors, but by implementing several different techniques, like integral images for convolution and Haar wavelet response for description, it can be computed several times faster and is more robust to certain types of transformation.

For some of these reasons we opted for SURF above other image feature detectors and descriptors. That is why we discuss them more in depth in this section and explain, how they draw on SIFT and where it differs. Most of the information we collected from the original paper [3].

3.3.1 Integral images

Integral images play a vital role in improving the speed of SURF against its predecessors. Integral image, formerly known as *summed area table* (F. C. Crow 1984 [8]), uses an intermediate representation of the original image. Each new pixel is computed as a sum of pixel intensities from the origin. More specifically, as noted by P. Viola and M. Jones (2000 [37]),

$$ii(x,y) = \sum_{x' \le x} \sum_{y' \le y} i(x',y'). \tag{3.19}$$

where $ii(x,y)$ is the integral image and $i(x,y)$ is the original image. The integral image can be quickly computed in one pass using the following recurrences

$$s(x,y) = \quad s(x,y-1) + i(x,y) \tag{3.20}$$
$$ii(x,y) = \quad ii(x-1,y) + s(x,y), \tag{3.21}$$

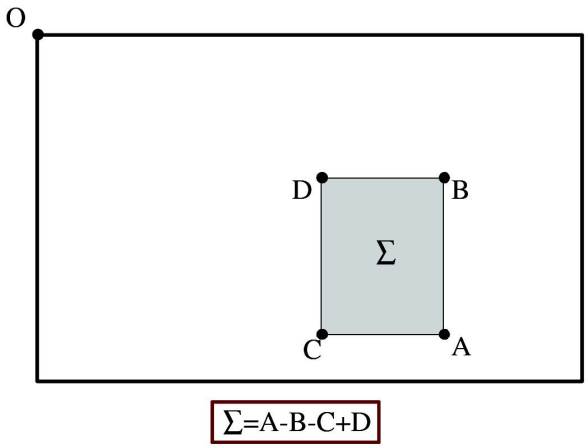


Figure 3.5: The advantage of using integral images is that it takes only 3 additions to calculate the sum of intensities inside any area [2].

where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$ and $ii(-1, y) = 0$. Once the integral image is computed, it takes only 3 additions (as shown in Figure 3.5) to compute a sum of intensities of any rectangular area. Also the computational time of normal convolutions grows with filter size. This is not the case for integral images where the time remains constant, which is very helpful for SURF as it uses large filter sizes.

3.3.2 Hessian matrix-based interest points

Similar to SIFT, SURF also detects blob-like structures by maximizing the determinant of a Hessian matrix (see Equation 3.13). Using a discretized and cropped Laplacian of Gaussian for smoothing has a negative impact on repeatability under image rotations around odd multiples of $\frac{\pi}{2}$ due to the square shape of the filter. But because the detector still performs reasonably well, SURF goes a step further with approximating the second order derivatives of a Gaussian with box filters (Figure 3.6). By using integral images, the computational time of these is very quick and is independent on the filter size.

The filter sizes of 9x9 in Figure 3.6 approximate the Gaussian with $\sigma = 1.2$. If we denote them D_{xx}, D_{xy} and D_{yy} we can find the blob response in image at location x by calculating the determinant similarly to SIFT:

$$\det(\mathbf{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{3.22}$$

where the weight w is needed for conservation of energy between Gaussian kernels and their approximations. For this scale the weight is [3]

$$w = \frac{|L_{xy}(1.2)|_F|D_{yy}(9)|_F}{|L_{yy}(1.2)|_F|D_{xy}(9)|_F} \approxeq 0.9. \tag{3.23}$$

3.3.3 Scale-space representation

Unlike Lowe [21], who implemented scale-space as an image pyramid of repeatedly smoothed images, then subtracting two adjacent layers and finally downsampling the new image, the SURF can apply box filters directly at any scale. As shown in Figure 3.7, the use of integral images

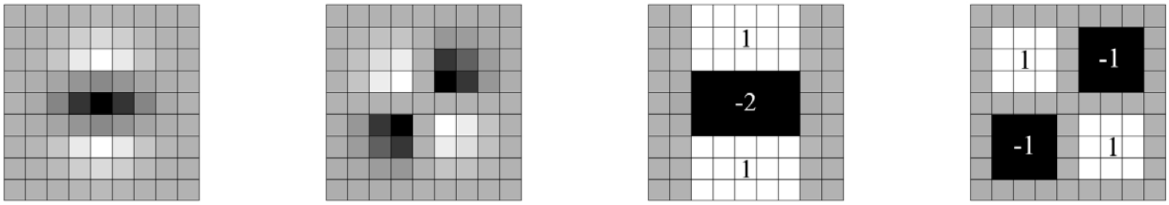


Figure 3.6: [3] The discretized and cropped Gaussian second order derivative in y- and xy-direction used in SIFT (two images on the left). Box filters further approximating those derivatives (two images on the right).

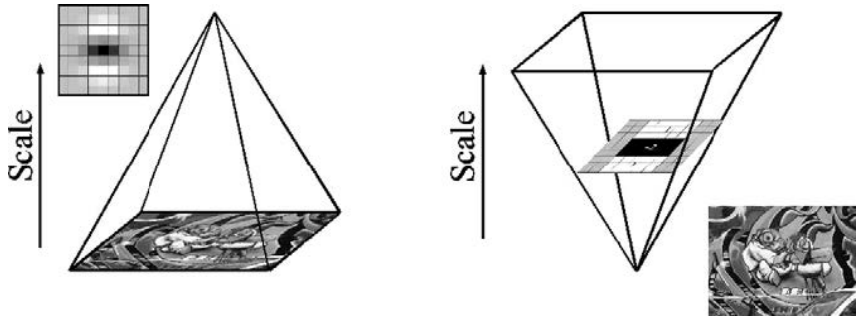


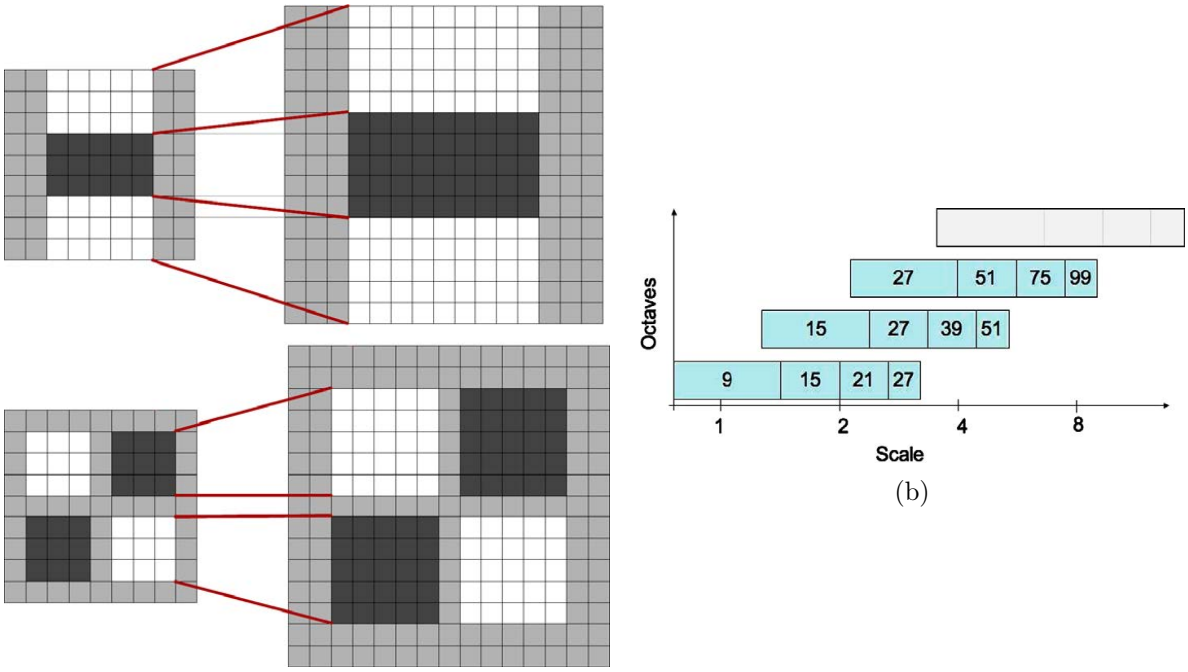
Figure 3.7: A pyramid of iteratively smoothing and downampling image used in SIFT (on the left) in contrast to using integral images for scaling the filter size at constant cost [3].

allows for constant speed of upscaling the box filter. The basic filter size of 9x9 is considered an initial scale $s = 1.2$ and the following scales are obtained by gradually using bigger masks.

In order to cover the full scales, scale-space is divided into several octaves. Each octave represents a series of filter response maps obtained by convolving the original image with a filter of increasing size [2]. The first filter is the above mentioned 9x9 filter. If l_0 is the length of each negative or positive lobe of the filter, then for the first scale level is $l_0 = 3$. To obtain a new level, each lobe has to be expanded by a minimum of 2 pixels (Figure 3.8a).This leads to a total increase of the filter size by 6 pixels for each new level in the first filter, thus also creating filters 15x15, 21x21 and 27x27. Because of the 3D non-maximum suppression explained in Section 3.2, the first and the last Hessian response maps cannot contain such maxima (they are only used for comparison). This is why the filter size is more than just doubled.

For each new octave the filter size difference is doubled (12 pixels are added in the second octave and 24 in the third) and it starts on the second smallest value of the previous octave. In particular, the filter sizes for the second octave are 15x15, 27x27, 38x38 and 51x51 and for the third 27x27, 51x51, 75x75 and 99x99 as can be seen in Figure 3.8b. A possible fourth and fifth octave can also be computed in the same way, but it has been shown by Bay et al. that number of interest points per octave decays quickly [3].

The Frobenius norm in Equation 3.23 remains constant for all scales ensuring that no further normalization is needed for the scales, neither is further weighting of the filter response.



(a) (b)

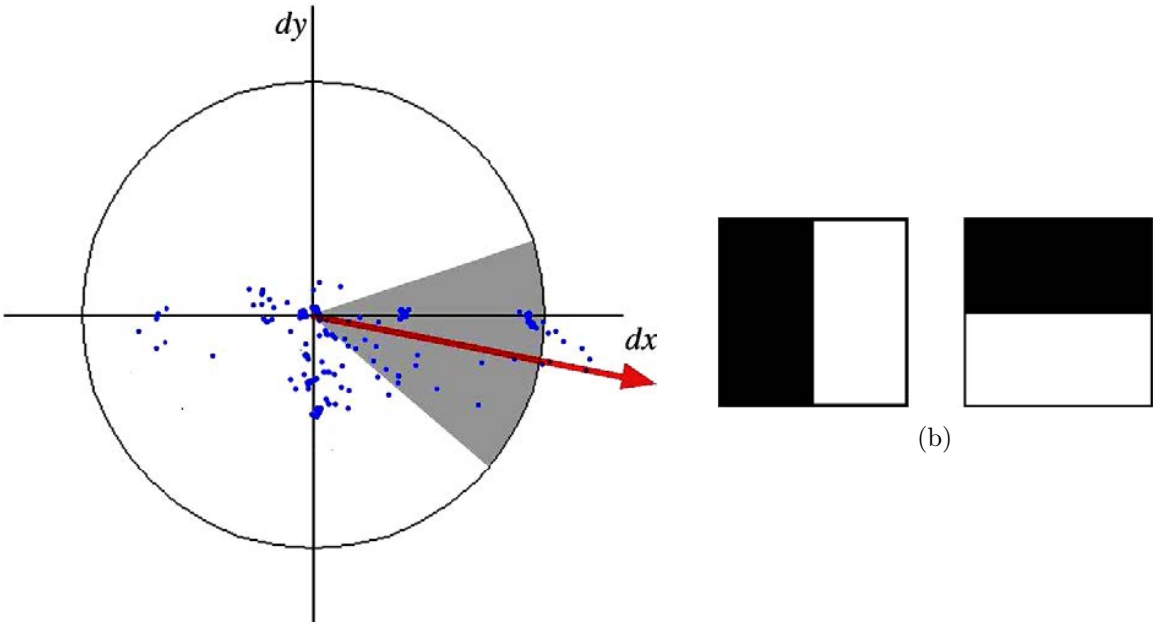
Figure 3.8: (a) Scaling of D_{yy} and D_{xy} filters from 9x9 to 15x15. (b) Representation of filter side lengths for 3 octaves. The logarithmic horizontal axis represents scales [3].

3.3.4 Interest point description and matching

The SIFT descriptor used the gradient information of a 16x16 area to create a 128D vector per each interest point. SURF descriptor, on the other hand, builds on the distribution of first order Haar wavelet responses in x and y direction and only uses 64D. This again in combination with integral images and introduction of the sign of Laplacian is not only several times faster than SIFT, but also has increased robustness. Thus the name Speeded-Up Robust Features [3].

The first step in description is orientation assignment to achieve orientation invariance. Circular area of radius $6s$ around each interest point is selected (s being the corresponding scale in which the interest point was detected). A sum of Haar wavelet responses is calculated inside each of these subregions. The Haar wavelet is shown in Figure 3.9b. The responses, weighted with a Gaussian of scale $2s$, are plotted in Figure 3.9a. The dominant orientation is estimated by computing sum of responses within a sliding orientation window of size $\frac{\pi}{3}$. This exact value was selected experimentally in [3] and is one of the reasons why SURF is patented. Then sums of responses in x and y direction are computed with the longest one determining the interest point orientation.

To extract the features, a square area with length $20s$ is laid over the interest point and divided into 4x4 squared subregions and oriented along the selected orientation from previous step. For each of the subregions, Haar wavelet responses are computed at 5x5 evenly spaced sample points. These responses are then summed in x and y direction (denoted as d_x and d_y) and also take into account their polarity, the sum of absolute values is computed in both directions. The resulting 4 dimensional feature vector for each subregion is $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Concatenating this vector for all 4x4 regions, SURF produces a 64D feature vector. This is half the size of SIFT feature vector, hence the faster speed and higher robustness. In addition, wavelet responses



(a) (b)

Figure 3.9: [3] The dominant orientation of SURF is selected by sliding a window of $\frac{\pi}{2}$ size and summing Gaussian weighted wavelet responses at each sample point in a circular neighborhood of the interest point (Figure a). Haar wavelet filters to compute responses in x and y direction. Dark area has a value of -1 and the light +1 (Figure b).

are invariant to a bias in illumination and also after normalizing the vector v to a unit vector, invariance to contrast is achieved. Also SURF features tend to be more robust to noise as can be seen in Figure 3.11a, where the gradients orientations as described by SIFT are affected while wavelet responses remain the same.

After finding feature vectors describing each interest point of the image, they have to be matched to an image from the training set. Here comes another advantage of SURF over SIFT and that is the inclusion of the sign of Laplacian, which is already computed from the previous steps. This helps to distinguish bright blobs on a dark background from dark blobs on light background (see Figure 3.11b). This simple information allows for faster matching, without any negative impact of the descriptor performance. Otherwise it uses the same technique for matching as SIFT does.

3.4 Overview of other methods

In this section we briefly sum up several other image recognition techniques that were developed after the introduction of SIFT and SURF, mostly in early 2010s. The boom in their development has recently been cooled down by the increased popularity of neural networks, which tend to be faster and are having much higher precision and variability of use. The methods presented here, were all correctly mathematically explained. Each of them has its benefits but we have chosen to be only using SURF for their overall performance and because they were already implemented in MATLAB.

Maximally Stable Extremal Regions (MSER), introduced in 2002 by Matas et al. [25]

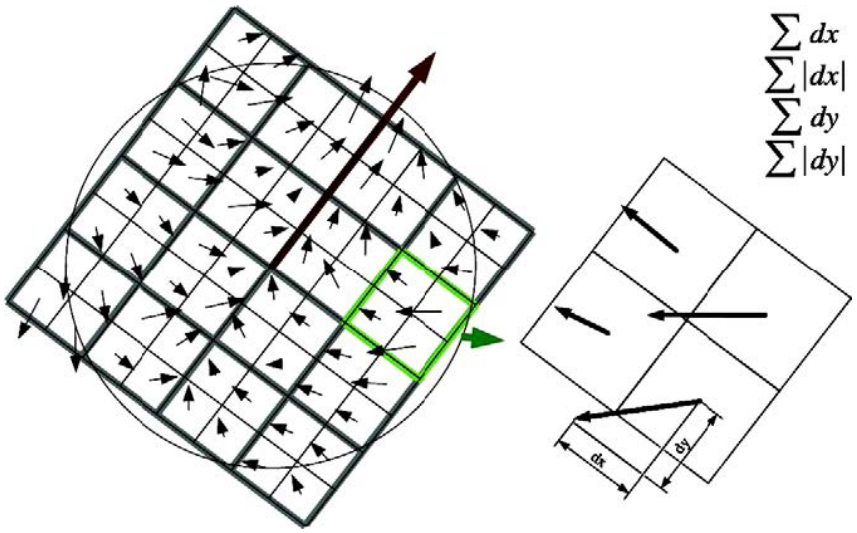
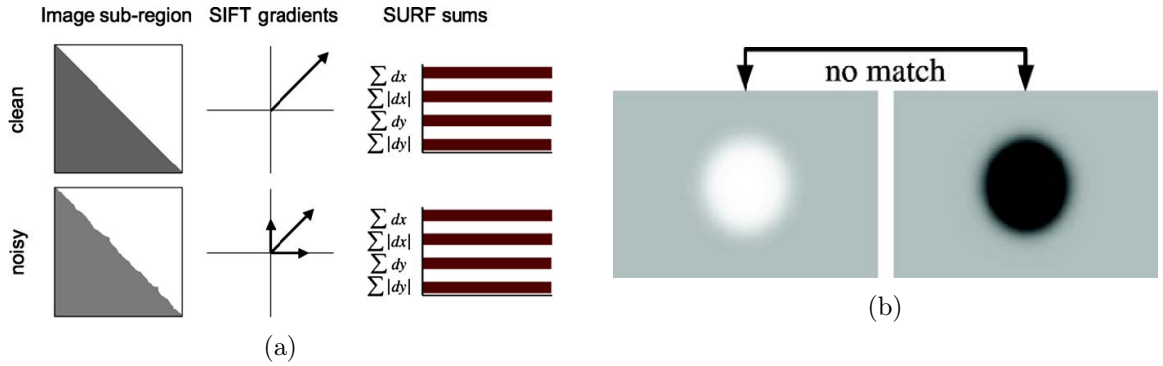


Figure 3.10: [3] To build a SURF descriptor, an oriented grid of 4x4 subregions is laid over the interest point. From each of the 4 subregions, Haar wavelet responses are computed for 5x5 evenly distanced sample points (only 2x2 are displayed in the figure). These responses, relative to the grid, are then for each subregion summed in x and y directions, as well as the sum of their absolute values. This complete 64D vector fully describes each interest point.



(a) (b)

Figure 3.11: [3] More global SURF descriptor is with wavelet responses more robust to small changes and to noise than a more locally operating SIFT descriptor (a). The importance of the sign of Laplacian is that it can differentiate between the same blob-like structures that only differ by the contrast (light blob on a dark background versus dark blob on light background) (b).

is a detector of local features that reaches high affine invariance by analysis the component tree. Those are based on intensity functions and display local extrema. The idea is to find areas which stay nearly the same through a wide range of thresholds. By creating a sequence of thresholded images, connected components can be extracted (extremal regions). Then a threshold is found in which the extremal region is "Maximally Stable". Found regions descriptors are then kept as features.

The advantage of MSER is its high repeatability and that not only it is invariant to affine transformations, but also to skewing and warping. The disadvantage however is that they are extremely sensitive to daylight effects such as shadows.

Features from accelerated segment test (FAST) [28] was supposed to provide a real-time frame-rate application in 2006. It is a corner detection algorithm and is based on the property of corners that the change of image intensity should be high in all directions. The algorithm indeed reaches multiple times the speed of SIFT and SURF, but lacks in robustness and accuracy.

Binary Robust Independent Elementary Features (BRIEF) [6] introduced in 2010 is a feature point descriptor, which is also focusing on real-time use. Unlike SIFT and SURF that need 128D and 64D feature vectors respectively, BRIEF uses only binary strings of lower dimensions for efficient feature point description. It also uses Hamming distance instead of L2 norm for even more efficiency.

Oriented FAST and rotated BRIEF (ORB), introduced in 2011 [29], aims to speed up SIFT by as much as two orders of magnitude, while also being less effected by noise. It builds on FAST keypoint detector and BRIEF descriptor, which both have low computational cost. Its advantage also is that it is free from licensing restrictions of SIFT and SURF.

Binary Robust Invariant Scalable Keypoints (BRISK) [18] is also supposed to have better performance than SURF with better accuracy. It also builds on FAST, in particular on its extension - AGAST (Adaptive and Generic corner detection based on the Accelerated Segment Test), in scale spaces to locate potential interest points.

Chapter 4

Building a database

This chapter discusses how a database of labeled images of Octavia models was created. The SURF features (Section 3.3) were used to detect the presence of a car engine space and its location (see Section 4.1). Using a training set of images of both Karoq and Octavia models, we were able to accurately locate each car passing on the line. The trade-off between algorithm speed and accuracy is discussed in Section 4.2. Using only one image was significantly faster but resulted in poor accuracy. Although a 7-image training set was still almost 1.5 times faster than the full training set of 15 images and the accuracy was only about 6% lower, there were problems at extreme cases when there was a person passing by or when two cars were parked too close to each other. That is why the final training set comprised of 15 images of Karoqs and Octavias from different angles.

By using SURF features with that full training set, a vector of car names was created for each video. In order to obtain only one image per each car, it was necessary to identify each unique car passing on the line. This was achieved with morphological opening on the car names vector to remove false positives as well as gaps. Each cluster then represented a single car (see Section 4.3).

Once each car was identified in the video, the presence of a cooling liquid tank was examined on images with optimal car position (Section 4.4). Using the hue layer of an image converted to HSV space, the purple color of the liquid could be found using an appropriate thresholding. To eliminate small purple areas (purple hue is also present in deep black) another morphological opening with sufficiently large structural element was used. Then a restriction to a specific area in the picture in combination with the knowledge of the car exact location ensured, that only the cooling liquid was detected.

In the final stage of building a database of images of Octavia models we needed to select only one picture of each car on which the cooling liquid tank was also visible (Section 4.5). We tested two approaches - one being more universal and faster, the other being more precise. The first technique searched for any frame of the car, on which the liquid tank was visible and then selected the potential best image. This resulted in a database of cars at a different position, relative to the camera. The latter technique only looked for images of a cooling liquid tank in an exact position for each car. The fact that workers were usually passing through the field of view of the *camera A* and thus blocking the tank meant, that more videos had to be examined in order to create a database of at least 2000 cars.

4.1 Engine space detection

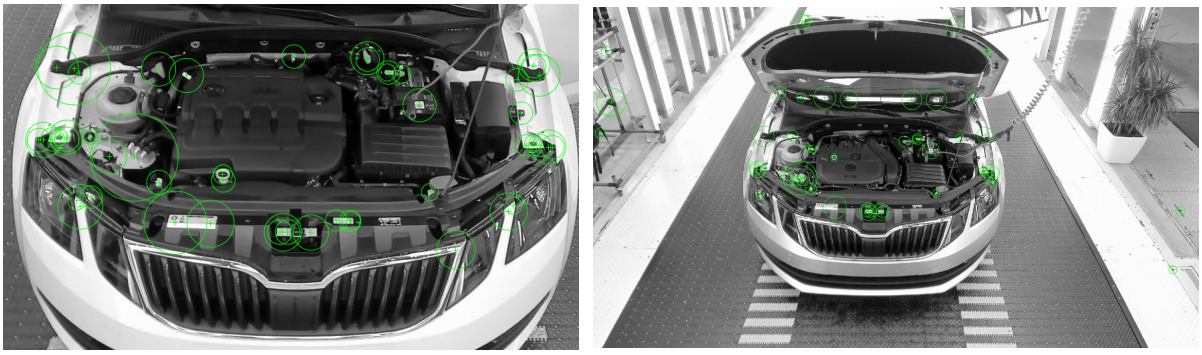
The method of Speeded-Up Robust Features (SURF), which was discussed in Section 3.3, was suitable for the detection of engine space for both its speed and for its partial affine invariance. Another strong reason for choosing local features detector in general is that they can perform well even under a partial occlusion. This was important for reliable detection as the workers were regularly passing by or standing in front of the camera. We have observed that SURF can detect the engine space even if one person obstructs the view.

The simplified Algorithm 1 takes one frame of the video, converts it to grayscale and finds feature points and their description. Then it is compared to the training set of several cropped images of car engine space and finds the one with highest number of matched interest points. Finally, a transformation of the training image is computed and the car type, transformed polygon and its centroid are stored.

We have experimented with various crops of the training images before settling on the whole crop of the engine space, as can be seen in Figure 4.1a. The figure also shows 50 strongest SURF features and their scales. We first experimented with only using crop images of the front grill as it should be unique for Karoqs and Octavias. However, there were several issues. Firstly, it is only a small area with very limited number of potential interest points (as can be seen on the figure, none of the 50 strongest points was actually detected on the grill) and it often got confused with either the black conveyor or the led lamps on the sides (Figure 4.1b). Then it couldn't correctly differentiate between the two models and lastly, it was ofttimes blocked by the car in front or by a person standing in front of the car. We also tried other smaller crops of either the engine or directly the cooling liquid tank, but again the results were poor. That is why we opted for the crop of full engine space. How many images were needed for a stable detection is described in the following section. With this we were also able to distinguish not only between models, but also between RS, Scout or the standard version. This might be useful in future but not for purposes of this work.

Algorithm 1 findFeatureVideo

$I \leftarrow$ convert $frame$ to grayscale
$scenePoints \leftarrow$ detect SURF features in I
for all i in length of $training\_set$ **do**
  $box\_pairs[i] \leftarrow$ match features of $scene\_points$ to $training\_image\_features[i]$
  $points\_count[i] \leftarrow$ number of matched points
end for
$car \leftarrow$ find the training image with highest number of matched points
$box\_polygon \leftarrow$ rectangle of the same size as the training image with best match
if number of matched points < 5 **then**
  discard putative match
else
  $new\_box\_polygon \leftarrow$ estimate geometric transform of training image to image I
  $centroid \leftarrow$ find the centroid of the $new\_box\_polygon$
end if



(a) (b)

Figure 4.1: 50 strongest SURF points on a training cropped image of engine space (a) and on a whole car (b).

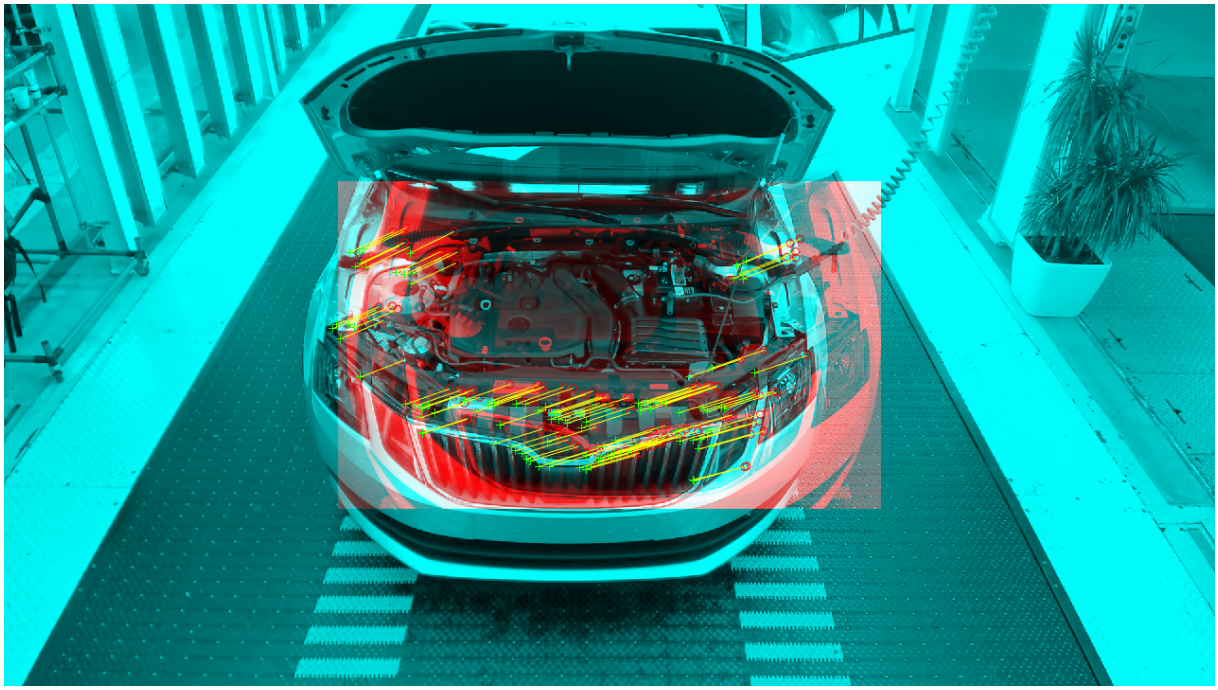


Figure 4.2: Matched SURF points on Octavia.

	Full training set	1 image	2 images	4 images	7 images
Sum of cluster sizes	243	137	151	165	208
Ratio of exact matches	94%	73%	82%	83%	88%
Elapsed time	1	39.4%	54.5%	58.9%	68.3%

Table 4.1: Comparison of efficiency of different training set sizes. Sum of cluster sizes states how many of frames in a 600 frame video belonged to a cluster. Ratio of exact matches is how many detected cars were inside any cluster in the full training set attempt. Elapsed time compares time of the function run on a smaller and the full training set.

4.2 Choosing an optimal training set

While the size of the training set has a direct impact on the accuracy of SURF descriptors and can lead to better distinguishing between car models as well as reducing false positives, it also has a negative impact on speed. Function `matchFeatures` has to run through all images from the training set before choosing the best match, hence the larger the training set, the slower the function runs. We iterated the size of the training set and observed how well it performed in detecting each in video.

The results are shown in Table 4.1. We tested the full training set against 4 smaller subsets. The full set was comprised of 5 images of Karoq and 10 images of Octavia engine space. Those were selected to cover most of the variability in real traffic such as different position of the car to the camera (further away as well as just under it) or different model variations such as RS or Scout.

The first subset contained only one image of Octavia engine space. While in the full training set the sum of cluster sizes was 243 of 600 frames, it was only 137 frames for the one image training set. Also the percentage of instances when the function found an exact match with the vector of clustered data is significantly lower for the smaller training dataset. This was mainly due to the fact that it could not differentiate between Octavia and Karoq, which were passing the assembly line. However, the elapsed time was shorter by more than 60%. The accuracy results slightly improved when an image of Karoq engine space was introduced in training set 2. Introducing two more images of different engine covers of Octavia resulted in another incremental improvement while also increasing the time needed to run the function to almost 59% compared to the full set. In the last subset, one additional image of Karoq and two of Octavia were included to improve the ratio of exact matches to 88% and the sum of cluster sizes to 208 frames. The 35 missing frames were mostly associated with Karoq which for our purposes would not be of harm. Also we noticed that different plastic engine covers do not play a key role for SURF descriptors and their performance.

However, there was a different issue that emerged. In some instances when two cars were placed just behind each other, the trunk door was open on the first car and workers were standing in between, the car behind would only be visible for as little as 7 or 8 frames. Then it was only the full training set that identified the second car as a standalone cluster. This can be seen in Figure 4.3a and 4.3b.

(a) (b)

Figure 4.3: Even when two cars were parked closely behind and the second car was covered by the trunk door of the first car, the algorithm utilizing the full training set was able to detect the car behind as a new car.

4.3 Car separation

To create the database of Octavia images, we first needed to be able to differentiate each next car on the line. This means that we do not only want to tell where the car is and whether it is an Octavia or Karoq, but also when a particular car enters the view and when it exits. This was part of a long script `SURFANDHSV` that we break down step by step in this and the following sections.

In this algorithm, we first go through the whole video from *camera B* and on each frame apply function `findFeatureVideo` (Algorithm 1) to determine if there is a match to any of the training images. We then store the car model name to a vector. As each video was 10 minutes long with a frame rate 1 fps, the vector (and all the following vectors for the whole video) has 600 rows, with each row being either empty or containing either *Octavia* or *Karoq*. For each found car we also store the coordinates of transformed bounding polygon of the engine space. It was needed later for narrowing the region for searching the cooling liquid.

To locate each Octavia only, we replaced the rows with Karoq name with zero. Then we could make the vector binary, 1 for Octavia and 0 for no match. This vector now contained clusters of ones, marking the frames on which Octavia car was present. However, the clusters were not compact - false negatives could be caused by a passing worker obstructing the view or sometimes the SURF mistook Octavia for Karoq. There were also several false positives in every video (almost always single frames). To distinguish between individual clusters, we decided to use simple technique of morphological opening for smoothing the vector. THe structural element size was chosen experimentally. This operation removed false positives and smoothed small gaps. We could then separate each cluster and assign it the corresponding car name and a number in a sequential order.

Although it was not in the scope of this work, we tried to correctly label not only Octavias, but also its specific versions like RS or Scout and the same for Karoq. For this we modified the Algorithm 1 to store the whole name of the matched training image. Then for each cluster we simply used the majority vote to determine the most common name. It could well differentiate between Octavia and Karoq, but had flaws in detecting the particular version. This could probably be fixed by enlarging the training dataset to more than 15 images, but that would again raise the computational time. Because we did not need it for achieving our goal, we did not examine it any further.

We also experimented with different and more sophisticated methods for identifying each car on the line. Namely we tried to interpret the movement of the centroid of engine space as found by `findFeatureVideo`. The idea was to assume a new car, when the centroid is found at the top half of the frame and is exiting the view when it was in the bottom area. We also tried to use linear regression on the centroid locations and match it with the constant speed of the conveyor. The main goal of the former mentioned was to be able to make it more robust to sudden stops of the production line (apart from the usual meal breaks), when one car is detected standing several minutes on the same place. But it turned out that since the cars were often parked closely behind each other, the entering and exiting areas were overlapping and we could not specify the threshold. The before mentioned simple approach with morphological opening proved to be most effective and practically flawless - when manually inspecting the database we never encountered two images of the same car.

4.4 Cooling liquid tank detection

Using SURF on detecting the cooling liquid tank yielded poor results as the cropped area is very small and in low resolution. Instead, we exploited the fact that the cooling liquid in Octavia has a strongly distinguishable purple color. Using the HSV space (Hue, Saturation, Value) we were able to threshold the hues and separate purple areas from the rest of the image. We wrote two algorithms for detecting the liquid from both *camera A* and *camera B*, which differ only in the parameter of polygon used for narrowing the searched area.

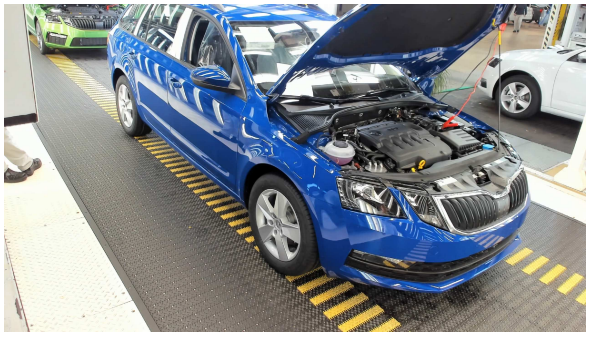
The function `HSVTOP` described in Algorithm 2 has an input of the image and the transformed polygon area. The image is first converted to HSV space and thresholded. Because the liquid always covers a larger area than most other remaining from the thresholding, morphological opening is used on the binary image to remove smaller structures. The algorithm then examines only the area restricted by the transformed polygon obtained from function `findFeatureVideo` and if there is more than one area with a surface of more than 400 pixels (this value was experimentally selected as optimal), then a morphological opening is applied once again but with larger structural element. This is repeated until only one area inside the polygon remains and then the coordinates of its centroid and of the bounding rectangle are stored. If no purple area satisfies the conditions, zero is returned as no cooling liquid has been detected inside the specified region.

It is important to scale down the observed area as much as possible, because the purple hue can be also present in other areas. We noticed, that large purple objects, that are not removed with morphological opening, occurred either when an employee was wearing a purple shirt (although standard white and green are mandatory) or sometimes in the bottom side of a car bonnet, because purple hue is also present in deep black.

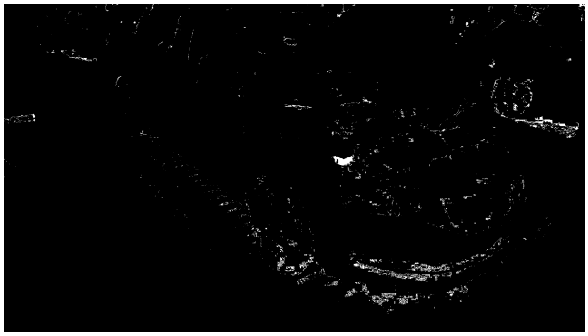
The steps of this function for the side view can also be seen in Figure 4.4. Because we only computed SURF features for each frame of a *camera B* video, we did not know directly the location of the engine space in *camera A* video, although they are connected as the frames are taken in the same time. Hence, for the function `HSVSIDE` we only used empirical coordinates of restriction for the detection of the liquid. This however caused us problem twice, when the *camera A* was hit by a worker and tilted to a different direction. We needed to adjust the script manually for proper detection.

Algorithm 2 HSVTOP

Convert image I to HSV space
$Ih \leftarrow$ Threshold hues between 0.85 and 0.95
$I\_open \leftarrow$ opening of image Ih with structural element se
if largest compact area in $Iopen$ is > 400 **then**
  $object\_in \leftarrow$ purple areas within $box\_polygon$
  while number of elements in $object\_in > 1$ **do**
    Increase the structural element size
    $I\_new \leftarrow$ opening with new structural element
    $object\_in \leftarrow$ remaining compact areas
  end while
  $liquid\_centroid \leftarrow$ centroid of the one remaining compact area of $object\_in$
  $bounding\_rectangle \leftarrow$ rectangle surrounding the detected purple area
else
  No cooling liquid was detected in a given frame
end if

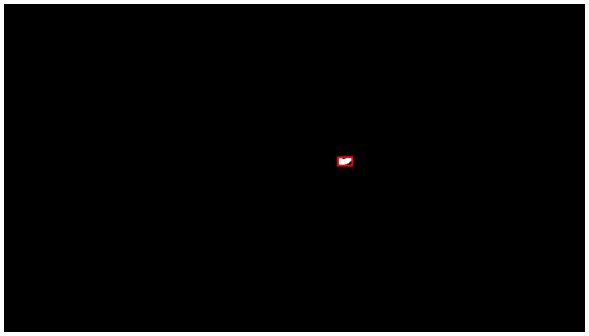


(a) RGB image



(b) Image thresholded in HSV

(c) Image after the first opening process



(d) Final detected liquid.

Figure 4.4: Cooling liquid detection process: (a) RGB image is converted to HSV space and thresholded (b). Morphological opening is applied to the binary image (c) and then repeated, if necessary, until only one object remains (d).

4.5 Database of representative images

After the previous steps - using SURF for finding engine space, clustering vector of car names to distinguish every individual car and locating the cooling liquid in HSV space - we knew where each car was in the video in any given moment. Then we needed to extract one image per each car passing on the line. Cooling liquid had to be in a desired location and could not be hidden behind any obstruction. Our first approach was to collect images from both cameras with highest possible efficiency. This meant that if the presence of cooling liquid was not observed in an optimal location, more frames were examined until it was detected or the car went out of frame. Although the algorithm produced slightly more than 90% of car images from both cameras in a full day of recording (if the liquid could not be located in either of the two images, the second was discarded as well), the position of cars in the frame varied significantly. Still this result was higher than we expected given that all videos were recorded during normal working hours and in an environment that was not adjusted for our needs.

The second approach was more specific as we wanted to produce a database of images from the side view only, but with the cooling liquid being always in the same position (with slight variance), so that we would be able to train a convolutional neural network with them. For this we again used SURF features of the engine space from the top view and clustered each car. However, this time we detected the cooling liquid directly on the side view, only on the frames on which the car should was present at the moment and in only a small rectangle area of approximately 150x120 pixels. This window was again chosen empirically as it had to be small enough to keep each car on every image at roughly the same position, but in the same time had to account for different placement of the car on the conveyor as they were manually parked by employees. Thus, usually only half of Octavias could be exported from every video.

The final script `Database` described in Algorithm 4 was written so that it could go through all video files and run until every video was not examined. First the training set of 15 cropped images and their SURF features was loaded to memory. Then in a loop over every subfolder (named by days) a video paths from both *camera A* and *camera B* was loaded and passed to a script `VIDEOCONV` described in Algorithm 3. This script converted both files from h265 codec at 4:2:2 to h264 in 4:2:0 which can be read by MATLAB. The video from *camera A* was also flipped vertically, as it was mounted upside down. A path to these two converted videos was then passed together with the training set to the function `SURFANDHSV`, which we described in the last 4 sections. It saved side images of Octavia models with the cooling liquid tank being well visible and in the same position and for each car also wrote, apart from the car number and its source video also the coordinates of the centroid of the localized liquid area as well as its bounding rectangle.

On our 6 core (12 thread) 3. gen Ryzen processor, it took about 3 days to produce a database of 2000 images. The most computational time consuming part was finding SURF features in every frame of the video and matching it to the training set of 15 images. As we discussed in Section 4.2, we could have used less images but at a cost of mismatching some frames which could lead to either not detecting the car at all or interpreting one car as two different. We also had to twice manually edit the script when a worker hit the camera and changed it's orientation. Although we tried to write the script as robust as possible, because of speed we only computed SURF features on top view images from *camera B*, and then only empirically defined the area in which the liquid should be detected in the side view. This could either be corrected with a better hardware - a strong mounting of a proper industrial camera instead of a web cam - or with rewriting the algorithm so that it would also compute SURF features on the side view and compared it to

another training set (of side view images) for at least those frames, on which we want to detect the cooling liquid.

However, it was important for us to show, that we can use SURF to create (in combination with thresholding in HSV space) a large database of images, which then could be used to train and test a convolutional neural network (Chapter 6). In the following chapter, we discuss how to read the level of the cooling liquid from a prepared cropped image.

Algorithm 3 VIDEOCONV

Require: VideoA, VideoB and the folder path
  if Converted $fileA$ doesn't exist **then**
    $command \leftarrow$ convert to h264 at 4:2:0 and crf18
    execute $command$
  end if
  if Converted $fileB$ doesn't exist **then**
    $command \leftarrow$ convert to h264 at 4:2:0 and crf18 and rotate 180
    execute $command$
  end if

Algorithm 4 Database

  $training\_set \leftarrow$ TRAINSETLOAD
  for all i in subfolders **do**
    $nameA \leftarrow$ all videos ending with $A.mp4$
    $nameB \leftarrow$ all videos ending with $B.mp4$
    if *ffmpeg.exe* doesn't exist **then**
      copy file to the folder
    end if
    for all i in length($nameA$) **do**
      $vidA \leftarrow$ i-th camera A video
      $vidB \leftarrow$ i-th camera B video
      $[fileA, fileB] \leftarrow$ VIDEOCONV($vidA, vidB$)
      $car\_data \leftarrow$ SURFANDHSV($training\_set$, $fileA$, $fileB$)
    end for
  end for

Chapter 5

Cooling liquid level detection

In previous sections, we have described the whole process from collecting video on site to creating a database of car images with a known location of the cooling liquid. In this section we introduce our method for the final step - detecting the level of cooling liquid and determining, if the amount of liquid is sufficient or not.

For this we decided to use *Method A* proposed by A. Novozamsky et al. [26]. In their paper, they needed to detect bleeding in wireless capsule endoscopy videos. Instead of any shape detection of the blobs, they opted for separating the blood color from the rest of the video. This is similar to our cooling liquid detection, as we also need to separate a distinctive red-purple color from the surroundings. Their method was based on creating their own color space (modifying RGB space) so that it would provide more discriminability for red color in specific. In its basis is similar to CMYK color model, which is also subtractive. Their algorithm can be summarized as follows [26]:

1. $$K = \min(1 - R, 1 - G, 1 - B), \tag{5.1}$$
$$M = (1 - G - K) \tag{5.2}$$

   where R, G and $B \in \langle 0, 255 \rangle$. The pixels with a low value in green and high values in red and blue are well separated.

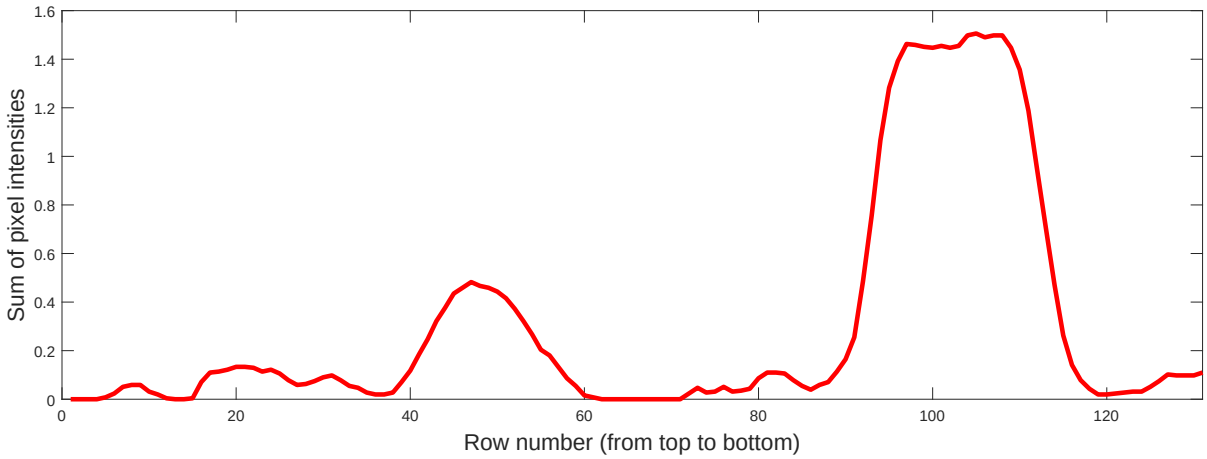
2. $$R_1 = \sqrt{G^2 + B^2},$$
$$R_n = \begin{cases} 0, & \text{if } R_1 = 0 \wedge R < 128, \\ 255, & \text{if } R_1 = 0 \wedge R \leq 128, \\ R/R_!, & \text{if } R_1 \neq 0. \end{cases}$$

Their classification criterion is then decided by counting the number of pixels which product of $R_n \cdot M$ exceeds 200. We have drawn on the first step, the subtraction, but we modified R_1 as a row sum of values in M.

In practice, a narrow cutout has been taken around the centroid of cooling liquid as described in the previous chapter. The lower part was 10 pixels beneath the centroid, the upper 60 pixels above and 5 pixels were counted to each side. The bounding rectangle of the detected liquid is in Figure 5.1a and the narrow cutout for the corresponding image of the liquid tank in Figure

(a) Detected liquid on Octavia

(b) A narrow cutout area



(c) Row sums of pixel intensities

Figure 5.1: A location of cooling liquid in an image is known from a function `HSVSIDE` (a). A narrow cutout of 6x130 pixels (b) is converted to a new color space and normalized. Then a sum over each row is computed and the resulting intensities plotted (c).

5.1b. The intensity values for this cutout were first normalized and then their channels were subtracted as described in Equations 5.1 and 5.2. Then the row sums of M were computed and the values plotted in 5.1c. We have observed, that two major peaks always occur. The first corresponds to the purple hues contained in the blacks of the liquid tank cover and the second peak to the liquid itself.

The level of the cooling liquid can be then calculated simply as a ratio of the width of first peak to the distance from its bottom part to the bottom part of the cover (beginning of the second peak). To isolate the two peaks from surrounding values, we heuristically found a proper threshold.

Chapter 6

Convolutional neural networks

All previous chapters describe the complete (and working) process of detecting the level of cooling liquid with the use of SURF for analyzing the video and selecting only a few frames, where the liquid is visible. This process is due to the nature of SURF and due to our design (described in Section 4.2) considerably slow. Thus we have decided to use neural networks for the real time application and only take advantage of the SURF method for providing it with a labeled set of training images.

In this chapter we first briefly discuss the history and necessary theoretical background for understanding convolutional neural networks (Section 6.1). Then, in Section 6.2, we describe U-net (introduced in 2015 [27]) and our own segmentation neural network built on it ([39] 2019, Z. Bílková). This network was first developed for text segmentation, but retraining it on car images yielded satisfying results. After using 1000 labeled images for training, the neural network was able to correctly detect the cooling liquid tank on all 1000 testing with at least 90% overlap with area detected with SURF algorithm. The implementation is further described in Section 6.3. In conclusion, we demonstrate how our network can operate much faster, analyzing a video shot at 1 fps at the speed of 50 fps, estimating the level of cooling liquid and graphically visualizing it.

6.1 Convolutional neural networks

Standard feedforward neural networks consist of input layer, neurons in one or more hidden layers, connections with weights and an activation function. In a fully connected neural networks, each neuron is connected to all neurons in the previous and the following layer. The process of training a neural network starts by a random initiation of weights and biases. Then, using an activation function, forward propagation is computed and its result is evaluated using a cost function. This is then used for backpropagation with a manually chosen learning rate and each weight is then changed. The goal is to minimize the cost function, which compares results with the training set (in supervised learning). This process is well known for several decades and has been thoroughly described in many books and articles ([4], [24], [30] or [14]).

However, these standard neural networks are not suitable for deep learning with more layers. In fact, even only 10 hidden layers can yield worse results than a smaller network, while also training much slower. Deeper networks have a large impact especially in natural language processing and visual imagery, as they can describe more complex features. That is where Convolutional neural networks (CNN) come into play. The first use of backpropagation learning the convolution kernel coefficients was in 1989, when LeCun et al. used it on hand-written numbers ([15]). This was later expanded to LeNet-5 [16], a first convolutional neural network.

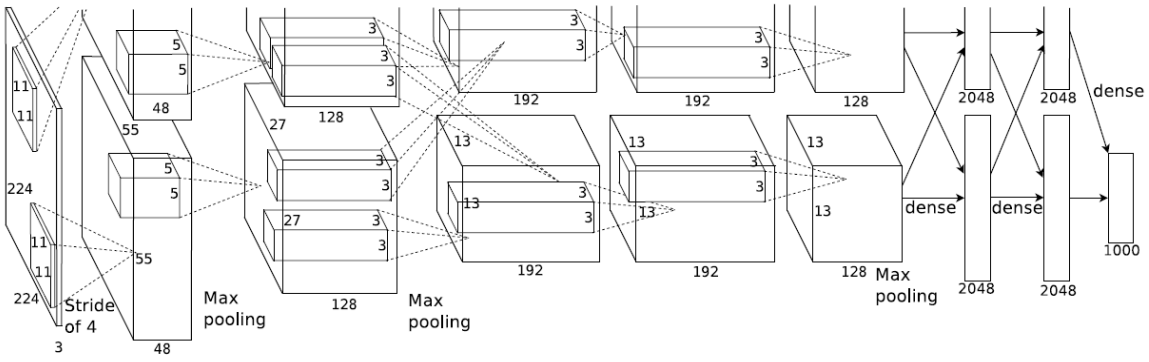


Figure 6.1: [13] The architecture of AlexNet has two different "streams", because it was trained on two parallel GPUs.

It was used by various banks to recognize hand-written digits on checks. However, CNNs gained popularity several years later, when they could be trained using GPU instead of CPU. In 2011 CNNs outperformed other methods on the MNIST handwritten digits benchmark and were winning machine learning competitions [32]. A. Krizhevsky et al. mentioned in [13], that compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and thus can be trained much faster, while their theoretical performance should be only slightly worse. Their CNN, AlexNet, is shown on Figure 6.1.

Convolutional neural networks exploit the fact, that neighboring pixels in images usually are not independent. While in fully a connected layer, each neuron is calculated using all input pixels by adding weight to each of them, and the same is repeated for every other pixel, CNNs observe interactions between neighboring pixels and typically only share 3x3 weights (size of the kernel). Each convolution layer convolves the input with a kernel and passes its result to the next layer. The third dimension of this layer comprises of extracted features from the convolution. Typically, the features from the first layer of CNN is similar to human visual cortex which recognizes Gabor functions. The padding scheme of convolution can either be *same* (extending the original layer to preserve dimensions) or *valid*. As can be seen in Figure 6.1, the two spatial dimensions of hidden layers decrease. This is because of operation called pooling. While convolution combines values around a single pixel, it is also useful to combine smaller areas into larger ones. Two most common fixed pooling operation are average pooling, which takes the average of values in a cluster of neurons in prior layer, and max pooling, which only takes the maximum value. In ImageNet, max pooling is used three times. The combination of convolution, non-linear activation and pooling can find various features on various levels, e.g. from circles and corners to more complex features like letters. Because these features also need more information to describe them, the depth of the layer (the number of channels) increases with each pooling operation. For example, a 224x224 RGB image becomes after 3 layers a 13x13x196 representation.

The two penultimate hidden layers in Figure 6.1 are fully connected layers of 2048 neurons and the last layer of 1000 neurons corresponds to the number of classes that the network classifies into. Two popular networks introduced in 2014 were VGG [33] and Inception (GoogLeNet) [34]. Drawing on the VGG, ResNet ([12] 2015) introduced adding residual connections which copies the input to the output with a weight. This allows for much deeper networks to perform well and improves their performance.

The activation for all output pixels is defined using the same set of weights. This means that

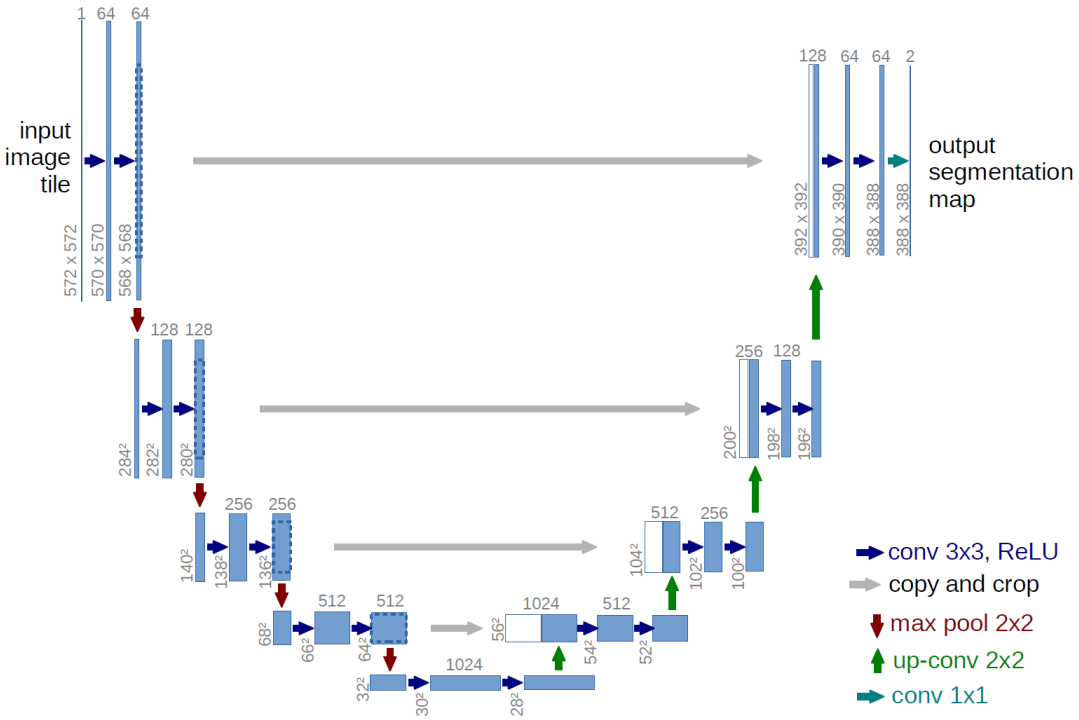


Figure 6.2: [27] U-net architecture with multi-channel feature maps illustrated with blue boxes and copied feature maps with white boxes. The arrows represent a corresponding operation.

the activation in a pixel does not depend on its position, which is useful for position invariance, but reduces localization of each feature. To extend image classification to image recognition, Fast R-CNN [10] was introduced in 2015. It uses Regions of Interest (RoI), which are each pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers. Those branch into two sibling layers: one produces softmax probabilities and the latter per-class bounding box regression offsets. For our purposes we decided to use a different approach of CNN: U-Net segmentation network.

6.2 U-net and our CNN

U-Net, presented in 2015 by O. Ronneberger et al. [27], builds on the so-called "fully convolutional network" architecture. It is similar to other CNNs in the contracting step, but instead of ending it with several fully connected layers, it starts an expansion process until the output segmentation map reaches the level of the input image (with cropped resolution). Also in the upsampling part, there is a large number of feature channels, which allow the network to propagate context information to higher resolution layers.

The architecture is illustrated in Figure 6.2. A valid part is taken from each 3x3 convolution, to predict pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. Repeated convolution is followed by rectified linear unit (ReLu) activation and a 2x2 max pooling. The expansion path on the other hand, starts with upsampling the feature map and then with 2x2 "up-convolution", which halves the number of feature channels. Then comes the important part of copying the activations to the output path and concatenating them together. Furthermore, two 3x3 convolutions are followed by a ReLu.

Overlay type	overlap > 80%	overlap > 90%	overlap > 95%	overlap > 99%
CNN	100	99.65	97.79	60.53
CNN + HSV	100	100	99.82	98.32

Table 6.1: Using overlay with CNN and HSV knowledge significantly improves the performance of segmenting the cooling liquid tank.

Because of the loss of border pixels in every convolution, cropping is necessary. In the end, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total, the network has 23 convolution layers. [27]

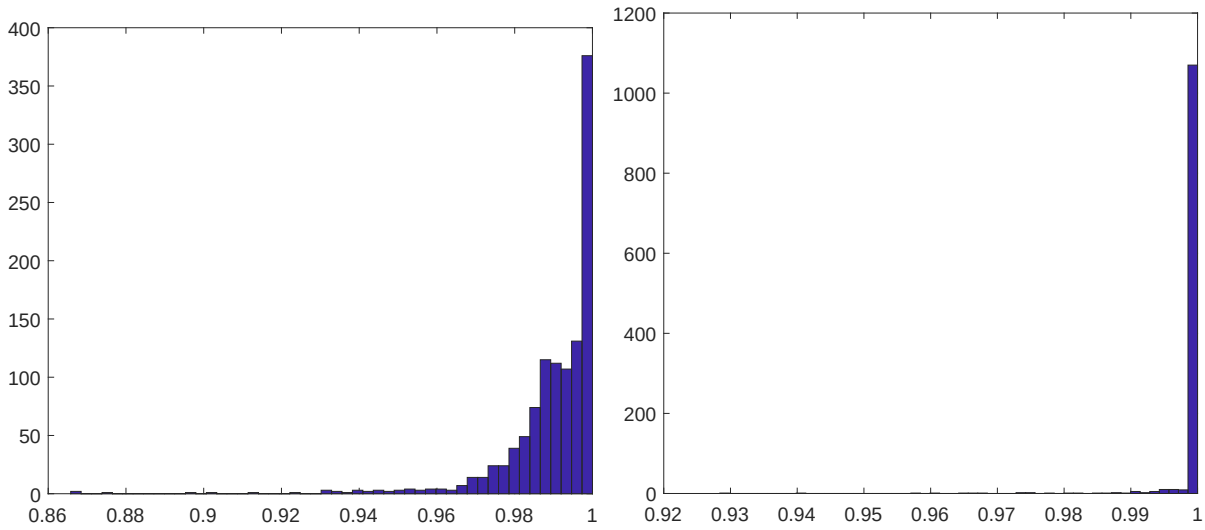
Our U-net network was built by Z. Bílková [39] on TensorFlow architecture and retrained by us on our car images. The labeled segment on the input was a cut out around the liquid centroid, as found using HSV thresholding and morphological opening. 10 pixels were counted down from the centroid, 60 pixels up and 50 pixels to each side. Our CNN tried to correctly locate this area, so that it could be passed to liquid level detection process described in Chapter 5. To measure performance of our CNN we computed the overlay of the input area (from the SURF algorithm) with the one that the network found.

6.3 Faster liquid detection

All CNNs mentioned in previous sections and most modern supervised learning segmentation CNNs require a labeled dataset, upon which the network can be trained. Our goal was to use SURF and HSV space for automatic labeling of the dataset, so that we would not have to do it manually over every picture, but instead only verify the results. We were able to generate more than 2100 pictures (from the side) of individual cars in approximately the same position with respect to the camera (as described in Section 4.5). Because of the time needed for creating such a database, we trained our network on the first 1000 images (90% for training and 10% for testing) and leaved the remaining 1150 for validation.

We first tried only using CNN for the detection and it shown good results with more than 99.6% of images having an overlap of more than 90%. The other quantiles and a histogram can be seen in Table 6.1 and in Figure 6.3a. When we also used HSV liquid detection in our CNN, the results were even more impressive with all of 1150 images having an overlap higher than 90% and more than 98% of them overlapped on more than 99%. We also saw a significant reduction of false positives, as we increased our training dataset. When we started only on 80 training images, it often flagged parts of engine space or even the side panels as a region of interest. These all disappeared as we enlarged the dataset.

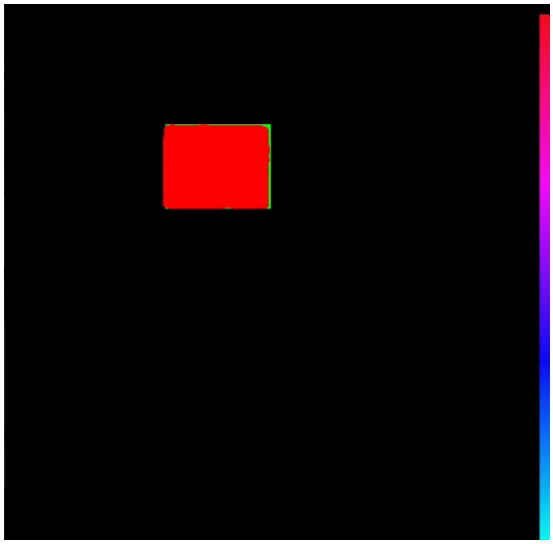
Our final result is a video with found cooling liquid tank masks and a graphical representation of the level of cooling liquid. We show 4 images from this video in Figure 6.5. It can be seen that the network correctly ignored Karoq models and also coped with a partially covered liquid tank by a passing worker. The level of cooling liquid is illustrated with a white bar on the right side of frame. Because we did not have any data about which cars had an insufficient amount of the cooling liquid, the threshold for flagging potential defective cars should be consulted from the part of the car-making company. Our scripts can be modified to run in real time and assist the employees with indicating on cars, that need to have their cooling liquid refilled.



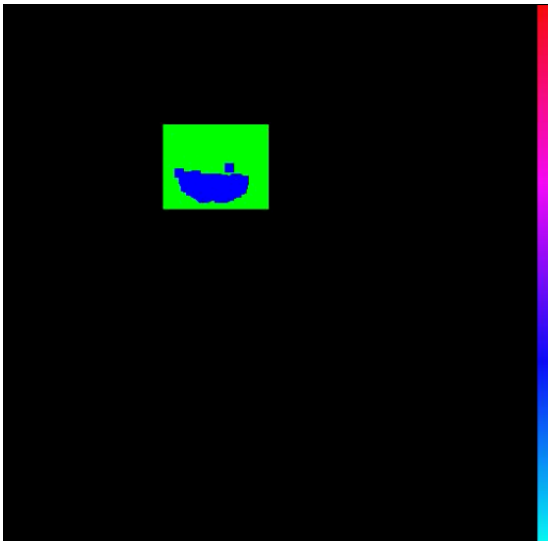
(a) (b)

Figure 6.3: Histogram of the overlap CNN and SURF with respect to CNN (on the left). When HSV prior was added to the training, 98.3% of found liquid cutouts had an overlap of more than 99% (on the right).

(a)

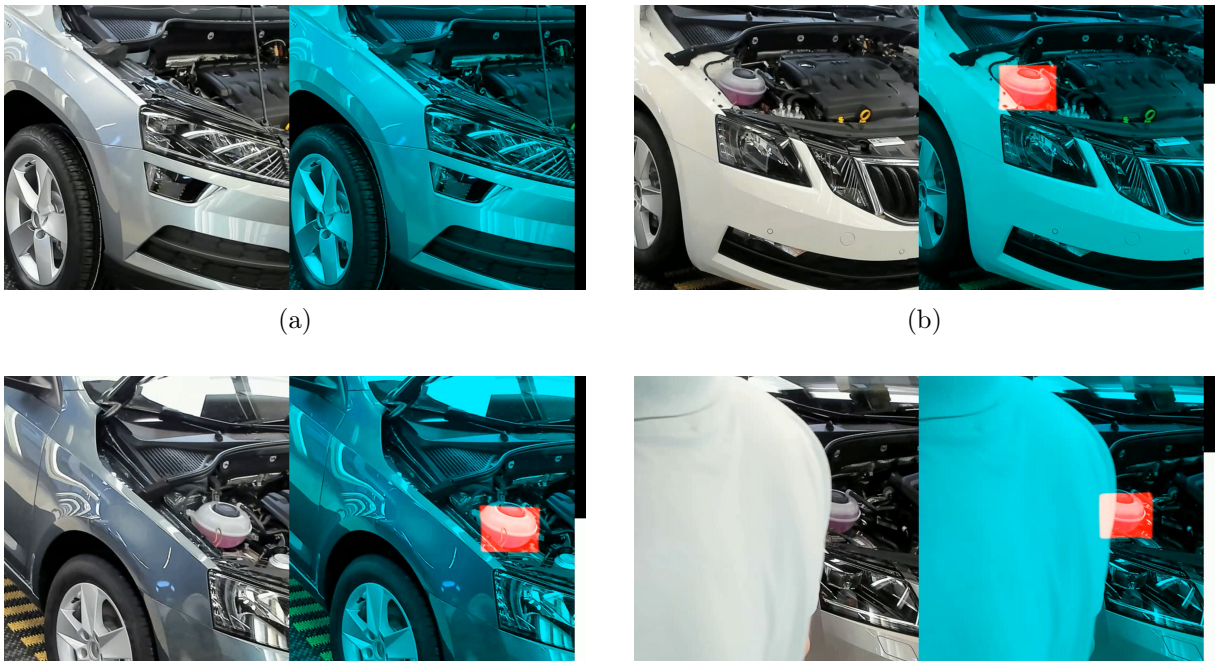


(b) Overlap of CNN with SURF



(c) SURF and HSV detection

Figure 6.4: For each image from the validation set (a) an overlay of CNN detected bounding area of cooling liquid tank with bounding area from SURF is displayed in (b). The amount of overlay is illustrated with the color bar on the right. Cooling liquid area, as detected using HSV thresholding and opening is shown in (c).



(a) (b)

(c) (d)

Figure 6.5: 4 representative images taken from the resulting video have the following structure. (From left to right) original cropped image, cropped image with red overlaying mask surrounding the liquid tank as found by CNN, narrow black/white bar illustrating the hight of the cooling liquid. (a) No cooling liquid found on the Karoq as its liquid tank is sunk beneath the edge of side plate. (b) The level of cooling liquid is well above the half line. (c) Low level of the cooling liquid. (d) partially covered liquid is still detected by our CNN.

Chapter 7

Conclusion

In this work we proposed a "Proof-of-concept" solution for automated visual car inspection in Škoda Auto. We have shown that it is possible to use computer vision to assist employees who have only limited time to visually inspect assembled vehicle for quality control. We have focused only on the detection of the level of cooling liquid in Octavia models.

Let us summarize the content of this work. In Chapter 2, we first introduced the project setup and the environment of Control Block KB8 in which we were expected to develop our system. Because we could not disrupt normal daily traffic of the factory, we only mounted two Logitech web cameras, one directly above the conveyor strand and one on the side. Output of these cameras was lead to a nearby computer to record several weeks of footage. This pilot approach however brought several problems that we had to overcome later in Chapter 4, for example we had to remove images of workers obstructing the camera view and correct the shift in position when the camera was hit three times. We also decided to focus only on Octavia models, as Karoq model has the cooling liquid tank placed deeper within the engine space, which makes it invisible from the side view.

In Chapter 3 we gave an overview of methods used in image recognition. Special attention is paid to methods SIFT and SURF. Scale invariant Feature Transform (SIFT) introduced in 1999 provides a complete feature point detection, description and matching for object recognition in images by combining scale-space, finding maxima of Laplacian of Gaussian, observing the trace of Hessian matrix and using gradients directions for keypoint description. Speeded-Up Robust Features (SURF) introduced in 2008 improves on this method by using integral images for faster convolution, a different scale-space representation and interest point description based on Haar wavelet responses. Because of its increased speed and robustness, we opted for using it for car engine space detection.

This is described in Chapter 4, where we gave a complete overview of how we processed hundreds of Gigabytes of video and exported only several thousands of labeled images with each car being represented only once. We discussed benefits of using smaller or larger crop for SURF detection and why we decided to use a set of 15 cropped images of the whole car engine space of both Octavias and Karoqs. We used morphological opening on the vector of detected cars to isolate clusters corresponding to each car passing on the line, because we did not have access to the company RFID system. With this script we were able to process each 10-minute video in roughly 7 minutes on our 3. gen Ryzen PC build and read the position of a car in every moment. Then we have shown, that the color of cooling liquid is separable from background with a corresponding thresholding in HSV space. Again, with the use of morphological opening we were able to detect position of the cooling liquid and extract an image of the car, when the

liquid was in a desired position and was not obstructed by a passing person. With the knowledge of centroid of the cooling liquid we showed in Chapter 5, that with the use of a special subtractive color model, we were able to read pixels containing both the liquid and a black cover on the liquid tank. Then by a simple comparison of the height of the liquid and the distance to the bottom part of the cover we could read the level of the cooling liquid.

Because the process of detecting local features was notably slow, we experimented with the use of machine learning for the detection of the liquid tank. In Chapter 6 we gave a short introduction to the theory of neural networks and discussed the development of convolutional neural networks, which are especially useful for image segmentation. Because we had a pre-trained network at our disposal, based on U-net architecture, we trained it on a thousand labeled images and then used another thousand images for validation. We have shown that the neural network can find the segment with more than 90% overlap in more than 99.6% of images. When we incorporated also the HSV space separation of the liquid, it correctly found all images and more than 98% of them with more than 99% overlap. However, we expect the neural network to have even better results, because the SURF labeled images used as ground truth might not be exact.

The two main advantages of this approach was first a significant increase in speed (each video shot at 1 fps could be processed at more than 50 fps) and importantly, we have shown that it is possible to use feature-based methods, such as SURF, for creating the training database of labeled images, which replaces the need of manual labeling by a human, which is needed for every supervised neural network.

However, we did not have any ground truth for the level of the cooling liquid itself, thus our main result could only be a graphical illustration of the proportion of the liquid in the tank, not a binary classification of sufficient and insufficient amount of liquid. This would have to be further developed with the expert knowledge from the company itself.

Bibliography

[1] F. Attneave, *Attneave, f. informational aspects of visual perception. psychol. rev. 61, 183-193*, Psychological review **61** (1954), 183–93.

[2] H. Bay, *From wide-baseline point and line correspondences to 3d*, Ph.D. thesis, ETH Zurich, Zürich, 2006, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 16606, 2006.

[3] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool, *Speeded-up robust features (surf)*, Computer Vision and Image Understanding **110** (2008), no. 3, 346 – 359, Similarity Matching in Computer Vision and Multimedia.

[4] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, Inc., New York, NY, USA, 1995.

[5] M. Brown and D. Lowe, *Invariant features from interest point groups*, vol. 13, 01 2002.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, *Brief: Binary robust independent elementary features*, vol. 6314, 09 2010, pp. 778–792.

[7] J. Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-8** (1986), no. 6, 679–698.

[8] F. C. Crow, *Summed-area tables for texture mapping*, SIGGRAPH Comput. Graph. **18** (1984), no. 3, 207–212.

[9] J. Flusser, T. Suk, and B. Zitová, *2d and 3d image analysis by moments*, (2016).

[10] R. Girshick, *Fast r-cnn*, Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (USA), ICCV '15, IEEE Computer Society, 2015, p. 1440–1448.

[11] C. Harris and M. Stephens, *A combined corner and edge detector*, Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147–151.

[12] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 770–778.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (USA), NIPS'12, Curran Associates Inc., 2012, pp. 1097–1105.

[14] P. Langley, *The changing science of machine learning*, Mach. Learn. **82** (2011), no. 3, 275–279.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural Computation **1** (1989), no. 4, 541–551.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, PROCEEDINGS OF THE IEEE, 1998, pp. 2278–2324.

[17] J. Lee, U. Lee, K. Baeksop, and J. Yoon, *A computational method for detecting copy number variations using scale-space filtering*, BMC bioinformatics **14** (2013), 57.

[18] S. Leutenegger, M. Chli, and R. Y. Siegwart, *Brisk: Binary robust invariant scalable keypoints*, 2011 International Conference on Computer Vision, Nov 2011, pp. 2548–2555.

[19] T. Lindeberg, *Scale-space for discrete signals*, IEEE Trans. Pattern Anal. Mach. Intell. **12** (1990), no. 3, 234–254.

[20] T. Lindeberg, *Feature detection with automatic scale selection*, International Journal of Computer Vision **30** (1998), 77–116.

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

[22] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004), no. 2, 91–110.

[23] K. Mikolajczyk and C. Schmid, *Indexing based on scale invariant interest points*, International Conference on Computer Vision (ICCV '01) (Vancouver, Canada), vol. 1, IEEE Computer society, July 2001, pp. 525–531.

[24] T. M. Mitchell, *Machine learning*, 1 ed., McGraw-Hill, Inc., New York, NY, USA, 1997.

[25] D. Nister and H. Stewenius, *Linear time maximally stable extremal regions*, vol. 60, pp. 183–196, Springer Berlin Heidelberg, 2008.

[26] A. Novozámský, J. Flusser, I. Tachecí, L. Sulík, J. Bureš, and O. Krejcar, *Automatic blood detection in capsule endoscopy video*, Journal of Biomedical Optics **21** (2016), no. 12, 126007.

[27] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015 (Cham) (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), Springer International Publishing, 2015, pp. 234–241.

[28] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection*, ECCV, 2006.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, *Orb: An efficient alternative to sift or surf*, 2011 International Conference on Computer Vision, Nov 2011, pp. 2564–2571.

[30] S. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 3rd ed., Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[31] C. Schmid, R. Mohr, and C. Bauckhage, *Evaluation of interest point detectors*, International Journal of Computer Vision **37** (2000), no. 2, 151–172.

[32] J. Schmidhuber, *History of computer vision contests won by deep cnns on gpu*, 2019, Accessed: 2019-12-30.

[33] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, CoRR **abs/1409.1556** (2014).

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 1–9.

[35] R. Szeliski, *Computer vision: Algorithms and applications*, Texts in Computer Science, Springer London, 2010.

[36] T. Tuytelaars and K Mikolajczyk, *Local invariant feature detectors: A survey*, Foundations and Trends® in Computer Graphics and Vision **3** (2008), no. 3, 177–280.

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

[38] A. P. Witkin, *Scale-space filtering*, Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2 (San Francisco, CA, USA), IJCAI'83, Morgan Kaufmann Publishers Inc., 1983, pp. 1019–1022.

[39] B. Zuzana, A. Novozámský, A. Domínec, Š. Greško, B. Zitová, and M. Paroubková, *Automatic evaluation of speech therapy exercises based on image data*, Image Analysis and Recognition : 16th International Conference, ICIAR 2019, Waterloo, ON, Canada, Aug 2019, pp. 397–404.