UiO : Department of Informatics
University of Oslo

Real-Time Road Estimation for Autonomous Driving

A computer vision based approach

Eirik Sundet
eirisu@ifi.uio.no

Master's Thesis Spring 2016

Real-Time Road Estimation for Autonomous Driving

Eirik Sundet
eirisu@ifi.uio.no

May 2, 2016

Abstract

During the last decade, there has been an increasing interest and dedication towards developing self-driving cars. The enhancements of sensor technology and processing power have made more advanced autonomous systems possible.
For self-driving cars, robust and accurate navigation is essential. In that context, cameras, in collaboration with specialized graphics processing units (GPU), has been shown to serve as a valuable tool to map the local scenery surrounding the vehicle.

This thesis proposes a computer vision based method for detecting road in a local scenery, purposed for road following. The method utilizes on-line machine learning techniques to model the road, in terms of a set of features. The choice of features is widely discussed in this thesis, as is the methods for modeling the road. The proposed methods for both modeling and detecting the road are executable in real-time and are adaptable to variation in road appearance.

The algorithm has been thoroughly tested in four different sceneries, which includes three recorded sequences containing tarmac road, and one containing dirt road. The algorithm has been evaluated in relation to similar state-of-the-art algorithms which has proved reliable for road detection.

The proposed algorithm shows great promise for detecting tarmac road in homogeneous scenery, where it almost matches the state-of-the-art algorithms. However, it struggles when the road appearance is less consistent throughout the images. Dirt road, inconsistently shadowed road and extreme illumination conditions currently pose as challenging situations.

Contents

List of Figures

List of Tables

Preface

My deepest gratitude goes to my two supervisors, Kyrre Harald Glette and Idar Dyrdal. Thank you for all your support and seemingly limitless knowledge.

I must also send my gratitude towards Trym Vegard Haavardsholm and Thomas Olsvik Opsahl, who has also been of considerable help during the work with this thesis.

A special thanks to Anders Ueland, for all your help, and for all the helpful discussions we have had.
My fellow students, family, and friends for your support, thank you.

Most of all, thanks goes to my better half, Nora Dalaker Steenberg. Thank you for all the loving support through these last weeks, it was greatly needed.

Chapter 1

Introduction

This chapter will give a short introduction to the motivation behind this thesis. A brief description of the planned outline of the UGV project is presented, as well as some of the most promising computer vision methods used for autonomous vehicle applications. At last, the goal and an outline of the rest of the thesis are presented.

1.1 Background and Motivation

In the fall of 2014, the Norwegian Defense Research Establishment (FFI) initiated the *Unmanned Ground Vehicle* (UGV) project, aimed at making an autonomous vehicle capable of conducting several tasks related to different war scenarios.
For the UGV to know where it is, how fast it is moving, and in which direction, a robust navigational system is essential. *Inertial navigation* is the typical approach to autonomous navigation, which involves integrating data from several sensors [2]. The inertial navigation system proposed for the UGV is a combination of *Inertial Measurement Unit* (IMU) and GPS. However, this combination is not accurate enough for the UGV to reliably stay on the road. Also, the inertial navigation system does not have any information about obstacles and variations in the scenery which may deviate from prior information gathered from maps, or similar. Therefore, an analysis of the local scenery is necessary.
Several approaches to mapping the local scenery are planned, utilizing different sensors and methods. The UGV is currently equipped with four types of sensors for navigational purposes: GPS, IMU, *Light Detection and Ranging* (LIDAR) and cameras. On the roof, there are three 9.1MP cameras. Two of them are monochrome (black-and-white), capable of higher detail and sensitivity. In between the two monochrome cameras, the chromatic (color) camera is located.

1.1.1 The UGV System Architecture

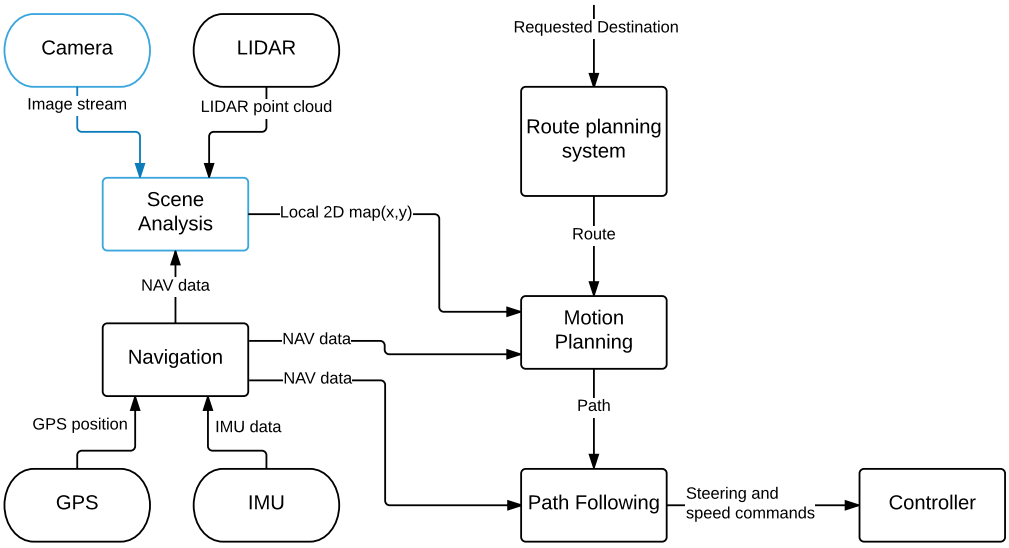


Figure 1.1: Overview of the systems at work on the UGV. The Scene Analysis System, where this thesis belongs, is marked in blue. The square boxes represent the individual processing systems, and the rounded boxes represent the different sensors. The lines represent exchange of data between the sub-systems.

The planned UGV system, illustrated in figure 1.1, is complex. Many different sensors and processing steps depend on one another. It consists of several processing systems, each taking care of a certain aspect related to the navigation. How all of these sub-systems should be stitched together, is still investigated, and the goal is to do so in a manner where the sub-systems can best complement each other.
The planned functionality is that an operator requests a destination (geographical coordinate), which is processed by the *Route Planning system*. This route planning is based on maps and topological information. When the route is established, the system waits for the approval of the operator to initiate. After the operator approves the planned route, the route is forwarded to the *Motion Planning system*. This system utilizes information from all the sensors in the process of planning how to move, so that the UGV stays on its planned route. The motion planning depends on two other systems, namely the *Navigation system*, where the global position is estimated based on GPS and IMU data, and the *Scene Analysis system*, which is the topic of this thesis. The object of the Scene Analysis system is to identify the local surroundings, both road and others, and thereby identify the scenery in terms of driveability. Along with prior information from map data, this information will then be used to avoid drift arising in the Navigation system. The purpose of the Scene Analysis system is also

to prevent the UGV from crashing into local obstacles in the scenery. The planned movement path generated from the Motion Planning system is sent to the *Path Following system*, which generates the necessary steering and speed commands. These commands are sent to the *Controller*, where the engine power, steering, and brakes are controlled.

Within the Scene Analysis system, many different methods are being researched. In addition to the road estimation algorithm, *Stereo Vision*, *SLAM* and *Object Detection* is planned to be integrated into this system. All of these methods will be merged to establish a measure of the local scenery, and the position of the UGV with respect to this scenery. This is illustrated in figure 1.2.

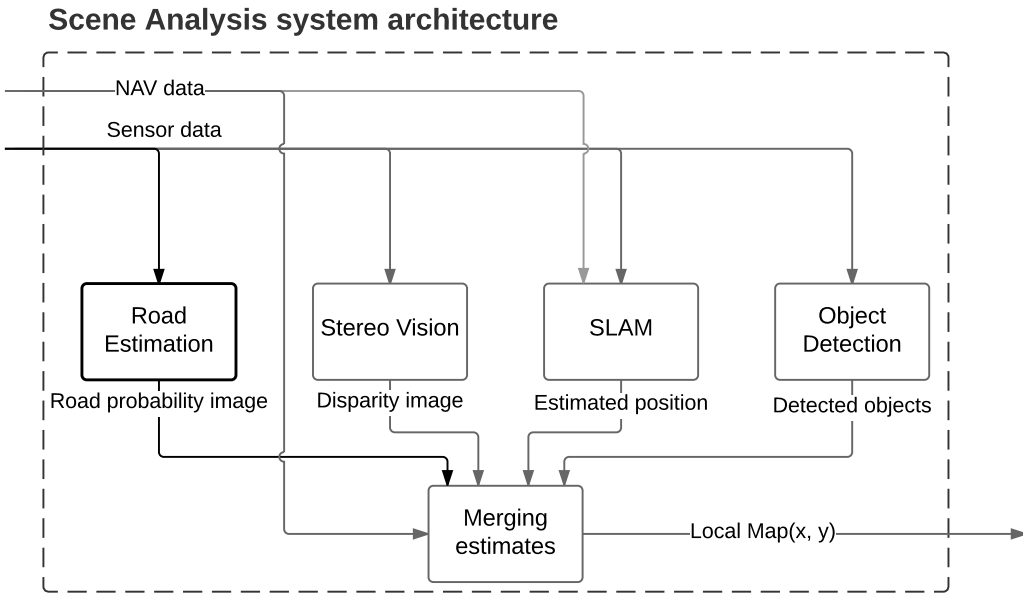


Figure 1.2: Illustration of the planned Scene analysis system. The square boxes represent the individual systems at work in the Scene Analysis system while the lines represent data exchanged between these systems

1.2 The Self-Driving Car

In this section, some of the most successful attempts to develop self-driving cars are presented.

Development of self-driving cars has been attempted several times since at least 1926 [3], to different extents and success [4]. The modern day breakthrough, however, came with the *DARPA Grand Challenge* [5]. Funded by the *Defense Advanced Research Projects Agency* (DARPA), the

DARPA Grand Challenge was a competition for robotics researchers, engineers, inventors and hobbyists across the United States, aiming to spur innovation in unmanned ground vehicle navigation. The objective of the competition was to develop a mobile robot that was able to traverse a predetermined route autonomously. The competition was arranged three times; the first in 2004, the second in 2005 and the third in 2007. The first two competitions took place on a desert road, where the vehicles had to traverse unrehearsed off-road terrain. The third focused on navigation in urban scenes.

1.2.1 DARPA Challenge 2005 - Off-road Navigation

The DARPA Grand Challenge of 2005 was by far more successful than the first challenge. In the 2004 race, none of the 15 finalists completed more than 11.9km of the 212.4km course. In the 2005 race, 22 of the 23 finalists traveled further than 11.9km, and a remarkable 5 vehicles completed the entire 212.4km course. In the 2005 challenge, the autonomous car *Stanley* won the race [6], finishing at the winning time of 6 hours, 53 minutes. Stanley was created at Stanford University, and the vehicle utilized some promising methods for vision based navigation.
To estimate the road, Stanley used a combination of LIDAR data and computer vision. It utilized the LIDAR to extract a patch of the area in front of the vehicle that resembled drivable terrain. This patch was then projected into the camera plane, and the pixels in the image corresponding to this patch then served as the training samples for modeling the road, with respect to the feature space defined by the three color channels Red, Green and Blue. The road was modeled as a *Mixture of Gaussians* (discussed further in 2.3.1). For each new image, the samples extracted from the laser patch was modelled as k new "local" Gaussian densities using the *EM algorithm* (discussed further in 2.4.1) [7]. These new Gaussians were then merged into the existing n Gaussians, learned from the previous images. Thrun et. al claims that this allowed for both robust, time-dependent modeling of the drivable terrain, and at the same time fast adaption to sudden changes in the scenery.

1.2.2 DARPA Challenge 2007 - the Urban Challenge

In 2007, another DARPA Challenge was organized. This version of the self-driving car competition was called the "Urban Challenge", and its purpose was to focus on self-driving cars in urban scenes. The competition did not only concentrate on the problem of navigating through a course, like the 2005 DARPA Challenge, but also challenged the vehicles to obey the laws of driving on public roads, such as keeping in the lane, avoiding

hitting other cars and pedestrians, and following the rules in intersections. The top three self-driving cars finishing the challenge was *Boss* [8], *Junior* [9] and *Odin* [10].
For *Boss*, both offline and real-time perception techniques were used. The offline approach utilized aerial imagery along with predefined GPS waypoints to obtain prior estimates of the road shape. This method involved training *neural network classifiers* (see section 2.4). However, for the real-time road estimation, none of the autonomous vehicles used vision-based techniques. For the vehicle *Odin*, it is explained how vision-based techniques were tested previous to the competition, but that these techniques were scrapped due to lack of robustness. Instead, they landed on only using high-density LIDAR data to detect the road and drivable surfaces. As did the two other autonomous vehicles, *Boss* and *Junior*.
This was an interesting change from the 2005 DARPA Challenge, where most of the contenders used a vision-based approach to improve the perception of the nearby scenery. One factor that may be crucial for the difference in approaches may be the fact that the 2007 DARPA Challenge was conducted at a parking lot, where vision based approaches may yield a lot of ambiguity, while in the 2005 challenge there was a clear road to follow. Another important aspect was the vast improvement in high resolution LIDAR sensors that was developed as a result of the second DARPA grand challenge in 2005. The LIDAR manufacturer *Velodyne* was inspired by the 2005 challenge to improve their LIDAR technology, and before the 2007 challenge, the *HDL-64E* sensor was developed, producing higher resolution LIDAR data [11].

1.2.3 Repercussions of the DARPA Challenges

Since the DARPA Urban challenge, a huge interest in self-driving cars has arisen. The automotive industry has increased their research in advanced driver assisting systems, such as autonomous parking, lane following, etc. In 2009, Google initiated a project aiming to develop a self-driving car. The technology and techniques used by Google builds on much of the research and discoveries which rose from the DARPA challenges, such as high density maps in combination with high resolution LIDAR sensors, cameras and radar sensors, as well as complex software techniques such as machine learning [12].

In this context, it is also worth mentioning a branch of machine learning algorithms which has received a lot of attention in the last decade, namely *Deep learning*. Advances in *graphics processing units* (GPUs) and initiatives from several contributors on making larger datasets for image recognition purposes, like ImageNet [13], has helped to put

a greater focus on the field of machine learning for computer vision purposes. A particular Deep Learning algorithm showing great potential is *Convolutional Neural Nets* (CNN). Although it was originally developed in 1980 by Kunihiko Fukushima [14], it had its modern day revival in 2012 with the CNN model described in [15]. Since then, CNN has been used for many computer vision purposes [16], [17], [18], and is currently the dominating approach in the *KITTI Vision Benchmark* for road/lane detection [1], and in the KITTI Vision Benchmark for object detection [19]. CNN is also currently the leading approach to recognizing handwritten digits from the *MNIST database* [20].
Although CNN is not the focus of this thesis, it is an important part of the modern day state-of-the-art.

1.3 Research Goals

This thesis will focus on a computer vision based approach related to local navigation. More specifically, the primary goal of this thesis is to develop a method for detecting the location of road, perceived from the perspective of a chromatic camera positioned on the roof of the UGV, using computer vision and machine learning methods. The method must produce detection of the road in real-time. The real-time criterion in this context requires that the rate of estimation must match the frame rate of the camera, which for this camera model is adjustable.
The purpose of the algorithm is not to produce a binary segmented image, where each pixel are classified as "road" or "not road", but rather produce a *probability image*, where each pixel contains a probability of being a pixel representing the road. The reason for this approach is based on the assumption that it is more robust to merge this probability image with other soft estimates, as for example those computed from the Stereo Vision, SLAM, or Object Recognition.

The primary goal of this thesis has been to develop and implement a robust computer vision based road estimation algorithm, which is adaptable to variations in road appearance, and simultaneously executable in real-time. The focus has been on finding robust image features, which makes the problem of separating the road from other scenery object easier. The road is continuously modeled as the UGV is traversing the environment, making the estimation adaptable to changing road appearance. The road is modeled as a *Mixture of Gaussian* (MoG) distribution, using a machine learning algorithm known as the *Expectation Maximization* (EM) algorithm. The MoG modeling method through the EM algorithm is chosen for its ability to quickly parameterize complex data distributions, which is an important requirement for the adaptability of the road estimation algorithm.

As an effort to boost the confidence of the classical pixel classification approach, an additional method is described in this thesis. This method exploits the geometrical properties in an image, and seeks to detect the left and right road edge. The algorithm involves detecting the vanishing point of the two road edges, and then finding the lines in the image which leads up to the detected vanishing point, and simultaneously displays typical characteristics of road edges. This method shall serve as an independent algorithm, and the two shall be merged to provide a more robust estimation of the road, exploiting color, textural and geometrical properties of the road images.

1.4 Thesis Structure

This thesis is divided into five chapters: Introduction, background theory, implementation, experiments, and discussion.

Chapter 2 gives an introduction to the theory and related work used in the thesis work.
Chapter 3 contains an overview of the implementation of the different methods used for extracting features, modeling the road, and estimation of the road. In addition, this chapter will present how the different methods are optimized to meet the real-time criterion.
Chapter 4 describes three experiments conducted to investigate different aspects of the road estimation algorithm, and one larger experiment where the performance of the algorithm is evaluated.
Chapter 5 presents an overall discussion of the experiment results. Some suggestions for future work are discussed, and lastly, a final conclusive summary of the thesis is presented.

Chapter 2

Background Theory

Estimation of the road from images require three components. The first is a *model* of the road describing the parameters θ of the road in terms of the chosen feature space. The second is a *learning algorithm* which allows θ to be estimated from a set of training samples. The third component is a probability function which estimates the new pixel's probability of being a road pixel.

This chapter attempts to give an overview of the field of computer vision techniques applicable to this problem. Some of the most promising modeling techniques for the road estimation problem will be discussed, as well as some common image features which presumably will make the estimation of the road easier, including color, textural and geometrical features. Some different methods for extracting such features will be discussed. Lastly, a brief description of some machine learning algorithms for computer vision purposes are discussed, with emphasis on machine learning algorithms for fast parameter estimation.

2.1 Outdoor Imaging

Images are created by electronic sensors capturing light which is emitted from some source and then reflected off the surface of an object. Different object surfaces have different reflective properties which, when captured by an image sensor, creates different colors and intensities. An illustration of this shown in figure 2.1. For the purpose of detecting road, both color and intensity may serve as an important clue.

In outdoor scenes at daytime, the sun is the main source of illumination. However, the light reflected from objects are not the only electromagnetic radiation involved in the process of imaging. The sun, although often considered as a point source, is continuously emitting light all over the planet, causing photons to bounce all around. For imaging, this causes a lot of scattered light to enter the lens. These are photons reflected from

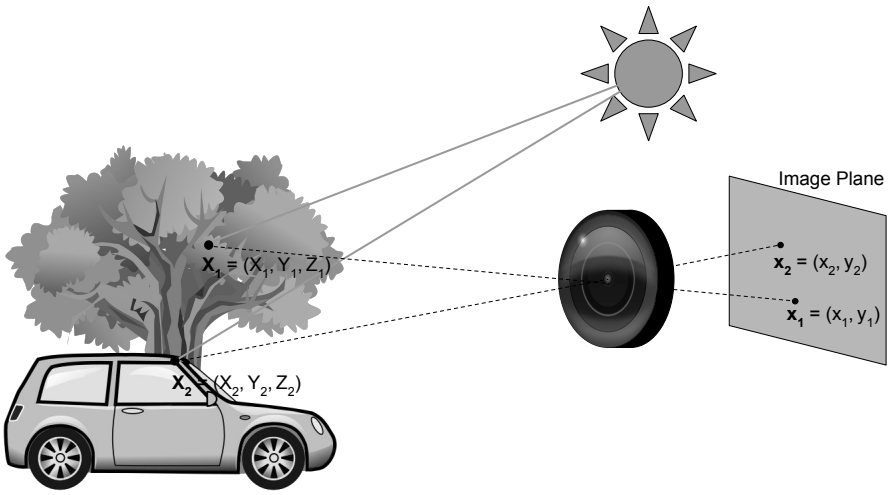


Figure 2.1: Illustration of how light is emitted from the sun, reflected from the surface of an object, and at last recorded as an intensity value at a specific pixel on the image sensor. The light reflected at a point $\mathbf{X}_i$ in the 3D real world is captured in the image plane at pixel $\mathbf{x}_i$.

object to object, losing some energy on the way, but finally making its way onto the image sensor. Such scatter light enables both imaging sensors and humans to see objects and elements that are not necessarily directly lit up by the sun, also known as shadowed areas, which aid to increasing the complexity of the scenery, in terms of light. Figure 2.2 presents an illustration of the many sources of electromagnetic radiation present in an outdoor scene.

2.1.1 Illumination Invariant Images

Shadows and illumination variations cause severe problems for color based scene analysis. It distorts the actual colors of the objects, and can confuse an algorithm in many ways. These variations are caused by various light sources illuminating different parts of the objects.
In daylight, when the sky is clear, the dominant illumination source is the sun. However, there are also plenty of indirect light sources present. As mentioned, this light originates from sunlight which is reflecting off different objects, like clouds. This causes shadowed areas to reflect light onto the image sensor with a different intensity and frequency than those reflected from direct sunlight.

Methods which deals with the problem of illumination variations has been developed [21], [22], [23]. This thesis adopts the method presented in [23] for the purpose of minimizing the effect of shadows in road images.

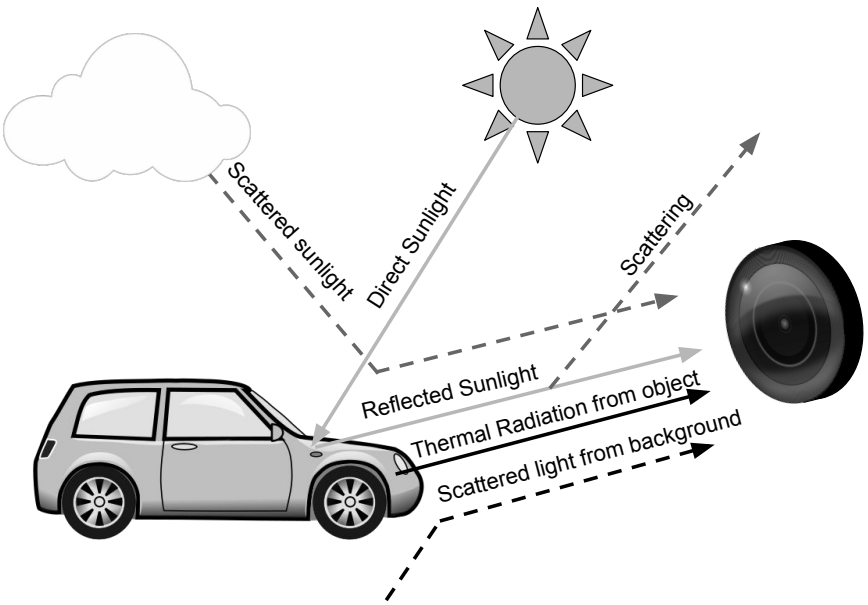


Figure 2.2: Illustration of the many sources of electromagnetic radiation that enters the camera lens.

How an image sensor R with spectral sensitivity $F(\lambda)$ behaves with respect to an illumination source with emitted spectral power distribution $E(\lambda)$ incident on an object with surface reflectivity $S(\lambda)$, is described by the following equation:

$$R^{x,E} = \mathbf{a}^x \cdot \mathbf{n}^x I^x \int S^x(\lambda) E^x(\lambda) F(\lambda) d\lambda \tag{2.1}$$

where the unit vectors $\mathbf{a}^x$ is the direction of the sunlight, $\mathbf{n}^x$ is the direction of the surface normal and I^x represents the intensity of the illumination on point x.
This relationship can be used to create an intensity image $\mathcal{I}$, which emphasizes the effect of the reflectivity of the material surface $S^x(\lambda)$, and minimizes the effect of the illumination source spectrum $E^x(\lambda)$ and intensity I^x.
As in [21], it is assumed that $F(\lambda)$ can be modelled as a Dirac delta function centered on wavelength λ_i, which leads to the response function:

$$R^{x,E} = \mathbf{a}^x \cdot \mathbf{n}^x I^x S^x(\lambda_i) E^x(\lambda_i) \tag{2.2}$$

For road images, [23] describes, with certain assumptions and simplifications, how Eq. (2.2) can be rewritten as a function of the 3 color channels of an image (R, G and B) to produce an illumination invariant image $\mathcal{I}$:

$$\mathcal{I} = log(R_2) - \alpha log(R_1) - (1 - \alpha) log(R_3) \tag{2.3}$$

where α is a scaling constant based on the peak spectral response of the color channels. α is found by solving the following equation with respect to alpha:

$$\frac{1}{\lambda_2} = \frac{\alpha}{\lambda_1} + \frac{1-\alpha}{\lambda_3} \tag{2.4}$$

where λ_1, λ_2 and λ_3 are the peak spectral response of the color channels *Blue*, *Green* and *Red*, respectively.

The result of Eq. 2.3, calculated from the color image I_{RGB}, is shown in figure 2.3b. As shown, the shadows in the gray-level intensity image I_G, shown in figure 2.3a, have almost been removed in the illumination invariant image $\mathcal{I}$.

(a)

(b)

Figure 2.3: Example of the different amount of shades contained in **(a)** the gray-level intensity image I_g, and **(b)** the illumination invariant image $\mathcal{I}$, both computed on an image from the *KITTI dataset for road segmentation* [1].

Several applications has shown to benefit from exploiting the illumination invariant images [22], [24]. For road estimation, this thesis will discuss the benefits and drawbacks of using the illumination invariant image to extract features.

2.2 Image Features

Choosing the right features is an important task in most practical computer vision application. There is a large assortment of measures and properties that can be extracted from an image. The image could be used directly as a feature, or a pre-processing step could be applied, which transforms the image into some new space of variables, where the problem hopefully will be easier to solve. This pre-processing stage is commonly known as *feature extraction* [25].

The problem of detecting road from images is complex. There are seemingly infinite variations of road texture, color, illumination, and shadows, so the choice of features should be carefully made according to these challenges. The goal should be to find features that are as invariant as possible to these variations while simultaneously emphasizing the characteristics of the road. It should also be kept in mind the real-time requirement of the application.

For clarity, in the equations, figures and examples used in the following section, the same symbolic names will be employed for the different image types. These names are listed in Table 2.1. An arbitrary pixel will be referred to with the (x, y) notation, where x is the horizontal image coordinate, and y is the vertical coordinate.

I_{RGB}	3-channel color image, using the RGB color-space
$\mathcal{I}$	Illumination Invariant Image
I_{G}	Gray-level intensity image

Table 2.1: Overview of symbolic names used for the different image types throughout the thesis.

2.2.1 Color Features

For pixel classification, color is one of the most obvious choices of features. For humans, colors and intensity variations are one of the main vision based sources of information that enable discriminate between objects. In image processing, there are several ways of representing color. The most common representation is through the RGB color-space, where all colors are represented as a combination of the three primary colors *Red*, *Green* and *Blue*. Different color intensities are expressed by adding intensity values to each of the three colors. Commonly, this value is within the range [0, 255].
For classification purposes, the three color channels could be used directly as features. In a color image, a single pixel I_{RGB}(x, y), can therefore be

expressed as a 3-dimensional feature vector, containing three intensity values for each of the colors Red, Green, and Blue.

Although the variation is great, the color of road is usually some shade of gray. Regarding classification, the color makes the road stand out in cases where it is surrounded by elements containing colors different from gray, like grass, trees, corn fields, and similar. However, in areas containing similar colors, as in cities, the color-based approach shows its weakness. Additionally, the color-based approach fails in the face of shadows covering the road surface, since the color of the shadowed road is so different from the rest of the road.

2.2.2 Textural Features

Another way of segmenting image elements from one another is to look at the *texture* of the elements. For humans, texture helps to distinguish between objects of similar color and intensity. Although the concept of texture is commonly understood, there is no precise definition of it. Wilson [26] describes texture as "Spatially extended patterns with more or less accurate repetitions of some basic structure element, called *texels*. Texture analysis can, therefore, be seen as the concept of segmenting objects which are uniform with respect texture. The size and shape of these texels are arbitrary, and, in order to differentiate between the different texels in a single image, local window operations are necessary. The choice of window size is highly problem dependent and requires a prior analysis of how texels of the relevant object varies in terms of size, shape, orientation, contrast, etc. For road estimation, it is important to be aware that drivable road can contain many variations of texture, so a single assumption about the road texture is rarely valid for all situations. However, a general observation is that the road texels are small, and occupies rarely more than a few pixels (this depending on the distance from the road to the camera, and the type of camera used). This property can help to separate the road from objects such as trees, buildings, cars, and people, where the textural patterns usually occupy larger regions in the image.

Gray-Level Co-Occurrence Matrix

There are several approaches to texture analysis. One of the most popular involves the use of the *Gray-Level Co-Occurrence Matrix* (GLCM) to gather statistical information about an image, or a local image region [27]. As the name suggests, the GLCM is usually calculated from gray-level intensity images, gathering information based solely on the pixel intensity. The GLCM describes an image texture by calculating how often pairs of pixel intensities, with a specific spatial relationship, occurs.

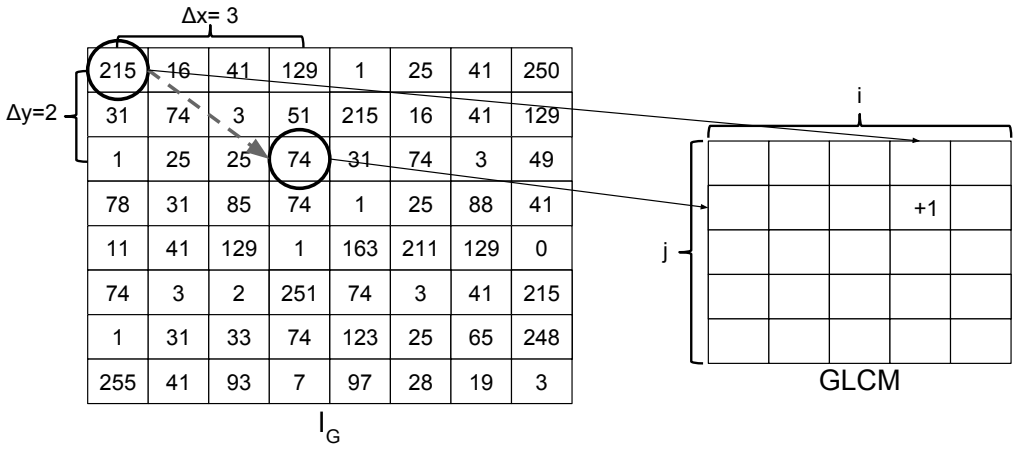


Figure 2.4: Illustration of one individual computational step of the GLCM calculation. In this example, the comparison offset (Δx, Δy) is set to (3, 2), and the intensities are normalized to the number of GLCM levels, which is set to 5. Note that this figure only represents one arbitrary co-occurrence observations. The same procedure is applied for every pixel in I_G.

To compute the GLCM from the gray-level intensity image I_G, each pixel in I_G(x,y) must be compared to the pixel I_G(x+Δx,y + Δy), where (Δx, Δy) is a specified offset which dictates the horizontal and vertical distance of the comparison. In the GLCM, the observation of the pixel intensities I_G(x,y) and I_G(x+Δx,y+Δy) counts as 1 co-occurrence. Mathematically, the Gray-Level Co-Occurrence matrix P is defined as:

$$P(i,j|\Delta x,\Delta y) = \sum_{x=1}^{N-\Delta x}\sum_{x=1}^{M-\Delta y}\begin{cases}1 & \text{if } I_G(x,y)=i \text{ and } I_G(x+\Delta x,y+\Delta y)=j\\ 0 & \text{otherwise}\end{cases} \tag{2.5}$$

where N×M is the size of I_G.
To avoid a large GLCM, the range of the gray-level intensity is normalized to the range $[0, G-1]$, where G is the specified size of the GLCM. The size G is commonly known as *GLCM levels*, and influences the possible resolution of the statistical measures extracted from the GLCM, but also the computational complexity of calculating these measures.

In figure 2.4, an example of a single computational step of a GLCM calculation is illustrated. The two pixels marked by the circles, containing the gray level intensities I_G(x,y) = 215 and I_G(x+Δx,y+Δy) = 74, are compared. Normalized from the range $[0, 255]$ to the range $[0, 4]$, these values translates to the GLCM coordinate:

$$P\left(\frac{I_G(x,u)}{255}\times(G-1), \frac{I_G(x+\Delta x,y+\Delta y)}{255}\times(G-1)\right)$$

$$= P\left(\frac{215}{255} \times 4, \frac{74}{255} \times 4\right) = P(3,1)$$

Furthermore, the observed co-occurrence of the intensity values 215 and 74 causes the value of $P(3,1)$ to increase by one. The same procedure is applied to all other pixels, I_G(x,y), and the final result is a GLCM containing the number of observed co-occurrences for every possible co-occurrence in the range $[0,4]$. Normally, the choice of GLCM levels is higher. The number 5 in this example is chosen for illustration purposes.

In practical applications, like the road estimation algorithm, it is rarely convenient to compute the GLCM for the entire image, since this only yields an overall statistical measure of gray-level intensity change. Instead, the GLCM should be used as a tool to distinguish between the different objects in a single image that most likely displays different textural properties. As described by Peckinpaugh, the GLCM can be used to compute such textural properties for local neighborhoods, or windows, throughout the gray-level intensity image I_G [28]. Figure 2.5 shows an example of the GLCM calculated for a local window before a statistical measure is extracted from it. Notice that the coordinate of the center pixel in the window is the coordinate assigned to the feature value in the feature image.

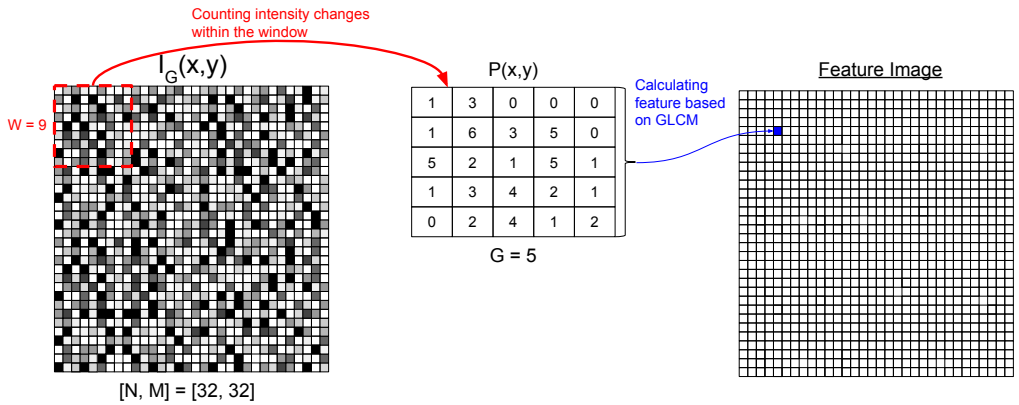


Figure 2.5: Illustration of an arbitrary feature computed by calculating the GLCM from a local neighborhood.

Several statistical measures can be calculated from the GLCM. For the purpose of detecting road, it is appropriate to choose measures which will yield different results for road patterns compared to the pattern of other objects. This thesis has chosen to focus on the features: *Variance*, *Angular Second Moment*, and *Inertia*.

$$VAR = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i-\mu)^2 P(i,j) \tag{2.6}$$

Eq. (2.6) is the equation for *GLCM Variance*, a measure which weights higher the elements in the GLCM $P(i,j)$ that differs from the average μ. Road has a pretty constant pattern, which means it will typically yield low variance measures, while in trees or bushes, the patterns are usually more random, consequently yielding a higher variance measure. An example of the results of computing the variance feature image from $\mathcal{I}$ is shown in figure 2.6.

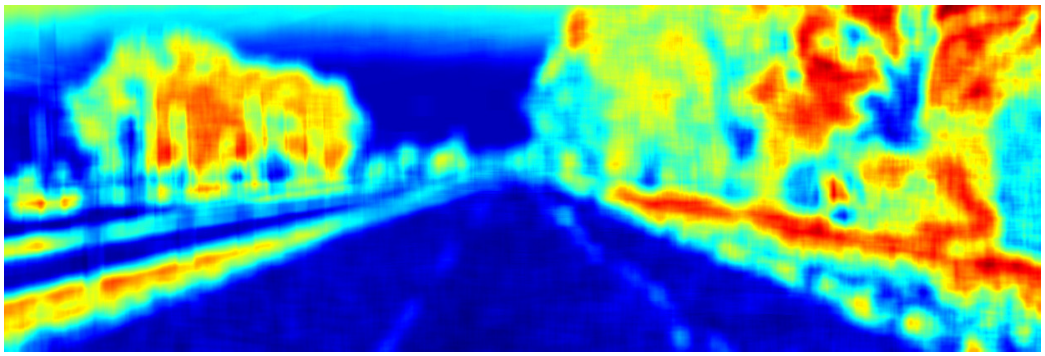


Figure 2.6: Example of the results of computing the variance feature image.

$$ASM = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \big(P(i,j)\big)^2 \tag{2.7}$$

Angular second moment, in Eq. (2.7), is a measure of the number of gray levels present in the neighborhood. Few gray levels will result in a GLCM with few non-zero elements. However, these few non-zero elements will have a high count, thus yielding a higher Angular second moment value. This will typically occur in neighborhoods where the pattern is highly uniform. An example of the angular second moment feature image computed from $\mathcal{I}$ is show in figure 2.7.

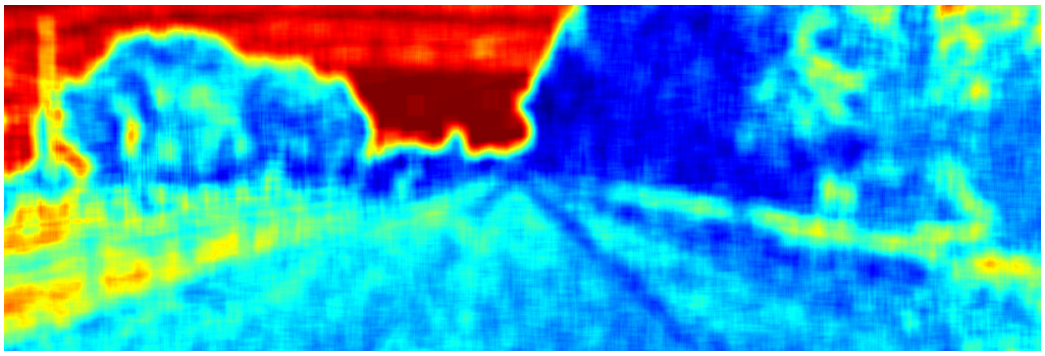


Figure 2.7: Example of the results of computing the angular second moment feature image.

$$INR = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i-j)^2 P(i,j) \tag{2.8}$$

Inertia, in Eq. (2.8), emphasizes the GLCM values away from the matrix diagonal. A high inertia value corresponds to a neighborhood where there is a high degree of pairwise variation, when comparing p_i to p_j. This occurs in neighborhoods with a lot of contrast, i.e. transitions from low to high gray level values, and conversely.

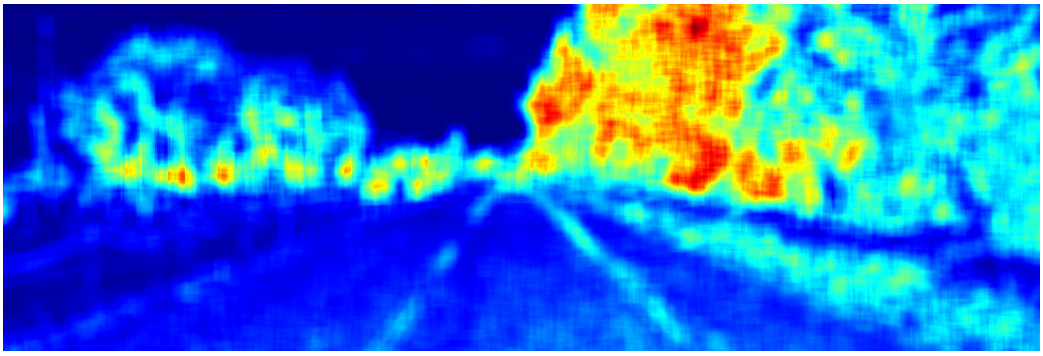


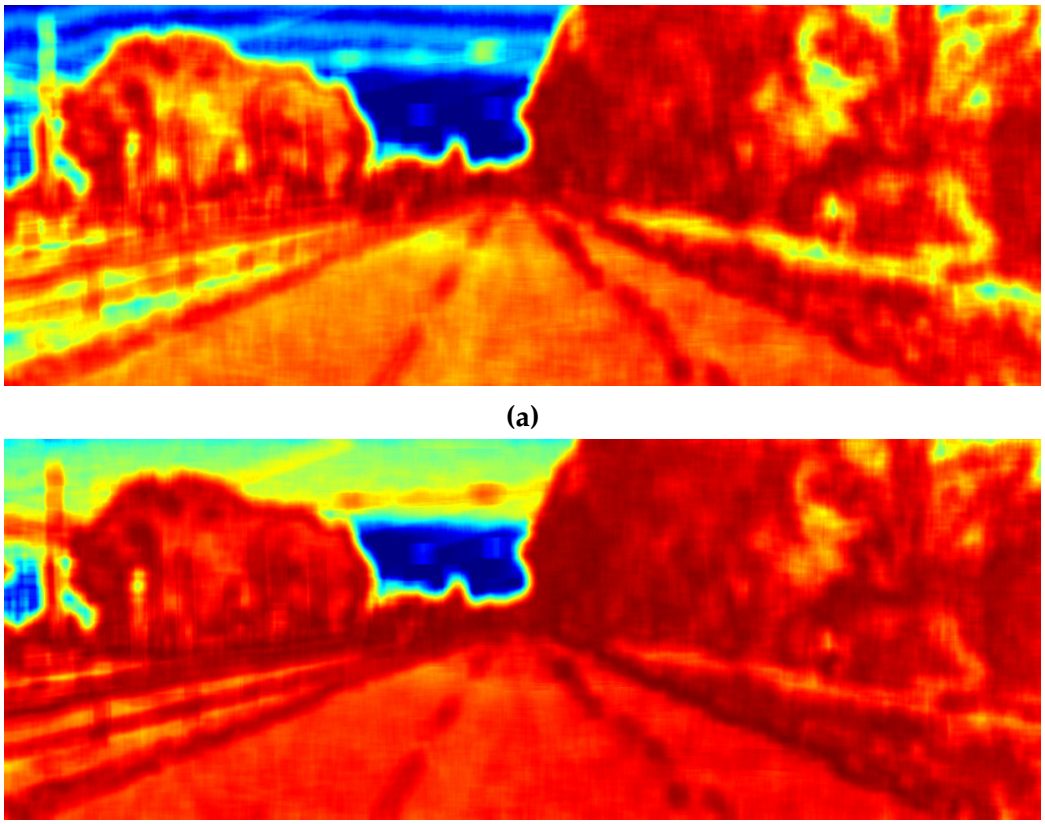
Figure 2.8: Example of the results of computing the inertia feature image.

The choice of window size, comparison distance and angle of comparison is important when working with pattern recognition. This will dictate both the quality of the statistical texture measures, the sensitivity to objects with certain spatial repetition of the patterns, and the sensitivity to objects with a strong orientation.

Texture from Histogram

A somewhat simpler method of extracting textural descriptions from an image is to use an *intensity histogram*. Intensity histograms are a histogram containing the number of occurrences of the different gray-level intensities within the image neighborhood [29].
Like the GLCM, the intensity histogram can be used to provide a statistical description of the neighborhood. However, unlike the GLCM, the intensity histogram only counts the occurrences of the gray-level intensities and not the pairwise co-occurrence of two individual intensities. Although this results in the loss of the spatial aspect of the statistical information, the intensity histogram still contains information which enables the approximation of many of the same features found from the GLCM. Furthermore, the computational complexity of estimating the textural features decreases greatly.

An example of such a feature is the *Entropy*, which is given by Eq. (2.9). Entropy is a measure of the degree of disorder in the neighborhood. Eq.



(a)

(b)

Figure 2.9: Example of the similarity of computing the entropy feature image from **(a)** the GLCM, and **(b)** the histogram, from the illumination invariant image $\mathcal{I}$

(2.9) yields high values if all histogram counts are equal, i.e. if there are several distinct intensity values present in the neighborhood. Entropy is the opposite measure of the angular second moment, which was given by Eq. (2.7). An example of the different results between computing the entropy measure from the GLCM and from the intensity histogram is shown in figure 2.9. Entropy is defined by:

$$ENT = -\sum_{i=0}^{G-1} P(i) log_2\{P(i)\} \tag{2.9}$$

2.2.3 Local Standard Deviation

Some image features can be extracted without having to depend on the GLCM, intensity histograms or other statistical neighborhood operations. More specifically, some features can be calculated very quickly by approximating certain statistical measures. One of these is the *local standard deviation* feature, which is a measure of how much a neighborhood devi-

ates from the mean of the neighborhood. Local standard deviation can be viewed as a measure of local energy.
Eq. (2.10) is the basic expression for standard deviation.

$$\sigma = \sqrt{E[(X - \mu)^2]} \tag{2.10}$$

where E is the expected value of the variable X, and μ is the mean value of X. Eq. (2.10) could be reformulated as a function only dependent on the expectation value.

$$\sigma = \sqrt{E[X^2] - (E[X])^2} \tag{2.11}$$

Eq. (2.11) is known as the *computational formula for variance*.
Utilizing Eq. (2.11) in image processing, requires a method for computing the expectation of a neighborhood surrounding each pixel in the image. A sliding window is an obvious choice, but estimating the expectation for each of window over the whole image, is a computationally exhausting exercise. However, the expected value can be approximated by the use of a blurring function. Image blurring involves convolving an image with a filter kernel. This exercise translates to replacing each pixel in the image with the average of its neighborhood, weighted by the values contained in the filter kernel.

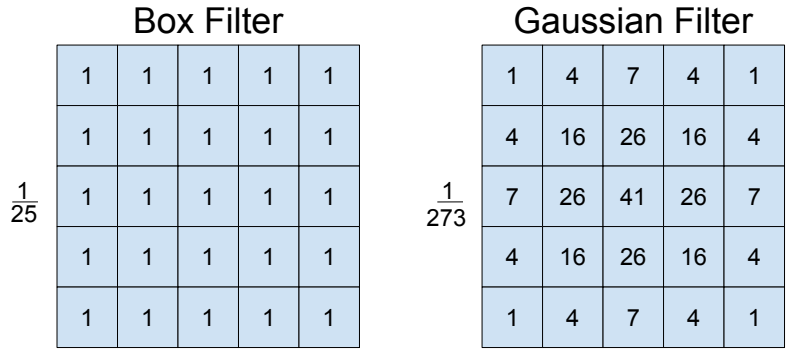


Figure 2.10: Example of two filter kernels commonly used for image blurring, i.e. the Box filter (left) and the Gaussian filter (right).

There are different types of filter kernels used for image blurring, each weighting a local pixel neighborhood in a different way. Two kernels which are frequently used, are the *Box filter* and the *Gaussian filter*, represented in figure 2.10. The Gaussian filter is an important blurring filter which weights the neighborhood based on the closeness to the center pixel. Gaussian blurring is often favorable for noise reduction, and usually produce a "prettier" image than when a box filter is used. Still, for the purpose of approximating a local mean, the box kernel is more appropriate.

Blurring $\mathcal{I}$ with a box filter can, therefore, be used as an approximation of the expectation $E[X]$, as used in Eq. (2.11), of local neighborhoods throughout $\mathcal{I}$. Similarly, $E[X^2]$ is described by blurring the quadratic image, i.e. the image multiplied with itself. The local standard deviation can be calculated very quickly with only the help of a blurring computation, and a few computational steps which are easily parallelized.

2.3 Classification, Modeling and Discrimination

Given that the objects in an image belong to different classes, these objects are viewed as having different *models*. A model is a parametrization of a class, described in terms of features.
Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification, which strides to estimate the probability that an observation belongs to a specific class. This probability is based on the probability of the class $\mathrm{P}(\omega_i)$ (or *prior probability*), and the *class-conditional probability density* function $\mathrm{p}(x|\omega_i)$, which can be considered as an observations likelihood of belonging to the class ω_i.

In real world problems, perfect descriptions of these distribution functions are rarely available. In fact, these functions are often unknown, and to solve this issue, the unknown functions must be estimated from a set of samples, through a training procedure. Such a training procedure usually involves fitting a model to the training samples through some optimization scheme. Two popular methods approaches to this, is the *Maximum likelihood* and the *Bayesian approach* [30].
However, to perform a parameter estimation of these functions, prior knowledge about the distribution functions must be utilized. In probability theory, there are as many probability distributions as there are problems, and they are divided into two main categories based on whether the variable $\mathbf{x}$ is discrete or continuous. *Discrete uniform distribution* and *Binomial distribution* are examples of two common discrete distributions, while the *Gaussian/Normal distribution* is an example of a very common continuous distribution. A collection of common probability distributions can be found in [31]

2.3.1 Normal/Gaussian Distribution

In many image analysis problems, the probability distribution is unknown. In these types of problems, the Normal/Gaussian distribution is often used. The Normal/Gaussian distribution is based on the *Central Limit Theorem* [32], which states that the average of large amounts of random variables will converge towards a normal distribution. The distribu-

tion becomes increasingly normal as the number of variables increase. The normal distributions popularity is due to its apparent analytical simplicity, and because it is easily specified by only the two parameters *expected value*, or mean, and the *expected squared deviation*, or variance.

In the *univariate* case, i.e. the case of a single continuous random variable x, the Gaussian density is expressed as in Eq. (2.12).

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\left\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \right\} \tag{2.12}$$

For the *multivariate* case, i.e. the case of a D-dimensional vector $\bar{x}$, the multivariate Gaussian density is expressed as in Eq. (2.13).

$$f(\bar{x}|\bar{\mu},\Sigma) = \frac{1}{2\pi^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{ -\frac{1}{2}(\bar{x}-\bar{\mu})^T\Sigma^{-1}(\bar{x}-\bar{\mu}) \right\} \tag{2.13}$$

where $\bar{\mu}$ now is a D-dimensional mean vector (for each of the D features), Σ is a D$\times$D covariance matrix, and $|\Sigma|$ denotes the determinant of Σ.
An example of a univariate Gaussian density is illustrated in figure 2.11. An example of a multivariate Gaussian density, specifically a bivariate density, is illustrated in figure 2.12b.

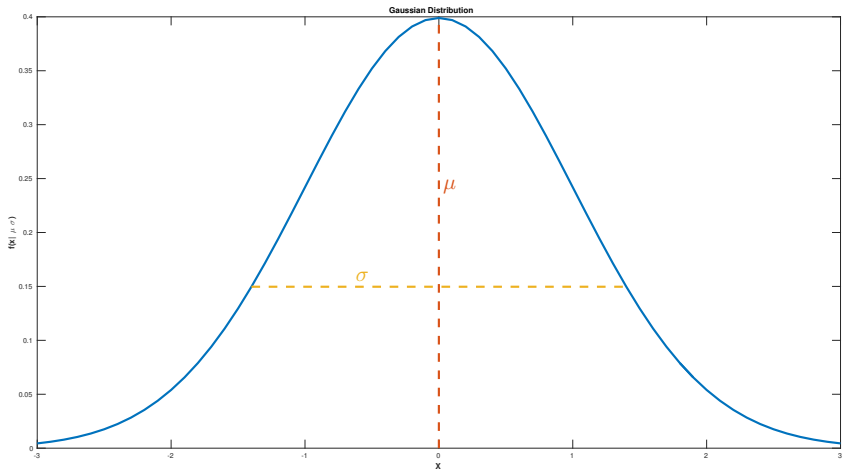
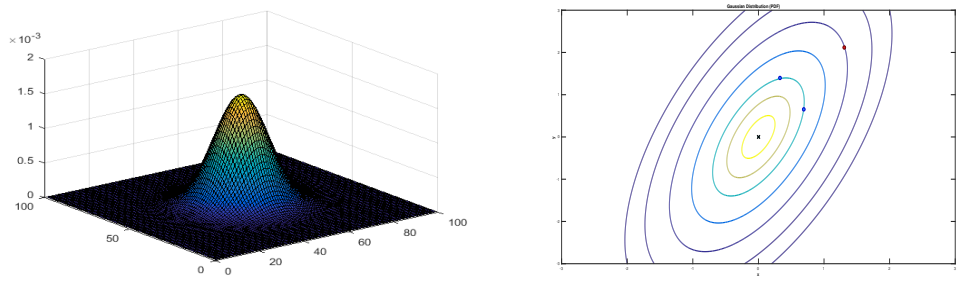


Figure 2.11: Example of a univariate Gaussian density. The mean and variance of the Gaussian is marked by the two dashed lines.

Distance from the Gaussian distribution

The objective of generalizing data into models is to obtain a more manageable object to which new data can be compared. These models could,



(a) **(b)**

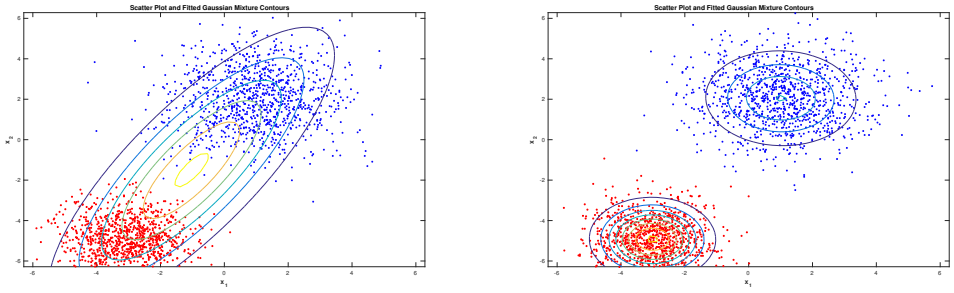
Figure 2.12: The two figures show examples of bivariate Gaussian densities. In **(a)**, the Gaussian is viewed in 3D, where the density is represented as a value on the vertical axis. In **(b)** the Gaussian is viewed in 2D to illustrate how it can be used as a distance measure. The colored contours represent specific distances of the Mahalanobis distance. All points on such a contour will have an equal distance to the mean, as illustrated by the points plotted.

for instance, be used to investigate the similarity of one set of data to another by comparing their models. In pixel classification, it is appropriate to compare a model to each new data sample that occurs. The comparison is performed by comparing the data sample to the parameters of the model through some cost/distance function.
For the Gaussian distribution, if the mean and the variance of the distribution is known, the parameters can be used to compute a distance measure for new, unobserved data. This method uses the probability density function, given by Eq. (2.13), with the known mean and covariance to compute the distance from the given input variable x to the mean, in the feature space. This distance is dependent on the difference between x and μ, and the "shape" of the class distribution, which is dictated by the covariance.
The concept is illustrated in figure 2.12b. In this example, a set of training data has been used to model a bivariate Gaussian density based on the mean $\bar{\mu}$ (marked as a blue star in the figure) and the covariance Σ of the distribution. The contours (ellipsoidal lines) in the figure represent the discrete, imaginary distances to the mean of the Gaussian. Points on these contours will be equally distanced from the mean, which is illustrated in figure 2.12b by the three points. The two blue points is equally distanced from $\bar{\mu}$ while the red point is further away. When the probability density functioned is used as a distance measure, the distance will not be linear throughout the feature space, contrary to Euclidean distance, which is why the contours in the illustration have an ellipsoidal shape.

These principles are the same with higher dimensionality, but illustrations of higher-dimensional distributions are less intuitive, which is why

this thesis will only use 1-dimensional and 2-dimensional distributions as examples.

Mixture of Gaussians



(a) **(b)**

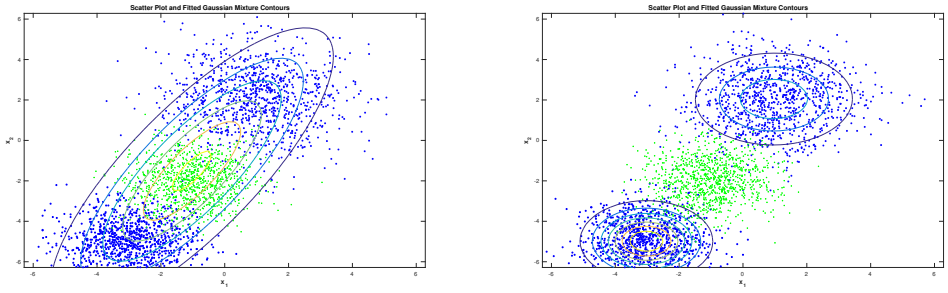
Figure 2.13: Example of a complex data distribution. In **(a)**, the data is modelled as a single Gaussian, while in **(b)** the data is modelled as a mixture of two Gaussians.

Although the Gaussian distribution is a helpful assumption for modeling real world datasets, it has its limitations. Real world data is often messy, and the assumption that a distribution could be best described as a single Gaussian is often inaccurate. As an example, inspect the distributions presented in figure 2.13.

In figure 2.13a, the data is represented with respect to two arbitrary features. The data is grouped into two clusters, with a slight random deviation for each sample. Often in classification problems, distributions like these originate from two different class-conditional distributions. However, this could very well be the characteristics of a single source. Although the purpose of extracting features from raw measurements is to simplify the problem, the result is not necessarily simple. As for this example, where the data is grouped into two clusters, modeling this data as a single Gaussian does not sufficiently describe the characteristics of the distribution. The Gaussian density is centered at a mean somewhere between the two clusters, and the covariance is fairly emphasized in the direction between the two classes.

A way to improve the description of the distribution is to represent the data as a superposition of several Gaussians, called *mixture of Gaussians* (MoG). This way of describing distributions is more versatile to complex data distributions, as illustrated in figure 2.13b, where two Gaussians have been utilized to describe the distribution. As for the single Gaussian case, this model contains the parameters $\bar{\mu}$ and Σ for each Gaussian, but now

there is one set of parameters for each of the K Gaussians, i.e. $\bar{\mu}_k$ and Σ_k. Additionally, to describe the influence each Gaussian has on the model, each Gaussian is assigned a weight, ω_k, which corresponds to the portion of the dataset used to parameterize that particular Gaussian. An example of a situation where the assumption of a single Gaussian distribution leads to inaccuracies, and how a MoG distribution assumption improves upon this, is illustrated in figure 2.14a and 2.14b.



(a) **(b)**

Figure 2.14: Illustration of the advantage of using mixture of Gaussian modeling. In **(a)**, the data marked in blue is modeled as a single Gaussian, while in **(b)** the data marked in blue is modeled as a mixture of two Gaussians. The ellipsoidal contours represents imaginary, discrete distances from the mean of the Gaussians.

In figure 2.14, two different class conditional distributions are represented in the same feature space. The data marked in blue serves as training samples for estimating the parameters of one of the distributions. In figure 2.14a, these samples are modeled as a single Gaussian. The mean of this Gaussian lies somewhere between the two blue clusters. The problem becomes evident when a second distribution is introduced, marked in green in the figure, which belongs to another class. Although this distribution is pretty well separated from the two clusters of the first class, the single Gaussian does not describe the distribution well enough to discriminate between the two. In fact, the distribution belonging to the second class is closer to the mean of the estimated Gaussian from the first class, which proves a problem if the two is to be discriminated. Figure 2.14b illustrates how a MoG distribution resolves this issue by modeling the distribution as two individual Gaussians. In the illustration, the second distribution is well separated from the parameterized model of the first.

As well as for the single Gaussian case, the MoG model can be used to compute a distance measure between the model and new, unobserved data. Section 2.3.1 described how Eq. (2.13) along with known means and

covariance could be used to compute the distance between a new sample and the model mean. These equations can be translated to apply for the MoG model as well. The distance measure of a new data sample x is estimated across all K Gaussian densities, introducing the weight ω_k of each density.

$$f(\bar{x}|\bar{\omega},\bar{\mu},\Sigma) = \sum_{k=0}^{K-1} \bar{\omega}_k \frac{1}{2\pi^{D/2}|\Sigma_k|^{1/2}} exp\bigg(-\frac{1}{2}(\bar{x}-\bar{\mu}_k)^T \Sigma_k^{-1}(\bar{x}-\bar{\mu}_k)\bigg) \tag{2.14}$$

Equation (2.14) weights the dominating Gaussian density highest, and consequently, the highest ranked density in the mixture contributes the most to the distance measure. Using the multivariate probability density function for a MoG model as a distance measure leads to a rather complex distance measure in the feature space. A single sample will be drawn to the different means of the individual densities, with respect to their variance and weight. An example of this complex distance measure is illustrated in figure 2.15.

As mentioned, in order to use this method to estimate the distance from sample $\bar{x}$ to the model mean, the parameters $\bar{\omega}$, $\bar{\mu}$ and Σ must be estimated for each Gaussian beforehand. This task involves finding the combination of Gaussian densities that best represents the distribution. For this purpose, a popular method for obtaining this combination is the *Expectation Maximization algorithm* [33].

2.4 Machine Learning for Model Estimation

In most image analysis problems that concerns recognition of patterns or specific objects, it is almost impossible to guess the optimal classification model beforehand. This is where *machine learning* is deployed. Machine learning in this sense involves initializing some general model, and then using training data, the unknown parameters of the model is learned or estimated. There are many forms of machine learning which is applicable, and there has been written several books addressing the subject [25], [30], [31]. Again, depending on the problem at hand, different machine learning approaches may be favorable. In general, in terms of the learning process, there are three main categories; *Supervised learning*, *Unsupervised learning* and *Reinforced learning*. Supervised learning involves presenting the learning algorithm with both the data and the ground truth labels for the data. In unsupervised learning, no information about the data is presented. The algorithm is left on its own to make sense of the data. In reinforced learning, the learning algorithm is given feedback on how well the model performs with respect to some goal or performance measure,

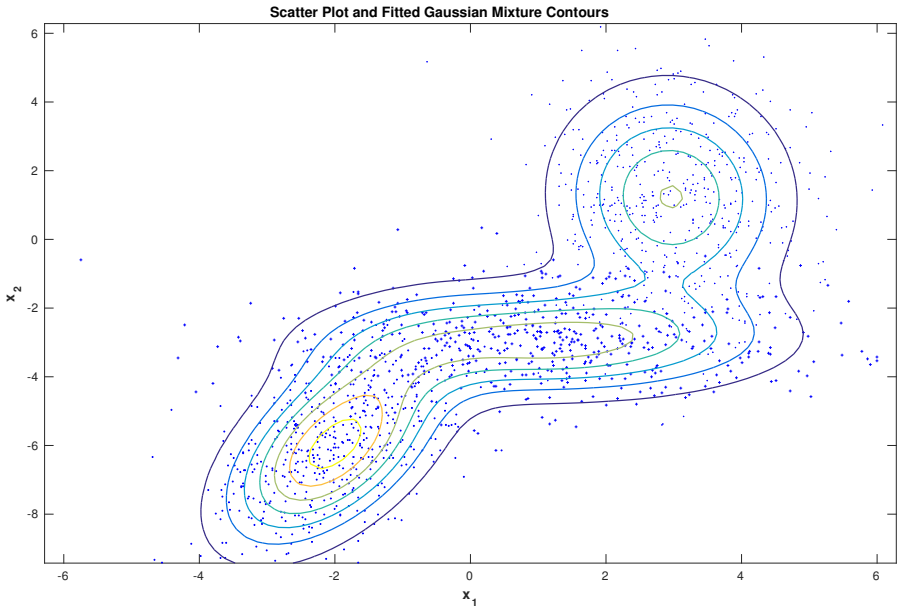


Figure 2.15: Visualization of the usage of Eq. (2.14) to compute a complex distance measure, with respect to a MoG model computer with 3 Gaussians.

which is then given as input to the next iteration. Approaches to machine learning includes *Decision trees*, *Support vector machines*, *Clustering*, *Genetic algorithms* and more. A machine learning approach which has gotten a lot of attention in the last decade is *Deep learning*, which is a result of the large advances in graphics processing units (GPU) [34]. Deep learning is a collective term for machine learning algorithms which uses multiple layers of complex data structures to model the data. In image analysis, *Convolutional neural nets* (CNN) is a specific deep learning approach which has become a popular method for object recognition purposes. An example of CNN used for image analysis purposes is presented in [35], where a computational model for face recognition is developed using a CNN.

However, CNN and many other machine learning algorithms require time-consuming training procedures accompanied by huge amounts of labeled training data to achieve adequate results. These kinds of learning procedures are referred to as *offline training*, where there is no time limit to when the learning has to be finished. Yet, for certain problems, there may not be enough data to perform such an offline learning scheme, or perhaps the classes change over time. In these situations, a system which is adaptable to such changes is desirable. Furthermore, a machine learning algorithm which is adaptable to change requires *online learning*. Online learning describes a learning scheme where learning goes on as the data are collected. For classification purposes, this enables alternating execution of learning and classification.

2.4.1 Expectation Maximization

The *Expectation-Maximization* (EM) algorithm is an algorithm which seeks to maximize the likelihood of parameters in a statistical model [33]. In this thesis, the EM algorithm will be discussed in the context of maximizing the likelihood of the parameters of a MoG model. However, it should be emphasized that it has broad applicability, and can be applied to a variety of statistical models [25].

As discussed in section 2.3.1, a MoG model is parameterized by the means $\bar{\mu}_k$, covariance Σ_k, and weights $\bar{\omega}_k$ of each of the K Gaussians, denoted as $\theta = (\bar{\omega}_k, \bar{\mu}_k, \Sigma_k)$. Given a set of N training samples, the EM algorithm seeks to establish the value for θ which optimizes the description of these samples. The algorithm starts of by an initial guess for the parameters θ. Then, the EM algorithm proceeds by an iterative process, which involves alternating between an *expectation-step* (E-step), and a *maximization-step* (M-step).

The E-step calculates the posterior probability of each sample $\bar{x}_i$ belonging to each of the K Gaussians, given the current estimate for the parameters θ. These posterior probabilities are often referred to as *responsibilities*, and are given by equation (2.15):

$$p(k|\bar{x}_i) = \frac{\omega_k f(\bar{x}_i|\bar{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \omega_j f(\bar{x}_i|\bar{\mu}_j, \Sigma_j)} \tag{2.15}$$

where $f(\bar{x}_i|\bar{\mu}_k, \Sigma_k)$ is the multivariate Gaussian density function of a single Gaussian within a MoG model (Eq. (2.14))

After the responsibility $p(k|\bar{x}_i)$ has been computed for all samples, the algorithm proceeds with the M-step, where θ is re-estimated using the current responsibilities:

$$\bar{\mu}_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} p(k|\bar{x}_i)\bar{x}_i \tag{2.16}$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{N} p(k|\bar{x}_i)(\bar{x}_i - \bar{\mu}_k^{new})(\bar{x}_i - \bar{\mu}_k^{new})^T \tag{2.17}$$

$$\omega_k^{new} = \frac{N_k}{N} \tag{2.18}$$

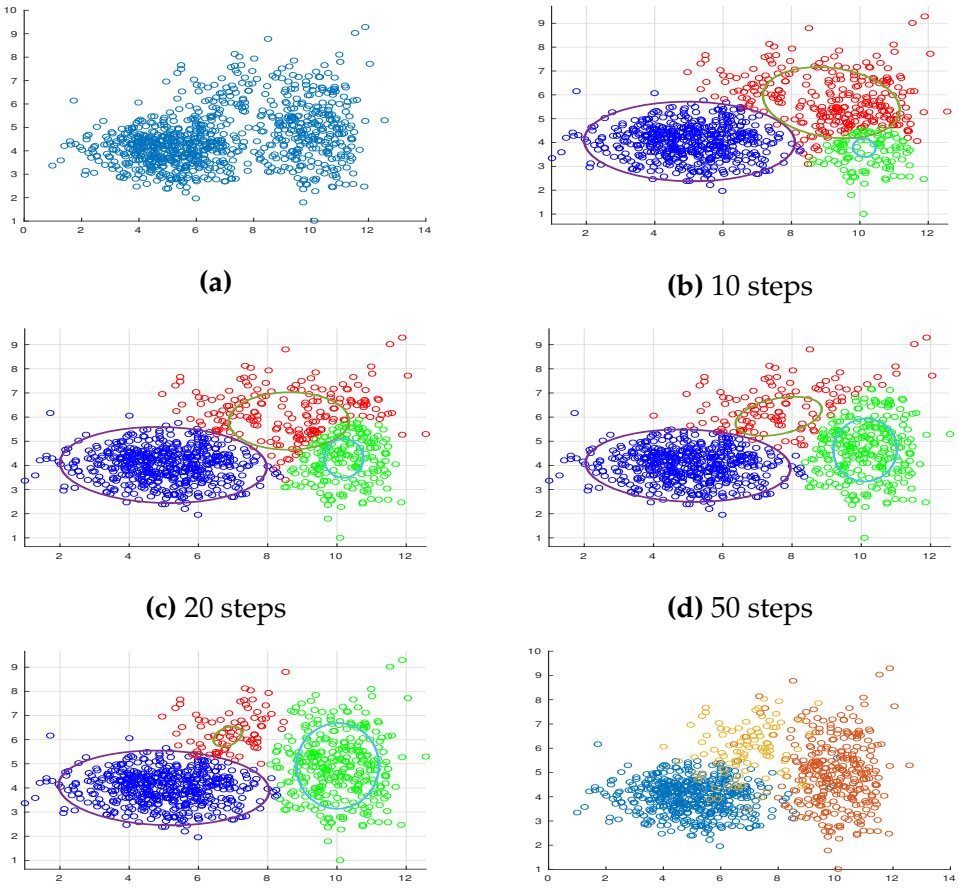
where N_k is the sum of the responsibilities in relation to k-th Gaussian.

After the M-step is finished, the *log-likelihood function* for the current estimation for the parameters θ is evaluated. The likelihood function of

MoG model is equivalent to the probability density function, evaluated for all samples $\bar{X}$. The log-likelihood function is therefore given by:

$$\mathcal{L}(\bar{X}|\theta) = \sum_{i=1}^{N} ln\big(f(\bar{x}_i|\theta)\big) = \sum_{i=1}^{N} ln\Bigg(\sum_{k=1}^{K} \omega_k f(\bar{x}_i|\bar{\mu}_k, \Sigma_k)\Bigg) \tag{2.19}$$

If the evaluation of the log-likelihood function does not reveal a convergence, the iteration continues by returning to the E step.



(a) **(b)** 10 steps **(c)** 20 steps **(d)** 50 steps **(e)** 154 steps **(f)**

Figure 2.16: Example of the progress of estimating the parameters of the Gaussians during the EM algorithm. The distribution in **(a)** is set to be modeled as a MoG model containing 3 Gaussians. The figures **(b)**, **(c)** and **(d)** show how the parameters of the Gaussians converge towards an optimal mixture, displayed in **(e)**. Figure **(f)** shows the ground truth distributions used to create the dataset in **(a)**. The size of the ellipsoids corresponds to the weight of each Gaussian.

The EM algorithm is not guaranteed to find the optimal solution for θ. Even finding a local maxima may be time-consuming. It is therefore

appropriate, especially for real-time applications, to set a threshold for the degree of convergence, and the allowed number of iterations, before the iteration is terminated. The choice of these two will be a trade-off between estimation quality and computation time.
Figure 2.16 illustrates an example of the convergence of the EM algorithm. Three bivariate normal distributions are created with slightly different mean and covariance. In figure 2.16b, the estimation is quite poorly compared to the ground truth distributions, showed in figure 2.16f. However, through several iterations, the algorithm reaches an optimal estimation of the MoG parameters, and convergence is detected in figure 2.16e. The color of the points indicates to what Gaussian in the mixture the points are closest to, and the separation of the points in figure 2.16e shows that the estimated MoG model discriminates the points in a way that is pretty close to the ground truth distributions.

2.5 Geometrical Features - Detecting Road Edges

Anthropogenic objects often exhibit prominent geometrical properties, otherwise not often present in nature itself. Road is an excellent example of such properties. From the perspective of a drivers, the edges of the road will appear as either straight lines, or smoothly curved lines (more or less, depending on the state of the road and road shoulder). This property is evidently also found in other structures along the road, especially in urban scenes. Distinguishing the lines which originate from the road edges from all other lines is a tricky matter, but there are some characteristics which can be generalized for the road edges. Firstly, these lines are usually pretty long, and they usually delimit the road area from an area of different characteristics. Also, the left and right road edge have a common *vanishing point*, i.e. the point in the *image plane* where the two spaciously parallel lines seemingly intersect. The apparent intersection of these lines is due to the perspective in which the lines are viewed from. The perspective of a camera, placed on top of a vehicle, is approximately perpendicular to the angle of the road edges. When the 3-dimensional road edges are projected into the 2-dimensional camera plane, they will no longer appear parallel as illustrated in figure 2.17.

The vanishing point can be used to aid the detection of road edges, which is shown in by the proposed methods described in [36] and [37]. The second approach, proposed by Hui Kong et. al, divides the road detection search into two main parts. The first part concerns finding the vanishing point, and the second part concerns the finding the edges which are moving towards the vanishing point, and simultaneously displays the characteristics of typical road edges.

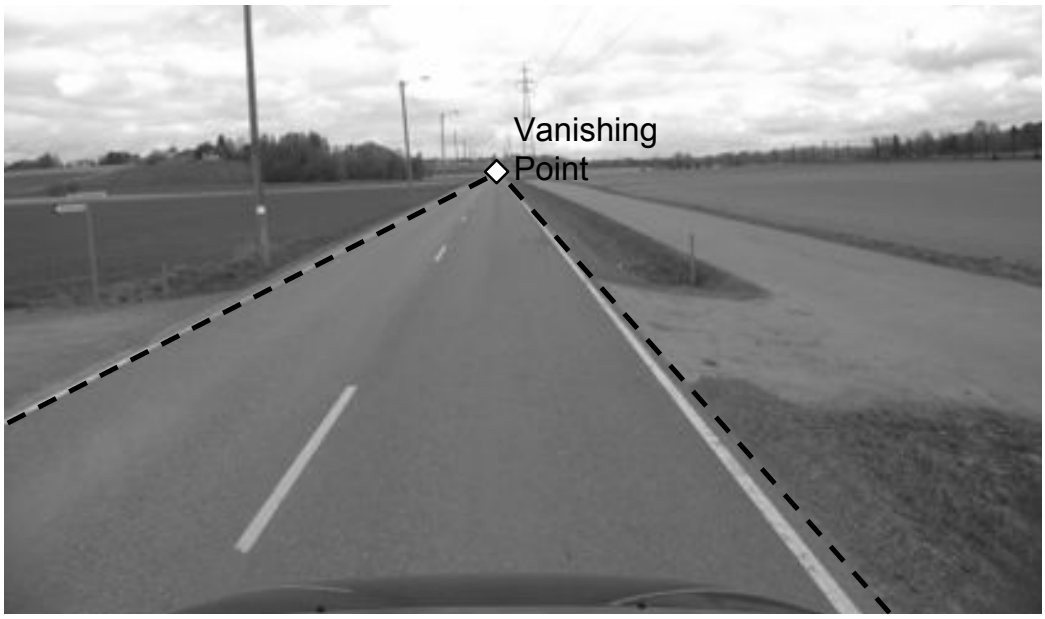


Figure 2.17: Illustration of two parallel road edges intersecting at a vanishing point.

The vanishing point detection part starts by finding the dominant orientation of each pixel in the image. This is conducted through a method known as *Gabor filtering*. Gabor filtering is an image processing procedure which weights higher the elements in the image with a certain frequency and orientation ([31], section 13.1). The procedure involves filtering the image with a 2D filter kernel, called a *Gabor kernel*, which is the product of 2D Gaussian and a 2D sinusoid. A Gabor kernel is parameterized by the covariance of a 2D Gaussian, and the phase, orientation, and wavelength of a 2D sinusoid. Consequently, a single Gabor kernel responds higher to elements with similar frequency, orientation, and scale.

In [37], to compute the response for several orientations, a set of Gabor kernels, $\mathcal{G}_{\omega,\phi}(x,y)$, is used. $\mathcal{G}_{\omega,\phi}(x,y)$ contains Gabor kernels with ϕ different orientations at ω different scales. Continuing, the image is filtered with each of these kernels, which produces $\omega \times \phi$ response images. These response images are then used to compute a *texture orientation image*, $\theta(x,y)$, where each pixel is assigned the estimated orientation of its associated neighborhood. $\theta(x,y)$ is computed by the *maximum average complex response* of $\mathcal{G}_{\omega,\phi}(x,y)$, which is a two-step procedure:

$$\begin{aligned} R_\phi(x,y) &= Average_\omega\{Real(G_{\omega,\phi})^2 + Imag(G_{\omega,\phi})^2\} \\ \theta(x,y) &= Argmax_\phi\{R_\phi(x,y)\} \end{aligned} \tag{2.20}$$

From this $\theta(x,y)$, the vanishing point is estimated through a local soft voting scheme, where each pixel is voted on by pixels gathered in a half-

circle beneath it (see [37] section IV). As explained in the article, this half-circle scheme is used to avoid favoring points higher up in the image.

When the vanishing point is found, the second part of the algorithm is initiated.
The search for lines which represents the left and right road edge is based on maximizing two different criteria. The first criteria is referred to in [37] as *the OCR criterion*. This criterion investigates the consistency of a line's orientation. OCR is measured by splitting the detected lines into a set of discrete points, and for each point, the angular difference between the point's orientation and the line's orientation is calculated. If this angular difference at each of these points is beneath a certain threshold, the line is considered to have a consistent orientation.
The second criterion is a color-based difference measure between two triangular areas, $A1$ and $A2$, on either side of line. The difference between area $A1$ and $A2$, is defined by the following equation:

$$diff(A1, A2) = \frac{|mean(A1) - mean(A2)|}{\sqrt{var(A1) + var(A2)}} \tag{2.21}$$

The two areas $A1$ and $A2$ are determined by three points: the vanishing point, the intersection point of the line and the image border, and lastly the intersection point of the image border and a line which is angled at a 20° offset of the original line.

The road edge is then selected as the line which maximized the product of the two measures, OCR and $diff(A1, A2)$.

Chapter 3

Method and Implementation

In this chapter, the road estimation algorithm is described in detail. This includes all of the methods and operations at work in the algorithm, and how these are assembled together.
The implemented methods for extracting the features are described, including the optimization methods necessary for fast computation. The usage of the Expectation Maximization algorithm for estimating the parameters of the MoG model of the road is presented. The output of the algorithm is also presented, along with a discussion of why this format is chosen.
Lastly, the implementation of the algorithm for detecting the road edges, through vanishing point detection, is described.

3.1 The Road Estimation Algorithm

The road estimation algorithm has been developed using a combination of Matlab, for prototyping and testing of different methods, and C++ with *OpenCV* [38] for implementation.
Matlab is utilized because of its efficient and clean graphical interface. It provides a lot of built-in functions which enables fast prototyping of code. The graphical interface provides helpful tools for debugging and visualizing of results and intermediate states.
After functions, methods or concepts have been proven valid in Matlab; the code has then been translated into C++ code, which is done for the purpose of computational efficiency. Matlab is a special-purpose tool, possessing optimized functions which make matrix iterations very efficient. However, when self-made matrix-iterating procedures are introduced, it becomes slow. For that purpose, C++ is a much more efficient programming language.
Together with the C++ programming language, the OpenCV library is used. OpenCV is an open source computer vision and machine learning library, which contains an extensive collection of optimized algorithms.

These algorithms are based on both classic and state-of-the-art algorithms.

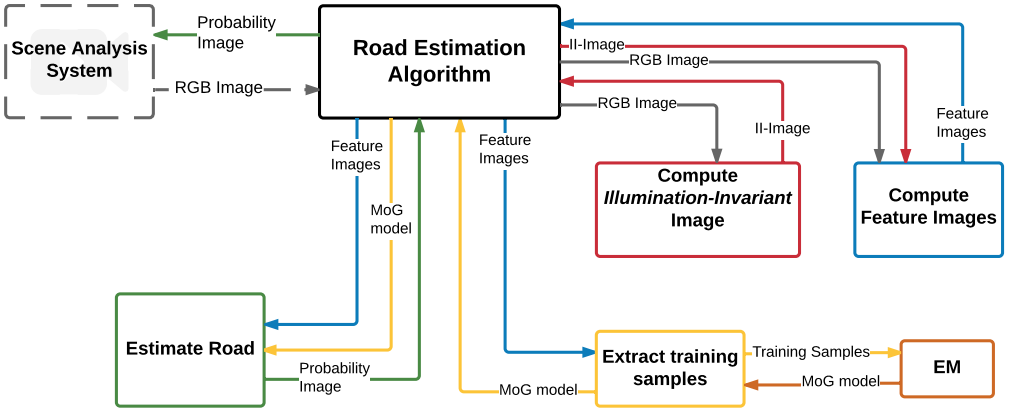


Figure 3.1: An overview of the different methods and operations at work in the road estimation algorithm; represented in the figure as rectangular boxes. The lines represent exchanges of data from one operation to another, and the arrow indicates the operation which receives the data. Each line is marked with a specific color, which matches the color of the operation from where the data was created.

In figure 3.1, a visual illustration of the road estimation algorithm is presented. As shown in the figure, there are several individual operations at work, and each is dependent on data from a different operation. Chronologically, the algorithm starts off by computing the illumination invariant image $\mathcal{I}$ from the RGB input image I_{RGB}. $\mathcal{I}$ is then used to compute the D-dimensional feature images $F = F_1, F_2, ..., F_D$, where D is the number of features. The color channels R, G and B in I_{RGB} are used as individual feature images in F. F is then used, along with the EM algorithm, to estimate the MoG model of the road, C_{road}. Finally, C_{road} and F is used to compute the probability image P, where each pixel contains a probability of being a road pixel.

Both the resolution and frame rate of the cameras used on the UGV, is adjustable. Therefore, the implementation of the algorithm allows all window sizes W used in the feature extraction methods to be specified by the user.

3.1.1 Features Extraction

The features chosen is a mixture of the three color channels R, G and B, and a set of textural features. The textural features include *Local Standard Deviation*, *Local Entropy*, and three features computed via the GLCM, namely *Variance*, *Inertia* and *Angular Second Moment*. As an effort to make the feature space more robust to shadows which are cast over the road,

all of the textural features are calculated from the illumination invariant image $\mathcal{I}$.

Local Standard Deviation

The local standard deviation feature is computed by the procedures described in section 2.2.3. A box filter H is chosen for blurring. If the user does not specify the size of H, it is set to $\frac{1}{35}$ the size of the image diagonal. The expectation of the local neighborhoods throughout $\mathcal{I}$ is thus interpreted as the result of blurring $\mathcal{I}$ by H, and Eq. (2.11) is then utilized to compute the standard deviation image. Algorithm 1 presents the pseudo-code for extracting the local standard deviation feature image.

Algorithm 1: Algorithm for calculating the local standard deviation feature-image

1 function localSdev $(\mathcal{I}, H)$;
Input : The illumination invariant image $\mathcal{I}$ and the box filter H
Output: Local Standard Deviation feature-image I_σ
2 x $\leftarrow \mathcal{I}$;
3 $x^2 \leftarrow \mathcal{I}^2$;
4 E[x] $\leftarrow$ blurImage(x, H);
5 E[x^2] $\leftarrow$ blurImage(x^2, H);
6 $I_\sigma \leftarrow \sqrt{E[x^2] + (E[x])^2}$;

Local Entropy

The local entropy feature is extracted by computing the histogram for the neighborhoods contained in a sliding window, iterating through $\mathcal{I}$. The entropy of each neighborhood is then calculated using Eq. (2.9).
To speed up the algorithm, the sliding window iteration is done in parallel. The parallel workers iterate trough the rows of $\mathcal{I}$. For each row, the $\frac{W}{2}$ previous and the $\frac{W}{2}$ subsequent rows of $\mathcal{I}$ are extracted, where W is the chosen size of the sliding window. This results in a $W \times M$ section of the image, $\mathcal{I}_{\Delta W,M}$, where M is the width of $\mathcal{I}$. Then, the window slides along $\mathcal{I}_{\Delta W,M}$ with a regular sliding window iteration, computing the histogram for each window, and calculating the local entropy for each histogram. This allows for sliding window computation to be executed in parallel, as illustrated in figure 3.2. If not specified by the user, W is set to $\frac{1}{35}$ the size of the image diagonal.

In addition to the parallelization of the sliding window, there is another possible optimization. The straightforward algorithm involves extracting

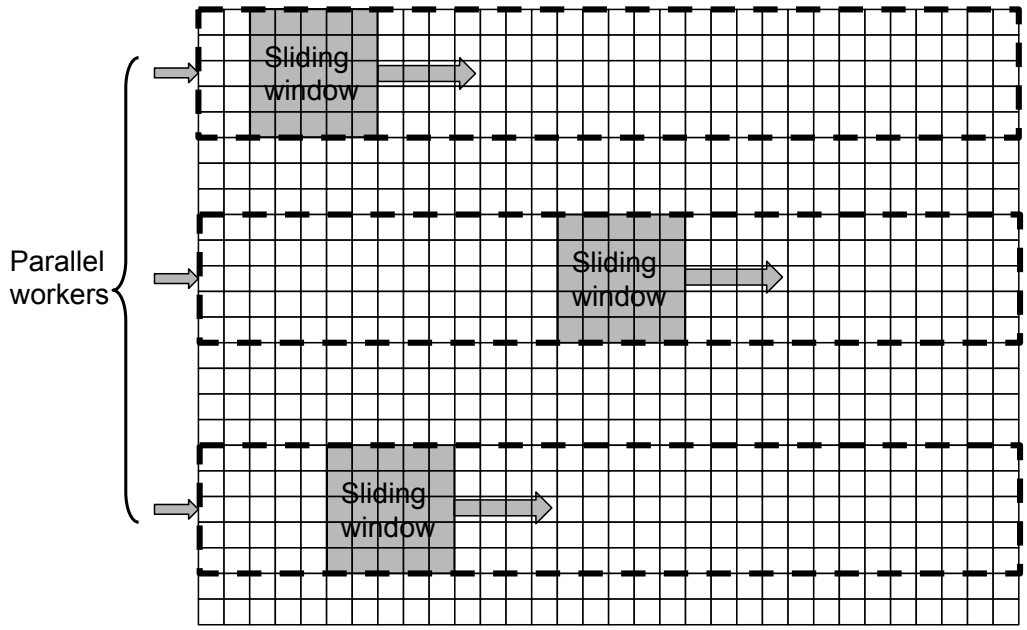


Figure 3.2: Illustration of how the sliding window algorithm is parallelized. This illustrations shows 3 parallel workers creating image regions of size W×M, where W is the window size, and M is the width of the image. The regions are marked here as the dashed lines.

a new window for each sliding step and then calculating the histogram based on the pixel values within that window. However, most of these pixels have already been seen in the previous window. There are in fact only W new pixels entering the window, and W pixels leaving it, culminating in a total of $2W$ needed histogram updates for each window iteration. So, to avoid counting most of the pixel values over and over again, the values of the pixels leaving the window is subtracted from the histogram, and the new pixel values are added, as described in [39]. This procedure decreases the complexity of computing the histogram for each window from W^2 to $2W$. The exception is, of course, the initial windows at the beginning of $\mathcal{I}_{\delta W,M}$, where all pixel values must be counted.
To illustrate the decreased computational complexity, both the straight forward algorithm and the optimized version were used to compute the entropy of 20 different images. The results, shown as a line plot in figure 3.3, shows a tremendous improvement in terms of computation time. These results stem from the two version of the algorithm implemented in MATLAB, so the time-consumption is not quite comparable to the C++ version. The results still give an indication of the degree of improvement. Algorithm 2 presents the pseudo-code for extracting the local entropy feature image E, with optimizations.

Algorithm 2: Algorithm for calculating the local entropy feature-image.

function localEntropy $(\mathcal{I}, W)$;
Input : Illumination invariant image $\mathcal{I}$, and the window size W
Output: Local entropy feature image E

for all n $\in$ *image rows of* $\mathcal{I}$ **do in parallel**
    $\mathcal{I}_{\Delta W,M} \leftarrow$ get_row_selection($\mathcal{I}$, n-$\frac{W}{2}$, n+$\frac{W}{2}$);
    H $\leftarrow$ calc_initial_histogram($\mathcal{I}_{\Delta W,M}$);
    E(n, 0) $\leftarrow$ calc_local_entropy(H);
    for $m \in$ *image columns except first* **do**
        $\text{col}_{out} \leftarrow$ get_column($\mathcal{I}_{\Delta W,M}$, m-1);
        $\text{col}_{in} \leftarrow$ get_column($\mathcal{I}_{\Delta W,M}$, m+W);
        H $\leftarrow$ subtract_from_histogram(H, col_{out});
        H $\leftarrow$ add_to_histogram(H, col_{in});
        E(n, m) $\leftarrow$ calc_local_entropy(H);
    end
end

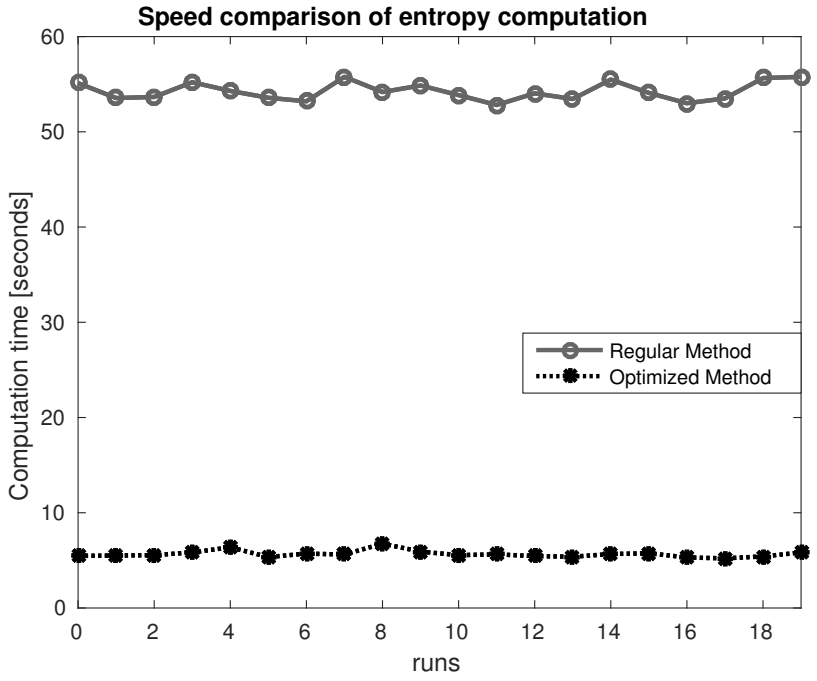


Figure 3.3: Comparison of the computation time for extracting the local entropy feature image, using the straightforward algorithm and the optimized algorithm.

GLCM - Variance, Inertia and Angular Second Moment

The textural features based on the GLCM are calculated as described in section 2.2.2. Still, estimating local texture features based on the GLCM is a highly complex process. Luckily, there are a few optimizations possible.

The first optimization is the same the one applied to the local entropy extraction, where intensity histograms of the local areas were computed for each window in a sliding window iteration. The same process of only adding the new columns which enter the sliding window is applied to the GLCM approach as well. However, because the counts in the GLCM are dependent on a spatial, pairwise relationship between the pixels, the sliding window optimization cannot be implemented directly. For the implementation to work, a transformation of the GLCM coordinate system must be applied first.

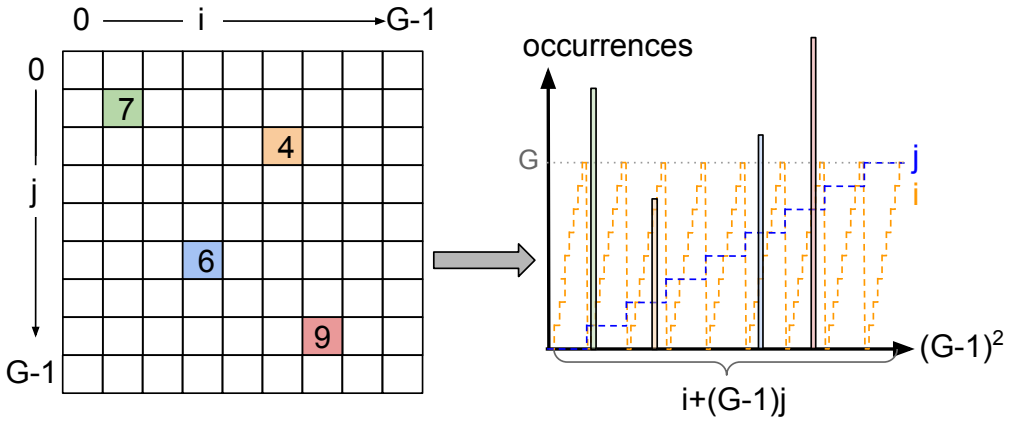


Figure 3.4: Illustration of the GLCM coordinate system transformation. The GLCM coordinates i and j, in the left figure, correlates to to the coordinate k, in the right figure. How k relates to i and j in the right figure, are visualized through the yellow and blue dashed lines. Each of the co-occurrence counts shown in the left figure corresponds to the bar in the right figure with the matching color.

In a regular GLCM matrix, each column i indicates the normalized intensity value of the base pixel, while the each row j indicates the the normalized intensity of the comparison pixel. Together this forms a 2-dimensional coordinate system, and the values contained in the the this array corresponds to the normalized intensity co-occurrences. This 2-dimensional coordinate system, however, could be transformed into a 1-dimensional coordinate system, according to the following equation:

$$k = i + (G - 1)j \tag{3.1}$$

where k is the coordinate of the new 1-dimensional GLCM and G is the number of GLCM levels.
Figure 3.4 shows an illustration of the transformation. The orange and blue lines is meant to illustrate how the increasing indexes i and j corresponds to new coordinate k. The values of these lines are not in any way correlated with the co-occurrence counts but are purposed for visualization.

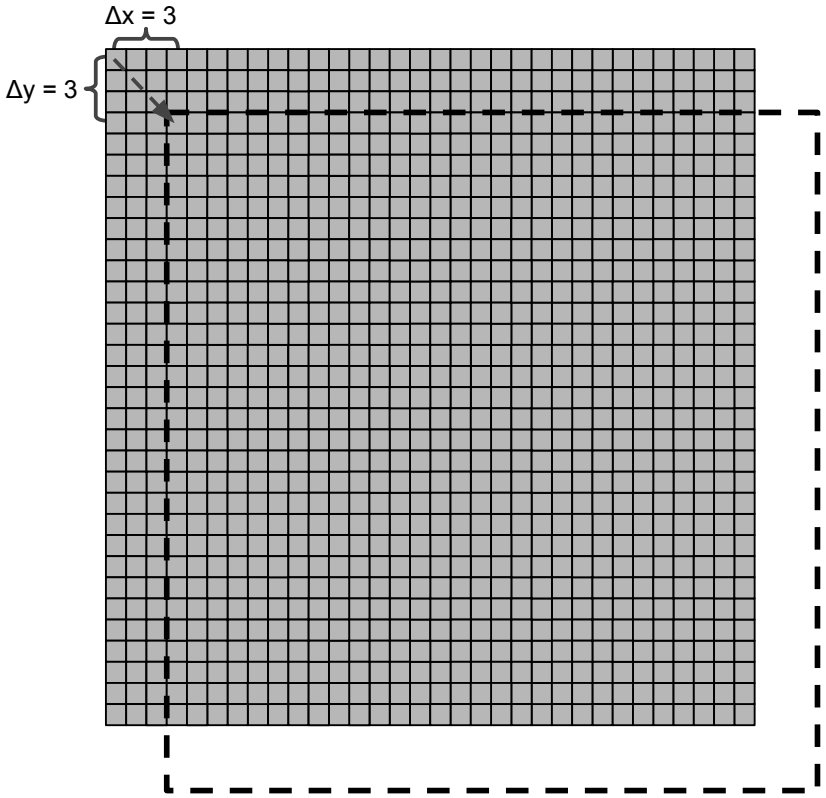


Figure 3.5: Illustration of the extraction of $\mathcal{I}_\Delta$, represented here as the area within the dashed line. The area outside the original image is not included in $\mathcal{I}_\Delta$.

Using this new coordinate system, it is now possible to compute the co-occurrence counts directly on the image. First, the image must be remapped from [0, R] to [0, G], G still being the number of levels in the GLCM, and R being the range of the gray-level intensity. The resolution of the image remains the same. Then, equation (3.1) is applied directly to the image. This is made possible by realizing that the gray-level intensity of all the pixels in the image $\mathcal{I}$, which are now remapped to [0, G], represents the coordinate i of the GLCM. Moreover, instead of locating the coordinate j by searching for the pixel displaced by Δx, Δy, a region of $\mathcal{I}$, which has its origin located Δx, Δy displaced from the origin of $\mathcal{I}$, is extracted. This

region, illustrated in figure 3.5 by the dashed black line, will be referred to as $\mathcal{I}_\Delta$ (the area outside of the original image $\mathcal{I}$ is excluded). $\mathcal{I}_\Delta$ contains the gray-level intensities of all pixels displaced Δx, Δy relative to all the pixels in $\mathcal{I}$ (except for those near the borders), and the pixels in $\mathcal{I}_\Delta$ represents the index j. Eq. (3.1) can then be used to compute the co-occurrence image $\mathcal{I}_k$, where each pixel represents a co-occurrence count, by replacing i and j with $\mathcal{I}$ and $\mathcal{I}_\Delta$, respectively:

$$\mathcal{I}_k = \mathcal{I} + (G-1)\mathcal{I}_\Delta \tag{3.2}$$

Then, it is just a matter of sliding a window over $\mathcal{I}_k$, increasing the GLCM counts for each new column, and decreasing the counts for each leaving column. Each pixel $\mathcal{I}_k(x, y)$ maps directly to the GLCM coordinate k.

3.1.2 Modeling the road class

To estimate the road model through the EM algorithm, a set of training samples are required. The training sample set is extracted in real-time from a small window at the lower center of the image, i.e. the part of the image that contains what is right in front of the UGV, and most likely contains road. The underlying assumption is that the UGV is positioned on the road at all times. Although this method fails if the window contains other elements than the road, it is very efficient and easily implemented, and worked well for early testing on recorded sequences.

When the road samples have been extracted from this window, they are used to build the model of the road. The road is modeled as a Mixture of Gaussians, using the EM algorithm described in section 2.4.1. The MoG modeling method is chosen based on its adaptability to unknown probability distributions of complex characteristics. As mentioned earlier, the condition of the road may change over time, and to adapt to these changes, the modeling method must be general enough to avoid too specific assumptions about the road conditions. The MoG density provides such a modeling method. The road may contain elements which do not resemble the usual gray tarmac. Such elements could, for example, be road markings, manhole covers, shadows, or similar. Should these elements enter the extraction window, it is undesirable to let these pixels affect the learning procedure too much. Although these elements may be part of the road, they can also contribute to pulling the model parameters towards other scenery objects, in terms of the feature space. The hypothesis is that, by modeling the road as a MoG model, these elements will only be part of a lower weighted Gaussian in the mixture, and the contribution to the model will hopefully be minimized.

The EM algorithm is chosen based on its suitability for fast, online machine learning. The aspect of the EM algorithm which makes it particularly attractive for the road estimation algorithm is the possibility of feeding known model parameters to the training, along with the training data, and, thus, achieving faster convergence. The underlying assumption is that the difference of the road class from image to image is small. Consequently, the MoG density model should not have to change a lot for each new image. By feeding the known model parameters from the last image into the EM algorithm for the current image, just a small adjustment of the Gaussian densities is necessary. The pseudo-code for this implementation is presented in Algorithm 3.

Algorithm 3: Algorithm for learning the MoG model from an extracted training sample set, through the EM algorithm

```
function learn_model (I_f, K);
  Input  : The N×M×D feature image F, where [N,M] is the image
           size and D is the number of features, and the desired
           number of Gaussians K
  Output: The model parameters [μ_k, Σ_k, ω_k]
W_f ← extract_training_samples(F);
if this frame is the first frame then
    [μ_k, Σ_k, ω_k] ← EM(W_f, K);
end
else
    [μ_k, Σ_k, ω_k] ← EM_with_initial_parameters(W_f, K, μ_k, Σ_k, ω_k);
end
```

It is desirable to have enough Gaussian densities to avoid sudden changes in the road appearance to influence the model too heavily. Simultaneously, an excessive number of Gaussian densities will result in a very complex MoG model, which further will cause high computation time for the EM algorithm.
The optimal number of Gaussians in the MoG model is not easily generalized to all situations. As an example, estimating tarmac country road, in scenery dominated by green grass, is significantly easier than detecting a dirt road. Therefore, the current implementation of the algorithm allows the user to specify the number of Gaussians in the mixture. By default, if not specified by the user, the number of Gaussians is set to 3.

To minimize the impact of temporary road elements, such as road markings, has on the road model, the modeling method has been implemented with a learning rate. For every new set of training samples

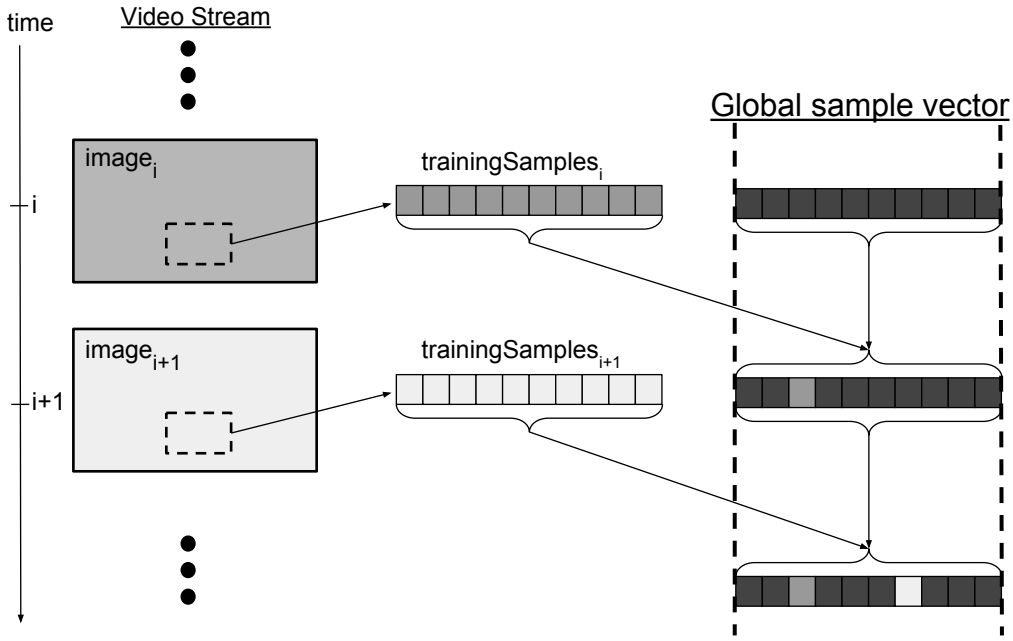


Figure 3.6: Illustration of how the learning rate is retained by only using 10% of the new training samples contained in the extraction window. For each new image, the training samples are extracted from the window, marked here as a dashed rectangle. 10% of the training samples are then merged with the global sample vector trough a random selection procedure. The 90% selection of the global sample vector is also randomly selected.

extracted from the window, only 10% of the samples are used to model the road. This 10% are chosen by first, randomly shuffle the sample vector, and then extracting the first 10% of the vector. The remaining 90% of the learning stems from an already existing, global sample vector. Similar to the new sample vector, the global sample vector is randomly shuffled before 90% of the vector is extracted. The two vectors are then concatenated before it is passed to the EM algorithm. The concept of the retained learning rate is illustrated in figure 3.6.

3.1.3 Probability Estimation

Without any proper way of locating surrounding elements in the image, the only model stored in the system is that of the road. Consequently, the only probability measure will be with respect to this model. Moreover, because of the large variation in road texture, lighting conditions, shadows and such, elements can appear very differently from image to image. The solution is to produce a soft probability image, where each pixel contains

a likelihood of being part of the road.
From the EM algorithm, the parameters μ_k, Σ_k and ω_k for each of K Gaussian density in the MoG model are known. The distance from each of the pixel to the mean of the model is calculated from the multivariate probability density function for MoG models (Eq. (2.14)). The calculation is done in parallel, which enables fast computation of the probability image P.

3.2 Vanishing Point Detection

The method for detecting road edges is based on the approach proposed by [37]. The vanishing point detection algorithm has been implemented with the same Gabor filter bank, where the Gabor kernels are computed at $\phi = 36$ different orientations, at $\omega = 5$ different scales. If the size of the Gabor kernels is not specified by the user, it is set to $\frac{1}{10}$ of the image diagonal. A selection of the Gabor kernels used is presented in figure 3.7.
The implementation for detecting the left and right road edge from the estimated vanishing point is also equivalent to the one proposed in [37]. For each detected vanishing point, 29 suggested linear road edge lines are computed. The best suggestion is chosen based on the OCR measure, and the color-based difference measure between the two triangular areas on either side of the line, as described in section 2.5. The two triangular areas, $A1$ and $A2$, are computed from the vanishing point, the point where the suggested line intersects with the image border, and the point where a new line, with a 20° offset from the suggested line, intersects with the image border.

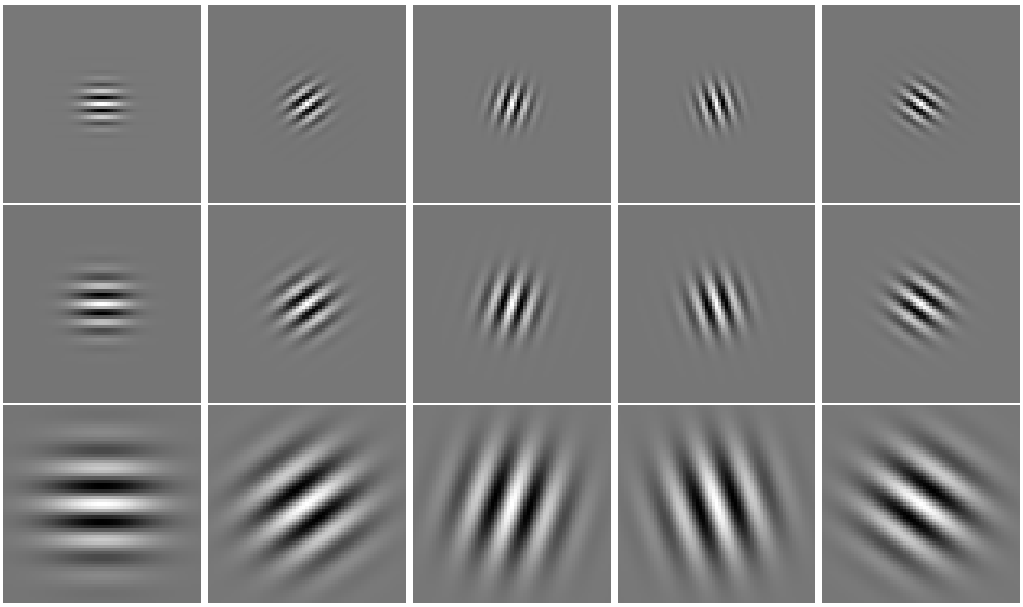


Figure 3.7: A selection of some of the Gabor kernels contained within the Gabor filter bank.

Chapter 4

Experiments and Results

This chapter will present the methods used to evaluate the performance of the proposed algorithm. First, the image sequences used for evaluation are presented. Then, the experiment procedures are explained, including the research question for each experiment. For each experiment, a detailed explanation of the procedure is presented, including parameter settings, the image sequences used, and the presentation method of the results. Since the experiments are dependent on the results of the previous ones, the experimental results are presented immediately after each experiment, including a short, intermediate analysis of the results.

4.1 Experiment Data

The experiments in this section have been conducted on several image sequences from different sources. From the UGV, there have been three video sequences selected for testing. The video sequences differ a lot in terms of road structure, lighting conditions, and surrounding scenery, and each offers a different degree of difficulty. Some of the images from each sequence is displayed in figure 4.1, 4.2 and 4.3.

(a) **(b)** **(c)**

Figure 4.1: A selection of image from the first recorded sequences from the UGV.

(a) **(b)** **(c)**

Figure 4.2: A selection of image from the second recorded sequences from the UGV.

(a) **(b)** **(c)**

Figure 4.3: A selection of image from the third recorded sequences from the UGV.

In extension, some additional datasets have been used [22], [1]. The necessity for this is based on the need for testing the algorithm in varying conditions.

4.2 Experimental Plan

The experiments are divided into three main focus areas, each aiming to determine different research questions. The first two experiments are focused on feature extraction, the third on modeling and feature combination, and the fourth experiment on vanishing point detection.

The first experiment evaluates the separability of textural features extracted from the illumination invariant image, compared the corresponding features extracted from the gray level intensity image. The second experiment extends the first experiment and seeks to investigate in greater detail how well different features are capable of discriminating between objects in a typical road scenery, and to compare this to the computation time of extracting these features.

The third experiment is by far the most extensive, and aims to evalu-

ate the performance of the road estimation algorithm under four different conditions. This performance is tested for different combination of features, giving an indication of the necessity of the feature combinations in the various conditions. Also, the third experiment seeks to evaluate the performance when the road is modeled as a MoG model with 3 Gaussians, compared to the version where the road is modeled as a single Gaussian. The purpose of this experiment is that the algorithm should be tested on sequences with varying conditions, and thus, an investigation of how the different features and number of Gaussians affect the performance in these conditions is necessary.

The fourth and last experiment aims to determine roughly the suitability of the vanishing point detection method for improving the accuracy of the road estimation algorithm. The experiment investigates the accuracy of estimating vanishing points, compared to the computational load of this estimation.

4.3 Reference Algorithm

The KITTI Vision Benchmark is one of the main computer vision benchmarks for autonomous driving [1]. It contains datasets for road segmentation, stereo vision, optical flow, and more. It also contains on-line leaderboards over the highest scoring algorithms for the different vision-based applications.

The road dataset is a collection of 289 training images, and 290 test images, which are the ones used for evaluation. All of the images are captured in urban scenery. However, none of the training or test images appear chronologically, i.e. both sets are an unsorted collection of random road images. The reason is that the KITTI road benchmark is meant to evaluate machine learning algorithms and classifiers which are trained off-line on the training images, and only evaluated on the test images. The road estimation algorithm proposed in this thesis violates this evaluation procedure since it is based on on-line machine learning. Furthermore, one of the key assumptions for the usage of the EM algorithm is that the variation in road appearance between two consecutive images is minimal. This assumption is no longer correct for the KITTI datasets since the images appear unordered.

Although the road estimation algorithm cannot be submitted to the KITTI Vision Benchmark, the Benchmark can still be used as a reference point. The submitted algorithms in the KITTI leaderboards are evaluated based several measures: *Maximum F1-Measure* ($F1_{max}$), *average precision* (AP), *precision* (PRE), *recall* (REC), *false positive rate* (FPR), and *false negative*

rate (FNR). The last four measure are based on the threshold value which yields $F1_{max}$. All of the highest ranked approaches in the leaderboards are based on Convolutional Neural Nets.

The road estimation algorithm presented in this thesis will not be evaluated against a reference algorithm, based on the lack of a suitable method for doing so. Still, the KITTI leaderboards will be used to get an idea of what estimation performance can be considered as adequate.

4.4 Experiment 1 - Texture from Illumination Invariant Images

The hypothesis explored in this experiment is that textural features will benefit from being extracted from the illumination invariant image, rather than from the grayscale intensity image. In order to test this, the *KITTI dataset for road/lane detection* [1] is used because it contains a good variations of road images.

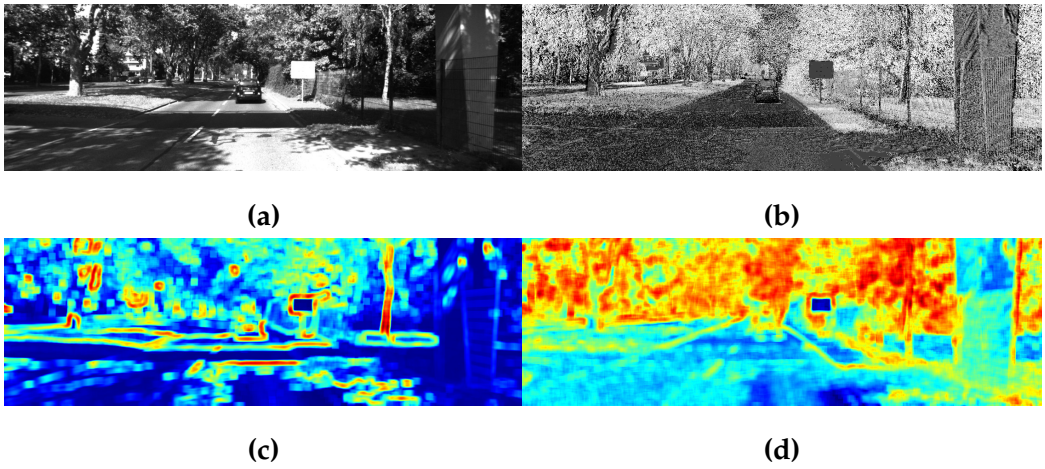


Figure 4.4: The images above shows **(a)** the grayscale intensity and **(b)** the illumination invariant version of the same image, obtained from the KITTI dataset. Image **(c)** and **(d)** shows the local standard deviation feature images computed from the two, respectively.

A selection of ten images, containing different amounts of shadows, is selected. For each image, the textural features are computed as presented in figure 4.4. Through the binary label images, the pixels belonging to each of the classes **Road** and **Environment** are collected into two vectors. When this procedure is done for all ten images, a Gaussian distribution is fitted to the data belonging to each of the classes **Road** and **Environment**, thereby producing a plot containing a visual representation of the feature distribution of the **Road** compared to that of the **Environment**. This is done for both the illumination invariant and the grayscale intensity version of the image, thus producing two plots which is used for analysis, presented in figure 4.5, 4.6 and 4.7
Since different texture features may respond differently to the two image types, three textural features have been selected for investigation, namely *local standard deviation*, *local entropy*, and *local variance* calculated from the GLCM.
The images in the KITTI dataset is of size 1242×375p. The window sizes chosen are 21×21, for local standard deviation, and 15×15p for local

entropy. The variance is computed from a GLCM of 16 levels, which again is computed from a sliding window of 31×31 pixels.

4.4.1 Results from Experiment 1

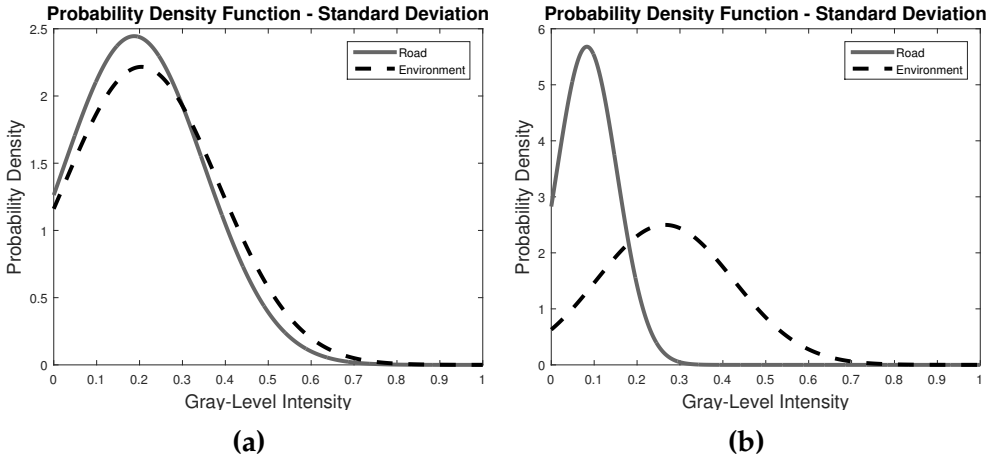


Figure 4.5: The plots show the distribution of the classes **Road** and **Environment** in terms of the local standard deviation feature image, calculated from **(a)** the grayscale intensity image, and **(b)** the illumination invariant image

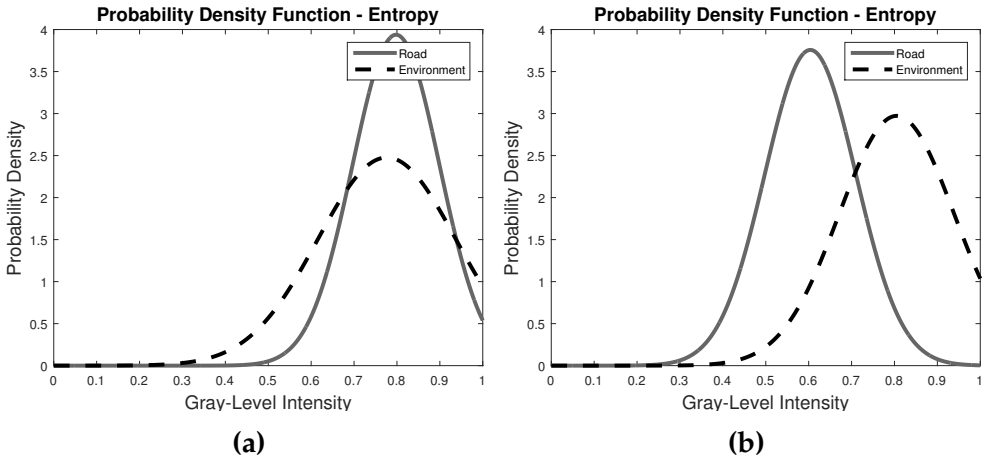
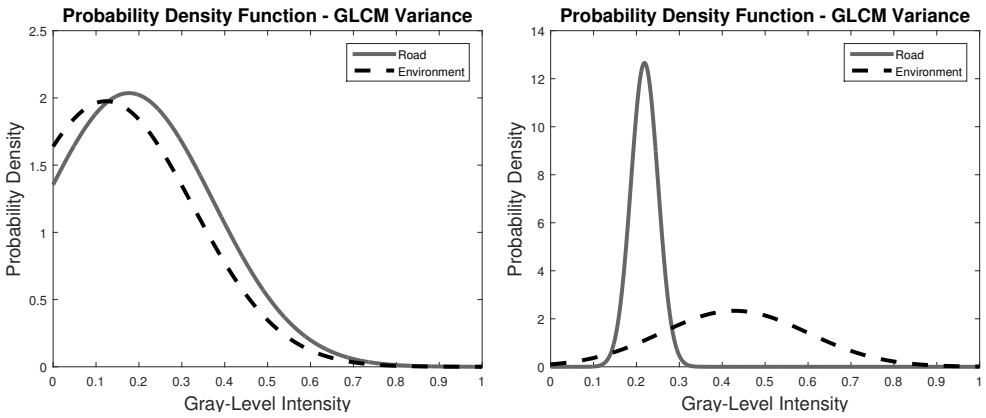


Figure 4.6: The plots show the distribution of the two classes in terms of the local entropy feature image, calculated from **(a)** the grayscale intensity image, and **(b)** the illumination invariant image



(a) **(b)**

Figure 4.7: The plots show the distribution of the two classes in terms of the local variance feature image, calculated via the GLCM from **(a)** the grayscale intensity image, and **(b)** the illumination invariant image

4.4.2 Analysis of the results of Experiment 1

The results from Experiment 1 display a distinct improvement in terms of separation of the road class from everything else in the image, when the textural features are extracted from the illumination invariant images. In figure 4.5a, 4.6a and 4.7a the two classes are more or less encased in one another, meaning it would be nearly impossible setting a threshold to separate the two. In figure 4.5b, 4.6b and 4.7b the two classes are much more distinguished. For local standard deviation and GLCM variance, the road class is much sharper when computed from the illumination invariant image, meaning the variance of the distribution is lower. For entropy, it is pretty much the same variance for both images. However, there is not only an upside of extracting texture from the illumination invariant image. As was presented in figure 2.3, the illumination invariant image adds a significant amount of noise, and it is evident that the computation cause a decrease in texture distinctiveness. Although this decrease is undesirable, it is crucial to have a method to avoid shadows spoiling the estimation. So, for now, texture computation from the illumination invariant image is more appealing.

4.5 Experiment 2 - Feature Separability vs. Computation Time

To test the *feature separability* for multiple object classes, a dataset containing images that are properly labeled, with individual labels for each of the many object types existing in the scenery, is necessary. The procedure in this experiment is chosen in light of the fact that there are no perfect features, i.e. each feature performs better in certain circumstances, and worse in others. The goal of this experiment is not to find the optimal features, but rather to give an indication of how well the features can discriminate between the different objects in the scenery.
To perform this analysis, 41 images from the KITTI dataset are selected. The original dataset only includes labeled images with labels for "road" and "not road", and for this experiment, labeled images for several object classes are needed. Therefore, the labeled images have been gathered from a second source [40].

The features tested in this experiment are:

- The three color channels [R,G,B]
- SDEV = Local standard deviation
- ENTR = Local entropy
- GLCM VAR = Variance computed from the GLCM
- GLCM INR = Inertia computed from the GLCM
- GLCM ASM = Angular second moment computed from the GLCM

The procedure of this experiment starts by extracting all eight feature images from all of the 41 images. For each feature, all pixels in the 41 feature images are divided into 7 separate vectors, according to the labeled images. The labels used are: *Road*, *Buildings*, *Sky*, *Cars*, *Pedestrians*, *Trees*, and *Vegetation*. This results in 7 class vectors for each of the 8 features. A Gaussian distribution is then fitted to each of these vectors, and lastly, the probability density function (PDF) is computed for each of these distributions. The result displays a probability density function computed for each object class at each feature. The results are presented as eight individual probability density function plots.
Also, a table listing the percentage of the *joint overlap* between the **Road** PDF and the PDF of all other classes, for every feature, is presented. The joint overlap is measured as the sum of *False Positive* (FP) and *False Negative* (FN) of the Road PDF when compared to the PDF of the other classes. A visualization is shown in figure 4.8. Since the density functions are normalized, the integral of these areas corresponds to the portion of the two

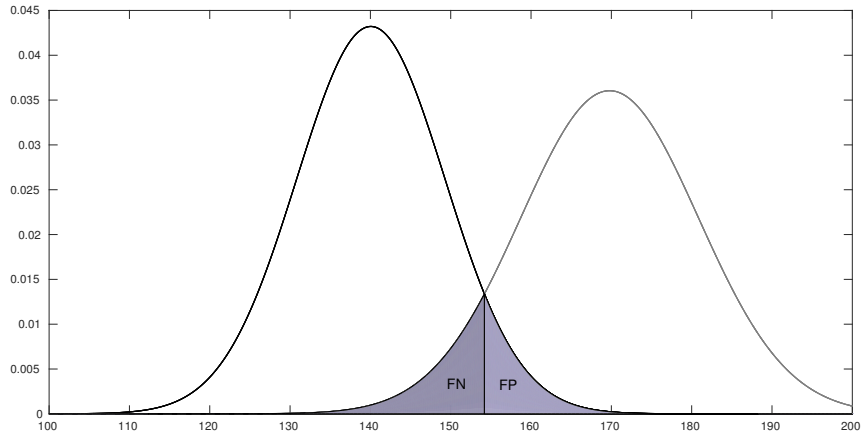
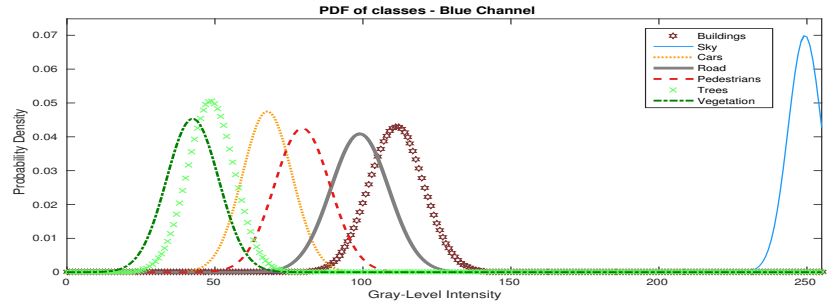


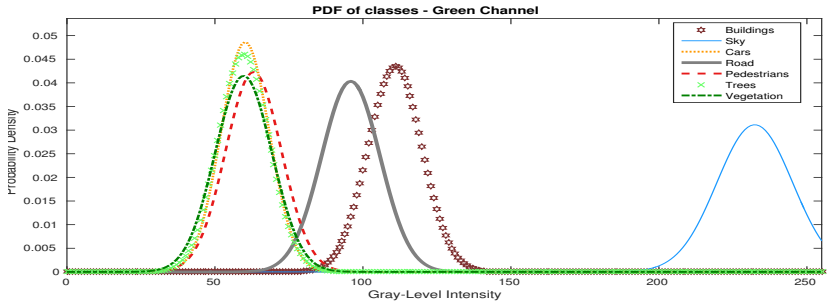
Figure 4.8: The figure show how the joint overlap percentage is calculated. The two areas, marked as "FP" and "FN", are summed together. Since the two functions are normalized, these combined areas amounts to the error between the two arbitrary classes.

densities which will cause either false positives or false negatives, when a threshold is set at the point where the two curves intersect.

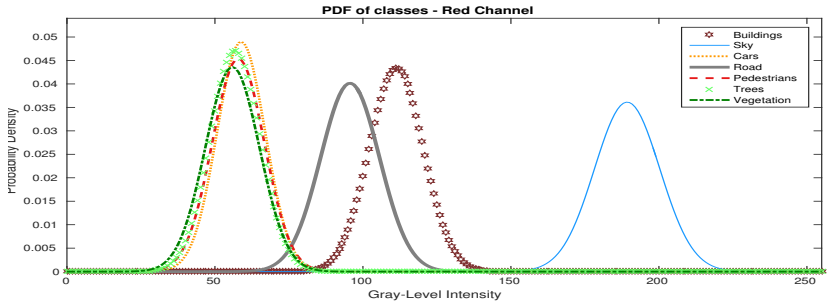
4.5.1 Result of Experiment 2



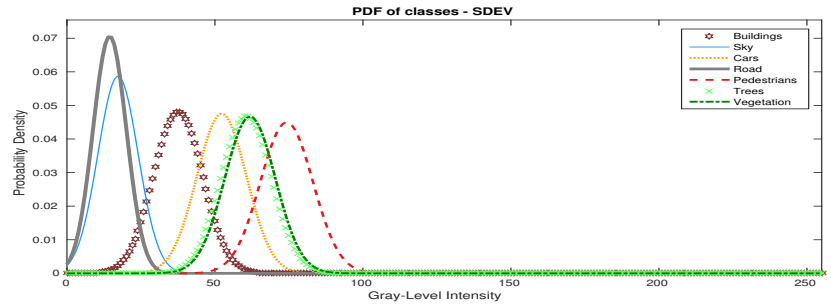
(a)



(b)

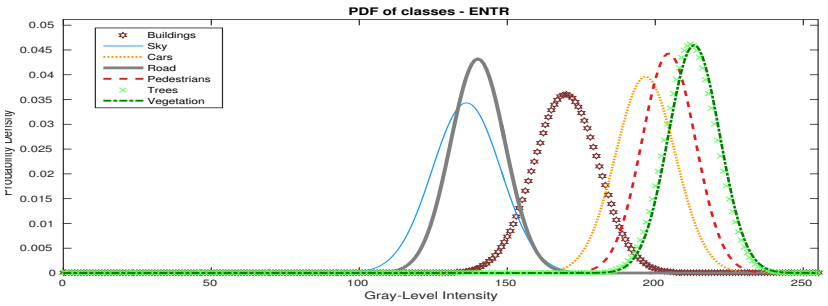


(c)

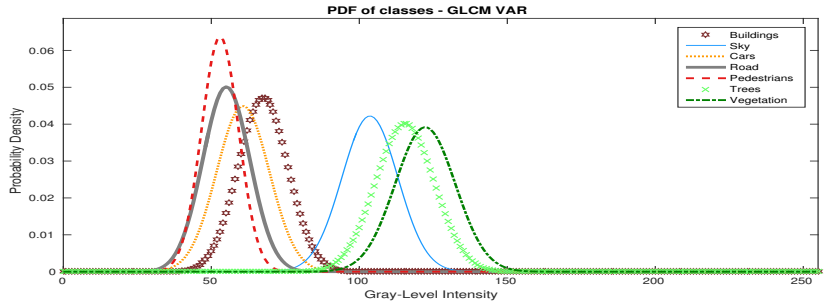


(d)

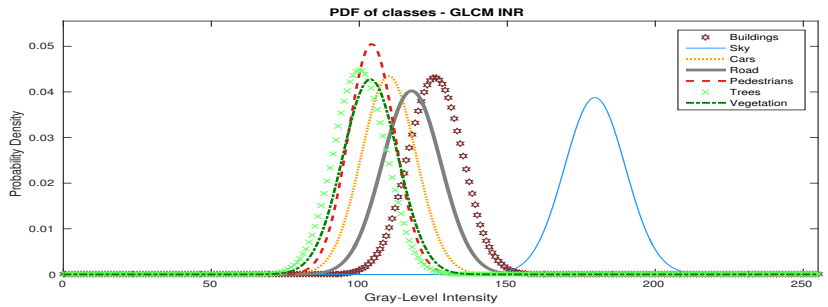
Figure 4.9: This figure displays the Probability Density function for the 4 features: (a) blue color channel, (b) green color channel, (c) red color channel, (d) local standard deviation.



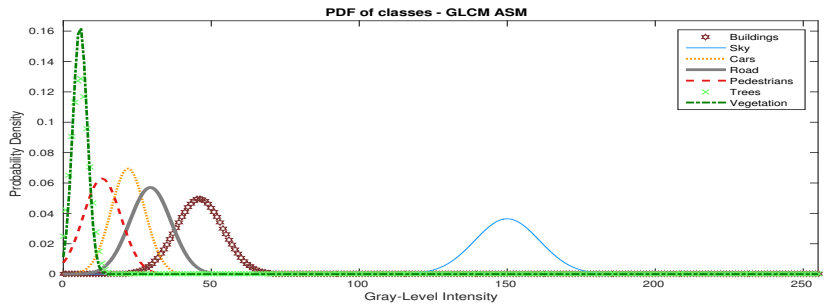
(a)



(b)



(c)



(d)

Figure 4.10: This figure displays the Probability Density function for the 4 features: (a) local entropy, (b) sum of squares (calculated from the GLCM), (c) inertia (calculated from the GLCM), (d) angular second moment (calculated from the GLCM).

	Joint overlap percentage					
	Blds*	Sky	Cars	Pdst*	Trees	Vgtn*
Blue	51.4%	0.0%	8.4%	30.9%	0.4%	0.2%
Green	43.1%	0.0%	4.8%	8.9%	5.1%	6.3%
Red	41.3%	0.0%	4.2%	4.4%	3.5%	3.7%
SDEV	9.5%	80.4%	0.7%	0.0%	0.1%	0.1%
ENTR	14.2%	82.7%	0.3%	0.0%	0.0%	0.0%
GLCM VAR	44.8%	0.5%	73.4%	84.4%	0.1%	0.0%
GLCM INR	67.5%	0.2%	68.5%	44.4%	35.1%	46.9%
GLCM ASM	26.9%	0.0%	54.6%	22.0%	1.6%	1.0%

Table 4.1: The table shows the shows the percentage of joint overlap of the densities for the classes: Buildings (Blds), Sky, Cars, Pedestrians (Pdst), Trees, and Vegetation (Vgtn), when compared to the density of the road class. The joint overlap is calculated for each of the 8 features listed.

Feature	Average computation time
SDEV	7 ms
ENTR	161 ms
GLCM VAR	568 ms
GLCM INR	542 ms
GLCM ASM	549 ms

Table 4.2: The table shows the average computation time of extracting the features of all 41 images. Since the color channels can be retrieved directly from the image, they are not included in this table. All images are of size 1242×375p.

4.5.2 Analysis of the results of Experiment 2

In figure 4.9 and 4.10, the densities of the different objects for the different feature images are show. The results suggest that the prior assumption, i.e. that the road's separability to other object types varies with feature investigated, is correct. For instance, in figure 4.9, the road is easily separated from the sky in all of the color channels, while in the local Entropy feature, it is not that easily separated. For the entropy feature, the two densities have a joint overlap of 82.7%. Contrarily, the road is very well separated from cars, pedestrians, trees and vegetation in the entropy feature. This is in agreement with the assumption that the road has a lower degree of disorder than these objects.
For all features, the hardest object class to distinguish from the road was buildings. Although undesirable, this problem is not easily dealt with for any feature. The reason is that buildings display significant variations in terms of building material, color, shape, patterns, etc. A particular difficult building type to deal with is concrete buildings. These buildings have

very similar features as the road, both textural and color. To solve this issue, one will have to explore other features than those based on color or texture, such as lines, curves, and object boundaries.
A surprising, and interesting result of this experiment, is that the features computed via the GLCM performs poorest by far. Since the GLCM takes into account the spatial aspect of the patterns, it is usually a better choice of texture description. However, because of the perspective of which the road is viewed from, the texture of the nearby road will not have the same spatial frequency as the far away road. The texture of nearby road may display similar properties as the texture of bush far away, and thus, the GLCM approach fails in this case. To make the GLCM approach reliable for feature extract, a better approach would perhaps be to estimate the GLCM from a window with a size that changes according to the vertical position of the image. Alternatively, by projecting the image onto the plane of the map (birds view); the road should display more consistent textures across the vertical position of the image, which again may improve the results. Nevertheless, the poor separability, combined with the extensive computation time, despite all optimization, leads to the conclusion that the features calculated via the GLCM are currently not feasible for real-time usage.

4.6 Experiment 3 - Road Estimation Performance

The purpose of the third experiment is to investigate the quality of the road estimation algorithm for different combinations of features. The features combined are the ones that showed most promising in the results from the second experiment, i.e. the three color channels, local standard deviation (SDEV), and local entropy (ENTR).
Each feature combination is additionally tested for two choices for the number of Gaussians in the road model, namely one and three. The purpose of this is to roughly investigate the impact of this choice. The combination of features and the number of Gaussians for each run is listed in Table 4.3.

To obtain an understanding of the performance of the road estimation algorithm in different scenarios, the experiment is conducted of four different image sequences. The first recorded sequences from the UGV, which is presented in figure 4.1, should be the easiest to estimate road from, based on the consistency of both the road and the illumination conditions in these images. The second recorded sequence from the UGV, which is presented in figure 4.2, is also used. This sequence is included to evaluate the algorithm's performance on dirt road. Additionally, two external image sequences, created for the research presented in [22], has been incorporated in this experiment. The first of these sequences is

Run #	#Gaussians	Features Combined
1	1	RGB
2	3	
3	1	RGB, SDEV
4	3	
5	1	RGB, ENTR
6	3	
7	1	RGB, ENTR, SDEV
8	3	

Table 4.3: Experiment 2 - List of features combinations and number of Gaussians in the MoG model for the different runs.

recorded under critical lighting conditions, where the road is greatly covered by shadows due to a low sun. This sequence will be referred to as the *Sunny-Shadow* sequence. The second sequence is captured after it has rained, and the road is therefore inconsistently wet. This sequence will be referred to as the *After-Rain* sequence. The resolution of the images in these sequences is a bit higher than the images in the UGV sequences. Although the difference is terms of size is not intentional, it adds another aspect to the experiment, i.e. how the different parameter combinations affect the performance of the algorithm, according to the image resolution. The window sizes used to extract the entropy and standard deviation features are therefore adjusted according to the image resolution. These sizes are listed in Table 4.4.

Sequence	Resolution	ENTR window	SDEV window
UGV seq. 1	422×338	15×15	15×15
UGV seq. 2	422×338	15×15	15×15
Sunny-Shadows	640×480	21×21	21×21
After-Rain	640×480	21×21	21×21

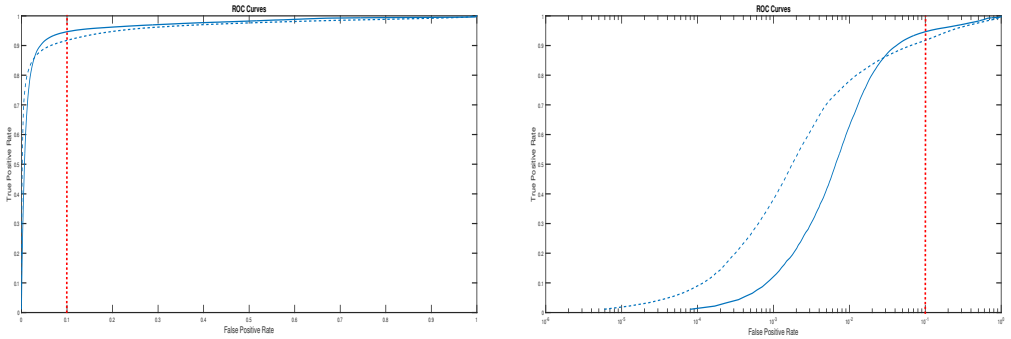
Table 4.4: The table shows the window sizes used to compute the entropy (ENTR) and the standard deviation (SDEV) feature images, for each of the four sequences

Ground truth label images are included for the two image sequences. For the two UGV sequences, labeled images have been manually created. For the third sequence, presented in figure 4.3, manual labeling is difficult since the snow makes the boundary of the road invisible. The third sequence is therefore not included in this experiment. All of the label images are binary, where each pixel is classified as either **road** or **environment**.

The procedure of the experiment starts by running the road estimation algorithm with each of the parameter combinations listed in Table 4.3, on

all of the four sequences. For each run, the probability images produced by the algorithm are stored. Then, using the binary label images, the pixels belonging to **road** and **environment** in all of the probability images are extracted, and sorted into two separate vectors. These two vectors are then used to compute true positive and false positive rate in a *ROC curve*. The results of this procedure will, therefore, be four plots, for each of the four sequences, where each plot contains eight ROC curves, for each of the eight parameter combinations.

Given that a pixel's probability of being a road pixel is represented as a value between 0 and 1, the ROC curve is a graphical plot displaying the true positive rate against the false positive rate as the discrimination threshold decreases from 1 to 0. Since the road estimation algorithm produces probability images and not binary, segmented images, the ROC curves are therefore an appropriate method for performance evaluation.



(a) **(b)**

Figure 4.11: Illustration of the advantage of using mixture of Gaussian modeling, (a) regular ROC curves, (b) x-axis scaled logarithmically. The area previous to the dotted red line, i.e. the area where the false alarm rate is below 0.1, is most interesting part of the ROC curve for evaluating the algorithm.

In the results from Experiment 3, an important thing to notice is that the x-axis in the plot of the ROC Curves is scaled logarithmically. This scaling is done because the important part of the ROC curve is the points to the left, i.e. those points corresponding to higher threshold values. The road estimation algorithm is not perfect, and there will always be a portion of missed detections. In the KITTI leaderboards, the best scoring algorithms have an $F1_{max}$ score above 90%. This score corresponds to the ROC curve reaching a 0.9 true positive rate before it reaches a 0.1 false positive rate. This ratio will, therefore, be the reference for an adequate performance result under this experiment. However, how quickly the true positive rate reaches the 0.9 mark is valuable information, as illustrated in figure 4.11a, the false positive rate in this range might be hard to make out. The bene-

fit of scaling the x-axis logarithmically for this purpose is shown in figure 4.11b. It is also for the logarithmic ROC curve desirable that the true positive rate rises as quickly as possible, but now the ratio between the two rates are more easily discerned.

In addition to the ROC plot, for each sequence, a table listing the area under the logarithmic ROC curves is presented. This value is given to offer a measurable value of the ROC curves. The $F1_{max}$ and the *Precision* score is also listed in these tables, supplying an impression of the performance in relation to the KITTI leaderboards. Though, it must be emphasized that the $F1_{max}$ measure in this experiment cannot directly be compared to the KITTI leaderboards, but is used to give an impression of what can be considered as adequate performance. The $F1_{max}$ and *precision* scores are functions of the true positive and the false positive rate, given in Eq. (4.1) and (4.2).

$$F1_{max} = \frac{2TP}{2TP + FP + FN} \tag{4.1}$$

$$PRE = \frac{TP}{TP + FP} \tag{4.2}$$

4.6.1 Result of Experiment 3

First UGV Sequence

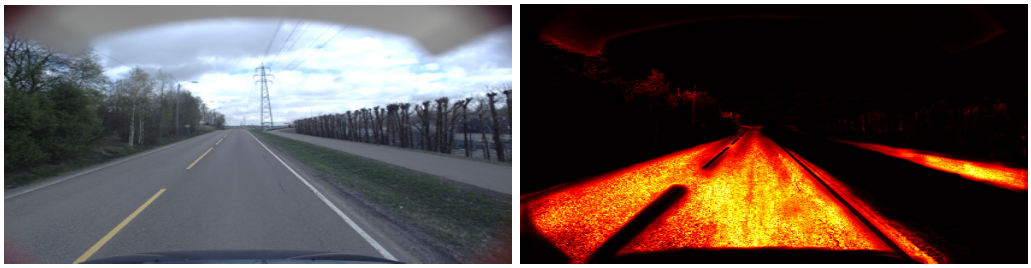


Figure 4.12: Example of an image from the first UGV sequence, and the corresponding computed likelihood image.

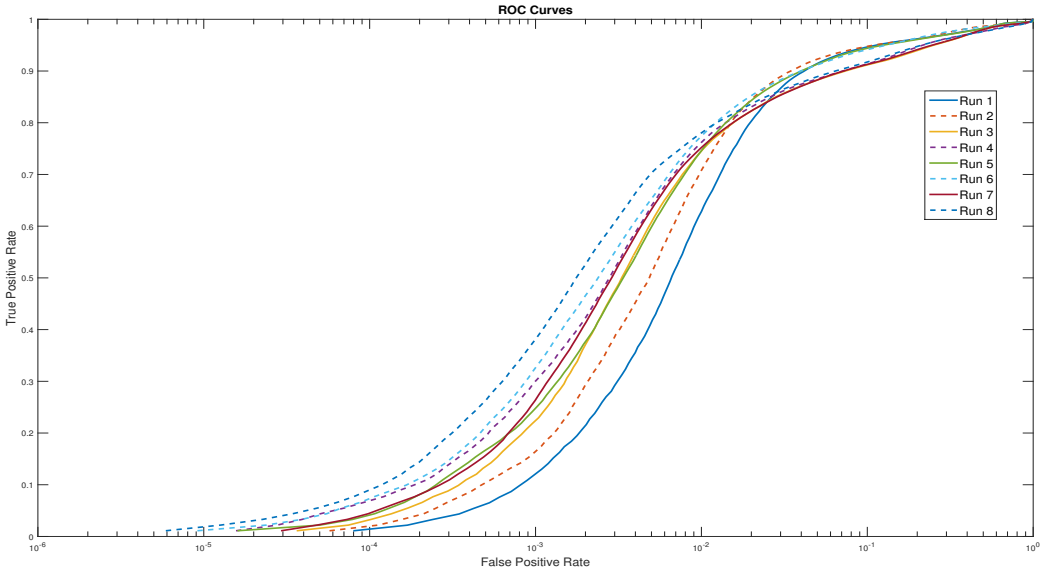


Figure 4.13: ROC curve from experiment 3 conducted on the first UGV sequence, where the x-axis has been scaled logarithmically.

Run #	area under log-ROC	$F1_{max}$	PRE	Runtime
1	5.0558	93.22%	93.71%	25.64fps
2	5.3782	93.57%	94.54%	7.63fps
3	5.5435	91.35%	94.43%	19.60fps
4	5.8302	91.46%	94.16%	5.21fps
5	5.7056	93.10%	93.94%	10.75fps
6	6.0137	92.98%	94.17%	4.50fps
7	5.6830	91.40%	93.79%	9.52fps
8	6.1865	91.71%	94.34%	3.77fps

Table 4.5: Table showing the quantifiable results for the first UGV sequence.

Second UGV Sequence

Figure 4.14: Example of an image from the second UGV sequence, and the corresponding computed likelihood image.

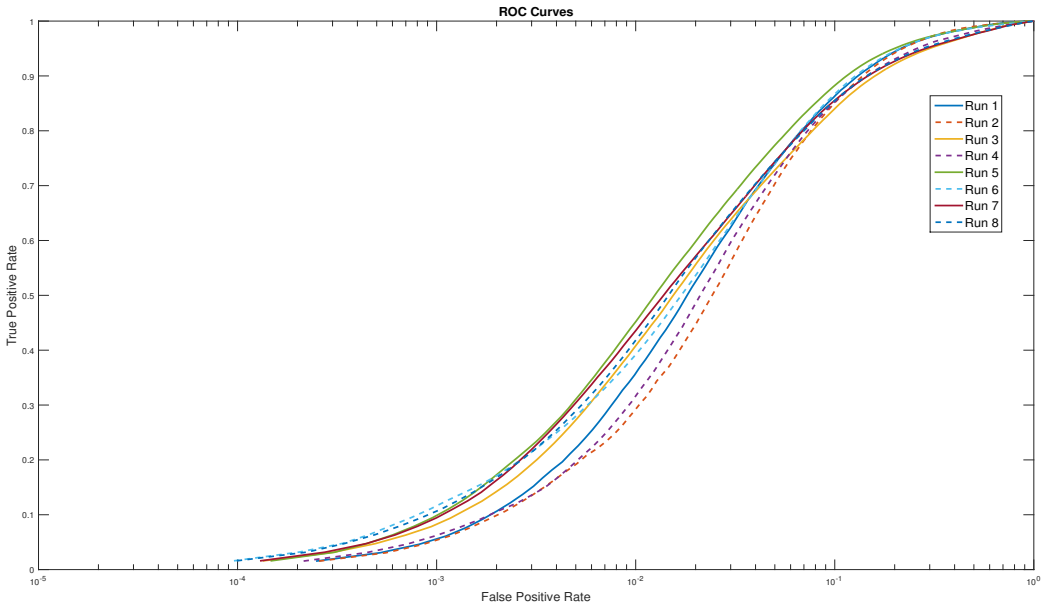


Figure 4.15: ROC curve from experiment 3 conducted on the second UGV sequence, where the x-axis has been scaled logarithmically.

Run #	area under log-ROC	$F1_{max}$	PRE	Runtime
1	4.0977	88.69%	86.60%	25.66fps
2	3.9166	88.39%	85.23%	7.61fps
3	4.2210	87.46%	86.22%	19.19fps
4	3.4516	88.03%	86.26%	5.18fps
5	4.4516	89.45%	87.25%	10.80fps
6	4.3342	88.86%	86.70%	4.49fps
7	4.3337	87.98%	86.75%	9.09fps
8	4.3404	88.04%	86.78%	3.70fps

Table 4.6: Table showing the quantifiable results for the second UGV sequence.

Sunny-Shadows Sequence

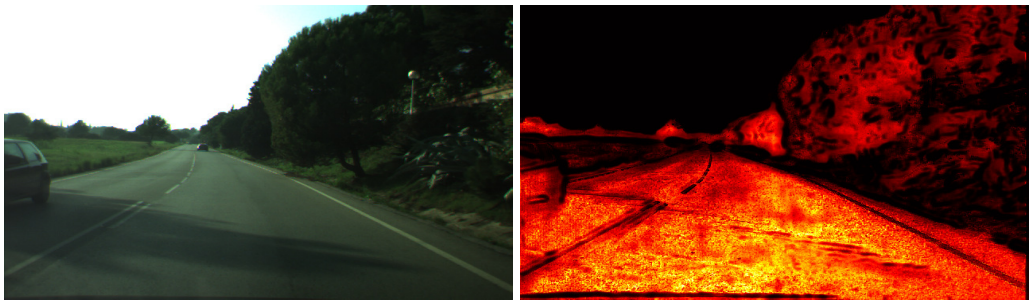


Figure 4.16: Example of an image from the *Sunny-Shadow* sequence, and the corresponding computed likelihood image.

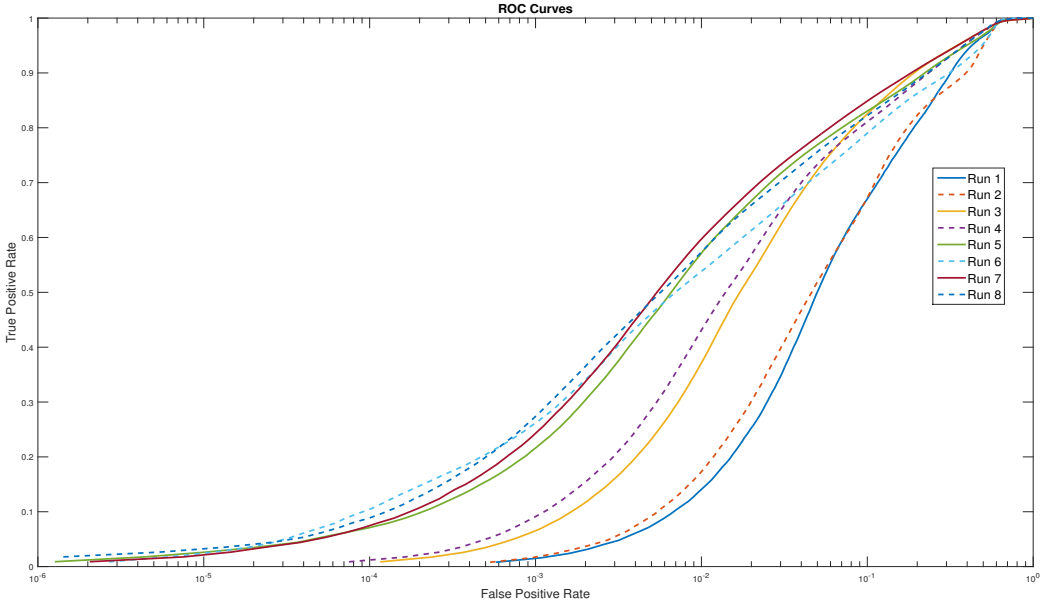


Figure 4.17: log-ROC curve from experiment 3 conducted on the *Sunny-Shadow* sequence.

Run #	area under log-ROC	$F1_{max}$	PRE	Runtime
1	3.0411	81.19%	75.58%	12.78fps
2	3.1357	81.37%	80.29%	3.69fps
3	4.0700	86.36%	85.28%	8.70fps
4	4.2335	85.16%	85.69%	2.28fps
5	5.0969	86.05%	88.47%	4.60fps
6	5.0980	84.11%	85.13%	1.98fps
7	5.2533	87.15%	88.44%	4.65fps
8	5.2709	85.66%	86.95%	1.89fps

Table 4.7: Table showing the quantifiable results for the *Sunny-Shadow* sequence.

After-Rain Sequence

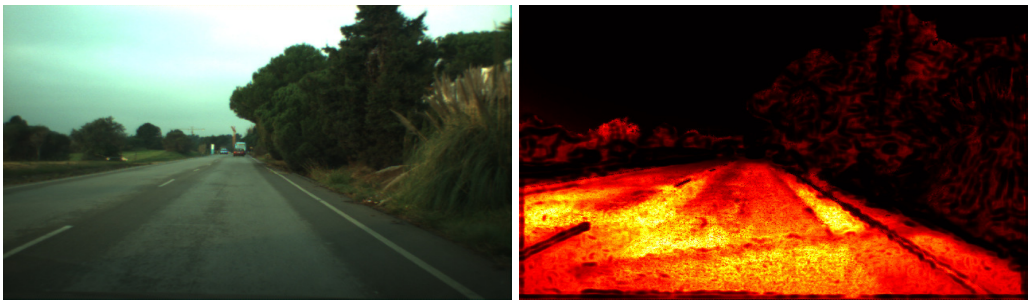


Figure 4.18: Example of an image from the *After-Rain* sequence, and the corresponding computed likelihood image.

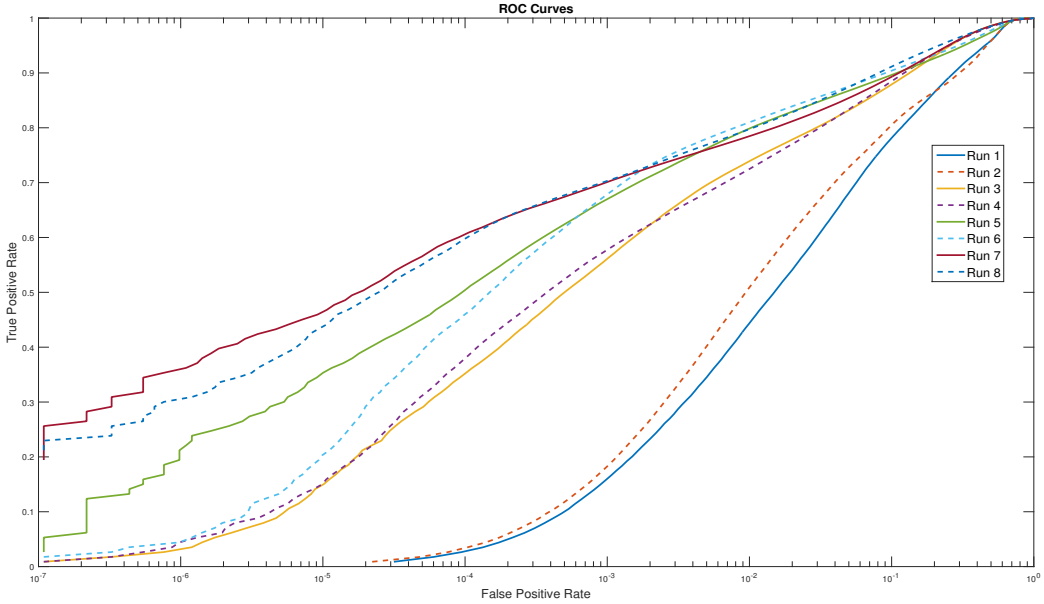


Figure 4.19: log-ROC curve from experiment 3 conducted on the *After-Rain* sequence.

Run #	area under log-ROC	$F1_{max}$	PRE	Runtime
1	4.3770	83.89%	83.64%	12.80fps
2	4.6493	84.79%	86.31%	3.69fps
3	7.4196	88.86%	91.10%	8.69fps
4	7.5251	89.17%	90.41%	2.27fps
5	7.6186	90.48%	94.74%	4.56fps
6	7.5257	90.94%	94.75%	1.96fps
7	7.9173	89.82%	92.84%	4.54fps
8	7.9557	90.94%	93.07%	1.85fps

Table 4.8: Table showing the quantifiable results for the *After-Rain* sequence.

4.6.2 Analysis of the results of Experiment 3

The results from Experiment 3 contain a lot of information and require some effort to analyze.

The results show that the road estimation algorithm performs far worse on the dirt road sequence than on sequences containing tarmac road. It is desirable that the ROC curve increase as fast as possible, i.e. that the true positive rate increases much quicker than the false positive rate. As mentioned, the algorithm is considered as well performing if the true positive rate reaches 0.9 before the false alarm rate reaches 0.1. In the ROC curves in figure 4.15, from the second UGV sequence, for all the runs, the 0.9 true positive rate is not reached before the false alarm rate has reached approximately 0.2, which is considered poor. This is also reflected in the $F1_{max}$ and precision score in Table 4.6, which averages around 88% and 86%, respectively. The $F1_{max}$ and precision score from the Sunny-Shadow sequence are also quite poor, which leads to the conclusion that extreme lighting conditions are still a challenge for the algorithm.

One conclusion which is reasonably clear from the ROC curves is that the inclusion of textural features has a positive effect on the estimation performance, regarding how quickly the true positive rate rises, especially for the sequences containing tarmac road. For all of the sequences, the true positive rate rose faster (with respect to the false positive rate) compared to the first two runs, where the features only included the three color channels. The consequence is especially dramatic in the results from the Sunny-Shadow and After-Rain sequences, which contained tarmac road under tough conditions. These results agree with the prior hypothesis, i.e. that under tough conditions, the texture of the road will be more consistent than the color, especially when computed from the illumination invariant image. The tendency is the same for the first UGV sequence, although not to the same extent. This sequence contains no shadows, and the road appearance is pretty consistent for all images. Consequently, the algorithm yields pretty good results, even when only the color channels were used as a features.

The only exception to improvement as a cause of adding textural features was in run 4 of the second UGV sequence when the local standard deviation feature was added. Table 4.6 shows that run 4 has a lower area under the log-ROC curve than that of run 2, which also used a MoG model with 3 Gaussian densities. This suggests that the local standard deviation feature does not have the same separation capability for dirt road as for tarmac road. The local standard deviation is a measure of energy throughout the image, and it is reasonable to assume that tarmac road has a significantly lower amount of energy than dirt road, since dirt road is less

(a) **(b)**

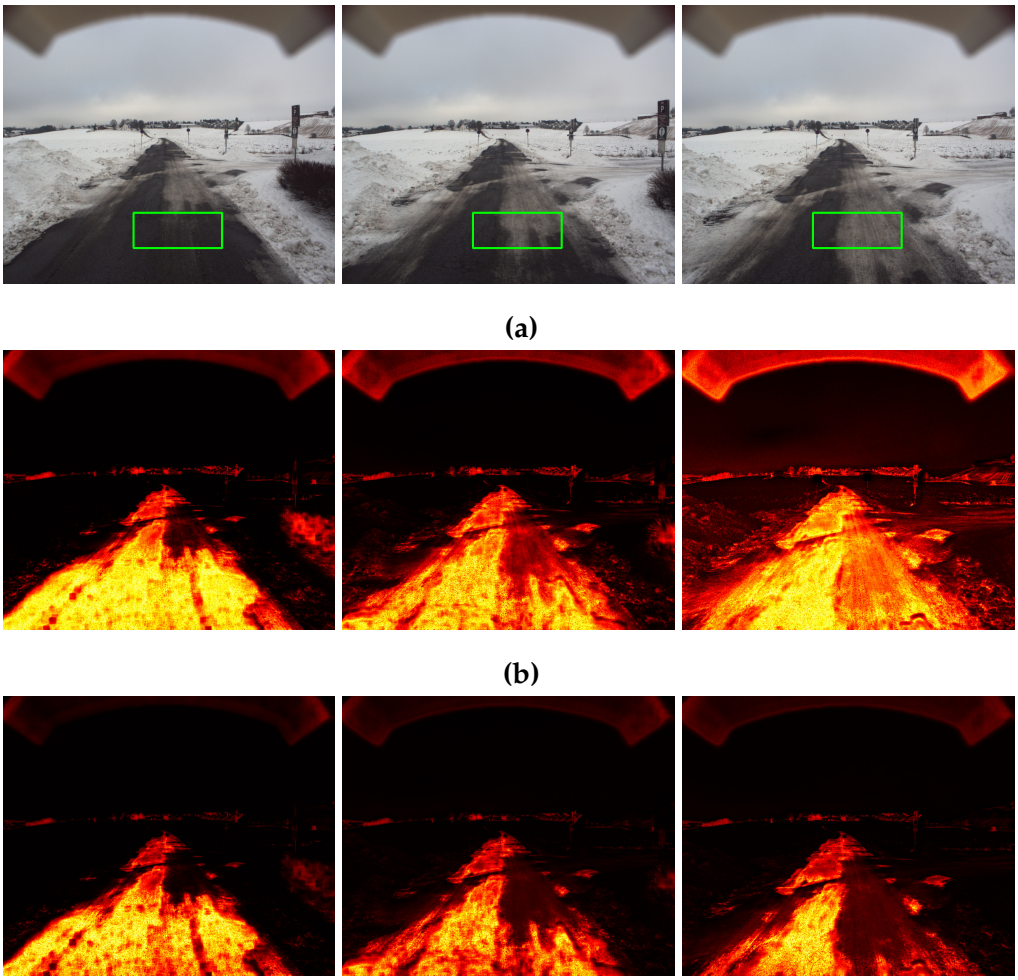
Figure 4.20: Example of the failures of computing the **(b)** a standard deviation feature image from **(a)** an image containing dirt road. The area in the image containing shadows has an impact on the feature-image.

smooth. However, in run 1 and run 3 of this sequence, where the same feature combinations were used in collaborations with a MoG model containing only 1 Gaussian density, the results points to the opposite. The exact reason why these results contradict one another is unclear. In general, dirt road displays a higher degree of variation, in terms of texture, than tarmac road. Therefore, even if the MoG model with 3 Gaussian densities captures the characteristics of this variation better, the road may appear too similar to other objects in the scene, in terms of the specific texture. Therefore, the problem may be caused by the feature's lack of separation ability relative to other scenery objects, rather than with the number of Gaussian densities in the MoG model. Since experiment 2, which regarded feature separability, only was conducted on sequences containing tarmac road, no conclusive opinion on this matter can be made without conducting experiment 2 again for the dirt road sequence. Figure 4.20 shows an example of the local standard deviation computed from a dirt road image from the second UGV sequence. The standard deviation image displays low value for most of the road, but at the area covered by shadow displays higher values. On tarmac road, the effect of the shadow is neutralized by use of the illumination invariant image. However, the method described in 2.1.1 for computing such images are not as invariant to dirt road shadows as for tarmac road. This may be caused by the fact the underlying assumptions which made the simplification of Eq. (2.3) possible, does not apply to the dirt road.

Including local entropy has an undisputed positive effect on the ROC curves for all sequences. Although smaller, this also applies to the results from sequence 2. Combining entropy with color leads to better results than the combination of color and standard deviation, thus implying

that entropy is the better choice of the two features. Yet, as presented in the *Runtime* column of Table 4.5, 4.6, 4.7, and 4.8, entropy is a far more time-consuming feature to extract. So, choosing between the two features, should that be necessary, seems to be a trade-off between estimation quality and computation time. Regardless, including both textural features yields the best overall results through all sequences, especially when the variations in road appearance increase, as it does in the Sunny-Shadow and the After-Rain sequences. The higher resolution in these sequences should also be taken into consideration. Higher resolution will cause better expression of the texture of the objects in the scene, and consequently, cause a larger impact on the performance of the algorithm. This might also be a cause to why the ROC curves from these sequences increase more rapidly when texture is introduced than the ROC curves from the first UGV sequence, which also contains tarmac road.

Regarding the difference between the numbers of Gaussians used to model the road, the results are unclear. In Table 4.5, 4.6, 4.7 and 4.8, the results are ambiguous for each sequence, and for each set of feature combinations. Also, when compared pairwise with the same feature combinations (f. ex: Run 1 vs. Run 2, Run 7 vs. Run 8), the difference is minimal. This leads to the conclusion that the number of Gaussianss has little effect on the overall quality of the road estimation. However, this a quantitative experiment, investigating the quality of the estimation algorithm across several images. The MoG modeling approach, however, is used as an attempt to minimize impact on the road model caused by occasionally appearing elements, such as road markings, manholes, and shadows. Therefore, a quantitative experiment is not well suited to reveal these advantages. On the other hand, it is hard to conduct such an experiment. By definition, such circumstances occur only occasionally, and, therefore, it is difficult to account for them. Still, empirical knowledge has been acquired from several observations of the benefits of choosing more than one Gaussian density under such circumstances.
An example of this is presented in figure 4.21, where three consecutive images from the third recorded sequence from the UGV is shown. The example shows an incident where the extraction window gradually covers more and more of the part of the road covered by snow. Although this area is considered as part of the road, its appearance is more similar to the snow on the side of the road, than the tarmac underneath the snow. Consequently, when these pixels are used as training samples for the model, the model will be "pulled" towards the snow in the feature space. The prior assumption, namely that a MoG model containing several Gaussians is less affected by such issues, is evident in this example. The conclusion is therefore that the road estimation algorithm benefits from modeling the road as several Gaussian densities, although the exact number should be investigated further.



(a)

(b)

(c)

Figure 4.21: The figures above demonstrates the benefit of the MoG density modeling method compared to the single Gaussian density modeling method. **(a)** shows an example of three consecutive images where snow inconsistently covers the road. When the snow area enters the extraction window (marked in green), these pixels becomes part of the model. From the images, it is clear that the impact of the snow pixels are far more drastic on the single Gaussian model in **(b)**, than on the MoG model in **(c)**.

4.7 Experiment 4 - Vanishing Point detection

Experiment 4 seeks to investigate the quality of the algorithm vanishing point detection, described by Kong et. al [37], compared to the computational time of the detection. The accuracy of the algorithm is tested by applying it to a selection of 100 images from two different UGV recording sessions. The first sequence was recorded in the summer of 2015, and the second in January of 2016. The latter sequence, therefore, contains images where snow inconsistently covers the road. The images are selected to achieve a variation of difficulty. The variation is achieved by selecting images with variations in terms of straight and curved road, road with different amount of inclination, cars, and other challenges caused by the scenery.

For each image, the estimated vanishing point is compared to a manually marked vanishing point. The Euclidean distance between these two points is calculated and normalized according to the length of the image diagonal. The normalization is applied to enable pixel distance comparison across image sizes. The performance of the vanishing point estimation is then represented as the mean and variance of normalized distance, as well as the average computation time.

The experiment is conducted for four different image sizes. A Gabor filter bank, containing 36 different orientations at five different scales, is used for all image sizes. The size of the Gabor kernels is chosen according to the size of the images. The image and Gabor kernel sizes tested are listed in Table 4.9, as is the confidence threshold for assigning pixel orientation. The size of the half-circle used in the soft voting scheme is set to $0.35 \times Image\_Diagonal$ for all image sizes.

Image size	Gabor kernel size	conf
800×600	89×89	0.6
400×300	49×49	0.5
200×150	25×25	0.4
100×75	15×15	0.4

Table 4.9: Table showing the Gabor kernel sizes used for the different image sizes in order to compute the pixel orientations. The confidence threshold (conf) for assigning pixel orientations is also listed.

4.7.1 Results of Experiment 4

Image size	Mean error	Variance	Average computation time
800×600	0.013	0.05	508.2sec
400×300	0.024	0.12	23.1sec
200×150	0.076	0.77	3.3sec
100×75	0.101	0.65	0.7sec

Table 4.10: Table showing results of Experiment 4.

4.7.2 Analysis of the results of Experiment 4

The results of Experiment 4 show that with sufficient image resolution, accurate and robust estimations of the vanishing points can be achieved. For the image size 800×600, both the mean error and the variance is low. For the lower resolutions, however, the robustness of the estimate quickly decreases, which is evident from the rapidly increasing variance. Still, the computation time for the 800×600 sized images is dramatically high, which makes it irrelevant for real-time usage. So, a lower resolution must be chosen, and with optimization, a window size somewhere between 400×300 and 200×150 might be fast enough.

(a) **(b)**

Figure 4.22: Example of three accurate vanishing point estimations. The manually marked vanishing point is marked in green, while the estimated is marked in blue.

When viewing the estimated vanishing points in their associated image, the situation which proves most difficult is curved road, as shown in figure 4.23a and 4.23a. The problem arises from the fact that the score in the voting scheme is based on a linear relationship between the potential vanishing point and the pixels in the half-circle. In images containing

(a) **(b)**

Figure 4.23: Example of three inaccurate vanishing point estimations. The manually marked vanishing point is marked in green, while the estimated is marked in blue.

curved road, there are multiple points which have a strong linear relationship with points from both the left and the right road edge. In addition, the pixels within the area which is closest to the vehicle, i.e. the lower part of the image, displays a more confident orientation values. The reason for this is simply that this area is closer to the camera, and therefore, the geometric details are much better. This leads to a favoring of points which are lower in the image, i.e. closer to the vehicle.

Kong et. al improves the vanishing point estimation in a second step. After the initial estimation, the left road edge is detected. This edge is then split into regularly spaced segments, and redefined according to the OCR criterion, thus allowing for curvature in the road edge. The vanishing point is redefined as the ending point of the new left road edge.
However, this improvement step only works if the initial estimation of the vanishing point is reasonably accurate. It is therefore concluded not conduct any experiment regarding the second part of this algorithm, i.e. the detection of road edges, before a more reliable implementation is found.

The conclusion of this experiment is that the current implementation is not adequate for reliable real-time road estimation. The results are promising, and with further work, it might help to improve the existing texture and color based road estimation algorithm.

Chapter 5

Discussion

This chapter concludes the thesis, beginning with a quick discussion of the overall results from the experiments. Then, some possibilities for future work is presented, and lastly a final, conclusive summary is presented.

5.1 Collective Discussion of the Experiment Results

The work with the road estimation algorithm is still not widely explored. Until now, it has only been tested on video sequences recorded while a driver has controlled the vehicle. However, the reliability of the algorithm under real-time, autonomous driving, is currently under investigation. Some early tests suggests that the algorithm is reliable for autonomous driving as long as the amount of shadows is small. This agrees with the results of Experiment 3, which revealed that the algorithm performs well under uniform light conditions, but still struggles to deal with shadows and backlight. For autonomous driving, an error like this would be fatal for the path planning, and will have to be addressed in the long term, for a properly working navigation system. It is still poorly investigated how well all subsystems at work on the UGV collaborates, and what yields good results for the road estimation algorithm, might be a disadvantage for other parts of the system. This is of course pure guesswork but is still something to keep in mind for future work.

The testing done until now has been a mix of estimation quality and efficiency performance. Achieving accurate and robust estimation has been the primary aim of the conducted work, but the real-time criterion has always been kept in mind. The combinations of features used in Experiment 3 are currently those shown to perform adequately in terms of this real-time criterion, and are therefore the ones implemented in the road estimation algorithm.

The experiments described in chapter 4 was conducted on an office computer with limited processing capacity, and without any specialized graphics processor, so there is likely more that can be done to achieve faster computation speed. Some optimization has been exercised, but this aspect is still to be further improved. The UGV is a long term project, so one should also account for future technology as well. Therefore, even though Experiment 2 showed that the GLCM features currently proved too slow, it might be possible to utilize them with further optimization and/or better hardware. As discussed in 4.5.2, this will also require the images to be projected onto the map plane.

5.2 Future Work

In this section, some of the main topics regarding future work will be discussed.

5.2.1 Extracting training samples

The largest weakness in the current implementation of the algorithm is the method for extracting training samples from the image. As mentioned in section 3.1.2, the current algorithm does this by extracting a square window at the bottom of the image, assuming that only pixels belonging to the road are contained in this window. Nevertheless, if the window covers objects which are not a part of the road, the model of the road will be corrupt. The MoG model, along with the 10% learning rate, is a method to avoid this problem for short occurrences of such objects. Still, should the window be exposed to other objects over a longer period, the MoG density model will slowly converge towards the parameters of these objects. This problem often occurs while the UGV is driving through sharp turns.

To solve this problem, a better method of extracting the training samples has to be applied. The Stanley vehicle, as previously mentioned, uses LIDAR data to estimate the nearby road [6]. This estimate is used to extract the training samples from the image, thus enabling Stanley to estimate road further away in the scenery than possible from only the LIDAR data. Another solution may be to use other prior probabilities from the other scene analysis methods, such as the stereo vision, as proposed in [41]. However, since all of these methods are currently under research, this thesis will not speculate in too much detail the matter of which these methods could best supplement the road estimation algorithm.

5.2.2 Improving the Illumination Invariant Images

As discussed, the variation of road appearance is almost endless, and it is difficult to identify all problems and challenges that may appear by only testing the algorithm on previously recorded sequences. The results of Experiment 3 showed that the algorithm, in general, performs well in circumstances where the road is uniform. In both the first UGV sequence, where the road was dry tarmac, and the After-Rain sequence, where the road was wet tarmac, the estimation achieved a true positive rate above 0.9 at false alarm rate 0.1, which is a pretty decent overall result. However, in the second UGV sequence, where the road was dirt and gravel, and in the Sunny-Shadow sequence, where there was a lot of shadows and backlight, the results were considerably poorer. Hence, shadows and extreme illumination conditions still cause problems. The performance especially decrease in situations where shadows are cast over dirt road. The reflective property of dirt road is not as consistent as that of tarmac road, and the assumptions and simplifications that allowed the illumination invariant to be computed according to Eq. (2.3) are therefore less valid. Although the first milestone of this project is to achieve autonomous driving on tarmac road, in long term, the UGV should be able to traverse dirt and unstructured road as well.
A solution to this problem may be to change the approach for computing the illumination invariant image. Instead of computing the illumination invariant image based on prior information about the peak spectral response of the image sensor, Álvarez et. al proposes a method which, in extension, is based on calibrating the spectral properties of the current scenery, based on a set of training samples from each image [22]. Although this method is computationally more complex, it may be more robust than the simplified version currently implemented in the road estimation algorithm. This method might also contribute to making the algorithm even more robust to shadows and extreme light conditions.

5.2.3 Increasing estimation rate

There is still much that can be done in order to further speed up the road estimation algorithm. The biggest potential for improvement is to execute the independent operations in the algorithm in separate threads. Recall figure 3.1, where the different operations of the road estimation algorithm are illustrated. Both the model estimation and the road estimation processes are dependent on the feature image computation. However, if the model is allowed to be updated at a different rate than at that which is used by the road estimation process; these two processes can be executed in two separate threads. Since the computation time of the EM algorithm will no longer influence the road estimation rate, the restriction on the number of Gaussians the road is modeled as can, therefore, be set to a more generous

number without having to account for the road estimation rate as much. This also applies to the number of iterations the EM algorithm is allowed before termination.

Another potential for increasing the overall estimation rate is to further optimize the methods for extracting feature images. A particularly promising approach is presented in [42], where the histogram of local neighborhoods are computed using a technique they have referred to as *integral histogram*. Unfortunately, further investigation of this method has not been included in this thesis, but it shows great promise for speeding up the extraction of the local entropy image. Also, the techniques proposed in [42] might also be used to further improve the efficiency of the GLCM based feature extraction methods described in 13.

5.2.4 Vanishing Point Detection

The results of Experiment 4 suggested that the current implementation of the vanishing point detection algorithm is infeasible for real-time usage. The required resolution for reliable estimations causes the algorithm to be too slow. As mentioned in 4.7.2, in fairness to the algorithm, the second step of the algorithm should be tested before a conclusion about the accuracy of the estimation is made. Unfortunately, this has not been included in this thesis.
However, in the final months of the work with this thesis, a new approach for detecting vanishing points was published [43]. The proposed method includes a particle filter that allows previously estimated vanishing point to influence the search for new vanishing points. The underlying assumption is that, between two consecutive images, the vanishing point is usually in the same general area of the image. Based on this, the proposed method normalizes the size of the image a fixed size of 61×81 pixels. Although the small image size may cause poor results for single image estimation, over time the estimation will be sufficiently accurate and reliable. In the article, the authors have compared this approach to that of [37] (and two others), and the results are impressive. Compared to [37], which had a normalized distance error of 0.0316, the proposed method of [43] only had a normalized distance error of 0.0189. Compared to the results of Experiment 4, this is almost as good as what was achieved when the image resolution was set to 800×400 pixels. In Experiment 4, this image resolution caused an average computation time of 508.2 seconds, while the method proposed by [43] only needs an average computation time of 0.027 seconds to achieve almost as good results. To conclude, the improvement of this approach is significant, and it shows great promise for improving the existing texture and color based road estimation algorithm.

5.3 Conclusion

This thesis presents a method for detecting road in a local scenery, using computer vision and machine learning techniques. The algorithm utilizes a Mixture of Gaussian distribution to model the road, which is learned and adjusted in real-time while the algorithm is provided with images. The features used are a combination of the three color channels Red, Green, and Blue, and the textural features *local standard deviation* and *local entropy*. The textural features are created from illumination invariant images, which in turn are computed from the RGB color image given as input to the algorithm.

The performance of the algorithm is tested for four different scenarios, with four different combinations of the features, and for two choices for the number of Gaussians in the MoG model. The performances has proved sufficient under homogeneous illumination conditions, i.e. if the amount of backlight and shadows are low. Although it is not directly comparable, under such conditions, the algorithm was shown to perform almost as good the top algorithms on the KITTI leaderboard for road/lane detection. Still, despite the efforts to make the algorithm more invariant to the illumination variations in road scenery, uneven illumination remains a problem. The performance also decreased when the algorithm was tested on a sequence containing dirt road. The combination of greater variation in road texture, and the fact that the illumination invariant image was less resistant to the effects of shadows on dirt road, was the reason for the lower performance in this scenario.

The evaluation of the algorithm's performance supported the prior hypothesis, namely that the inclusion of textural features had an increasingly positive effect as the scenery conditions got more difficult. Still, including these features decreased the rate of estimation. Furthermore, the degree of improvement was highly dependent on the resolution of the input image.

Bibliography

[1] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[2] Oliver J Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 14:15, 2007.

[3] "'phantom auto' will tour city". *The Milwaukee Sentinel, Google news archive*, 1926.

[4] Richard Wallace, Anthony Stentz, Charles E Thorpe, Hans Maravec, William Whittaker, and Takeo Kanade. First results in robot road-following. In *IJCAI*, pages 1089–1095. Citeseer, 1985.

[5] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The 2005 DARPA grand challenge: the great robot race*, volume 36. Springer Science & Business Media, 2007.

[6] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[7] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[8] Chris Urmson, J Andrew Bagnell, Christopher R Baker, Martial Hebert, Alonzo Kelly, Raj Rajkumar, Paul E Rybski, Sebastian Scherer, Reid Simmons, Sanjiv Singh, et al. Tartan racing: A multi-modal approach to the darpa urban challenge. 2007.

[9] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[10] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

[11] David S Hall. High definition lidar system, July 1 2014. US Patent 8,767,190.

[12] CHRIS URMSON. Progress in self-driving vehicles. *FRONTIERS OF*, 2015.

[13] J Deng, W Dong, R Socher, LJ Li, K Li, and L Fei-Fei. Imagenet: A large-scale hierarchical image database (2009). CVPR.

[14] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

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

[16] Rahul Mohan. Deep deconvolutional networks for scene parsing. *arXiv preprint arXiv:1411.4101*, 2014.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

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

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[20] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *Available electronically at http://yann. lecun. com/exdb/mnist*, 2012.

[21] Graham D Finlayson and Steven D Hordley. Color constancy at a pixel. *JOSA A*, 18(2):253–264, 2001.

[22] José M Álvarez and Antonio M Ĺopez. Road detection based on illuminant invariance. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1):184–193, 2011.

[23] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, 2014.

[24] Colin McManus, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 901–906. IEEE, 2014.

[25] M Bishop Christopher. Pattern recognition and machine learning. *Company New York*, 16(4):049901, 2006.

[26] Roland Wilson and Michael Spann. *Image segmentation and uncertainty*. John Wiley & Sons, Inc., 1988.

[27] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621, 1973.

[28] Sarah H Peckinpaugh. An improved method for computing gray-level cooccurrence matrix based texture measures. *CVGIP: Graphical Models and Image Processing*, 53(6):574–580, 1991.

[29] Rafael C Gonzalez and Richard E Woods. *Digital image processing*. Prentice hall, 2002.

[30] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[31] Simon JD Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[32] John Rice. *Mathematical statistics and data analysis*. Nelson Education, 2006.

[33] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[34] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[35] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, 1997.

[36] Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):309–318, 2004.

[37] Hui Kong, Jean-Yves Audibert, and Jean Ponce. General road detection from a single image. *Image Processing, IEEE Transactions on*, 19(8):2211–2220, 2010.

[38] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[39] Thomas S Huang, George J Yang, and Gregory Y Tang. A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(1):13–18, 1979.

[40] Ben Upcroft, Colin McManus, Winston Churchill, Will Maddern, and Paul Newman. Lighting invariant urban street classification. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1712–1718. IEEE, 2014.

[41] Kurt Konolige, Motilal Agrawal, Robert C Bolles, Cregg Cowan, Martin Fischler, and Brian Gerkey. Outdoor mapping and navigation using stereo vision. In *Experimental Robotics*, pages 179–190. Springer, 2008.

[42] Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829–836. IEEE, 2005.

[43] J. Shi, J. Wang, and F. Fu. Fast and robust vanishing point detection for unstructured road following. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):970–979, April 2016.