Universität Passau

Institut für Softwaresysteme in technischen Anwendungen der Informatik
(FORWISS Passau)
Fakultät für Informatik und Mathematik

Innstraße 33
94032 Passau
`http://www.fim.uni-passau.de`

Master's Thesis

Learning Roads From Images

Montassar AJAM

Matriculation Number: 75997

26.01.2017

Supervised by
Prof. Dr. Tomas Sauer

Forwiss Universität Passau
Innstraße 43 D-94032 Passau
Deutschland

First of all I would like to express my deep gratitude to Prof. Dr. Tomas Sauer my master thesis supervisor and the Director of the Institute of Software Systems in Technical Applications of Computer Science (FORWISS Passau) for giving me the opportunity to carry out this Thesis, for his valuable guidance and for many insightful conversations during the development of the ideas.

Also, I like to thank Dr. Alexander zimmernann for introducing me to the topic as well as for the support on the way and of course for sharing his precious time during the process of learning.

Special thanks to Mr. Steven Kienast for providing me with the tools that I needed to choose the right direction and successfully complete my thesis.

Finally, I owe more than thanks to my loved ones, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Passau, 26.01.2017

. . . .

Abstract

Road detection and tracking methods are the state of the art in present intelligent transportation systems and intelligent vehicle applications. It is however very challenging since the road is in an outdoor scenario imaged from a moving platform.

This master thesis focus on active safety for the automotive sector, or more specifically, on the problem of feature extraction and data classification for road detection applications developed to enhance driver assistance systems for semi-autonomous or self-guided vehicles.

In this thesis, we provide a novel feature vector construction approach based on blockwise Discrete Cosine Transform texture, color and position features which are then concatenated, Labeled and used to train some commonly used statistical binary classifiers, e.g Support Vector Machine. Finally, we elaborate the results and evaluate these classifiers.

In conclusion, the results of this study showed that the combination of these different features have led to a successful detection of road regions and were well suited for this kind of tasks.

Contents

List of Figures

List of Tables

1 Introduction

This thesis was written at the Institute of Software Systems in Technical Applications of Computer Science (FORWISS) which is a research institute of the Faculty of Computer Science and Mathematics at the University of Passau, that provides expertise in image and signal processing, 3D-printing applications, optical measurement algorithms as well as in driver assistance systems.

Intelligent Transport Systems, Advanced Driver Assistance Systems (ADAS), Vision-based driver assistance applications are within the main topics of this thesis.

1.1 Motivation

More than a million people die each year on the world's roads with an estimated road traffic death rate of 4.3 per 100.000 population in Germany and 24.4 per the same population in Tunisia ([TPI13] Global status on road report, 2015, Table A2). Besides, the cost of dealing with the economic consequences of these road traffic crashes reaches billions of dollars ([JTA00] Estimating Global Road Fatalities, 2000, Table 11).

Therefore, Strategies are formulated to reduce road traffic accidents, some of which focus on educating and training road users along with enforcing road traffic rules ([ETS14], p.7-8), some others, consider enhancing road infrastructures to make them safer ([ETS14], p.8). While emerging strategies are paying more attention to promoting vehicle safety and the use of modern technologies such as intelligent transport system (ITS) tools.

Hence, The automotive industry was directly affected by this new trend and invested into research focused on vehicle safety. The ultimate goal is to start manufacturing semi autonomous to fully autonomous "crash-less" cars with build in Advanced Driver Assistant Systems (ADAS) which would provide both comfort and security to drivers.

One of these systems being developed, consists of detecting and tracking road regions for autonomous driving. Its task is to analyze the images provided by a front camera, and then to find the road regions with a sensitive precision. This model rely on image and video processing techniques as well as on recent advances in computer vision field. But none of the previous conceived methods has reached perfect results.

For instance, the car manufacturer Tesla Motors was a subject of a U.S federal investigation [MHC16] during the progress of this thesis due to a fatal crash of a driver operating a Tesla Model S sedan car with its Autopilot system engaged. The car's Autopilot "Model S Software Version 7.0" which rolled out worldwide early this year is one of the most advanced autopilot systems in the world and includes autopilot features that help steer the car on the highway using a front-facing camera next to the rear-view mirror, the autopilot has faced questions as it has failed to distinguish between a white truck and a bright sky[Lev16] (`http://goo.gl/f55PJ7`).

1.2 Objective

This thesis describes a novel approach to detecting and tracking road regions. The method utilizes data classification techniques to solve the problem of distinguishing between road and non-road regions in a precise manner.

The training data for this Classification problem will be constructed from texture, color, position features which are extracted from pixel blocks of the input images which are collected by a camera installed in the front-view of a car.

We measure afterward these classifier's performance under different parameter choices then we adopt those that provide minimal errors and we demonstrate these optimal choices.

1.3 Scope

The implementation and testing are based on MATLAB R2016a. For building our algorithm , we used Image processing , Computer Vision System , Statistics and Machine Learning toolboxes. We used as well Imagemagick Software for image displaying reasons. and a labelling graphical user interface developed at the University of Passau by Mr. Steven Kiena, to Label the images. After designing the model, we use the "MATLAB Classification Learner tool" to train our model to classify data using supervised machine learning.

1.4 Definition of Terms

For the purpose of Clarification, the important terms used in this thesis have been defined.

Advanced Driver Assistance Systems (ADAS) : Is the collection of numerous intelligent units integrated in the vehicle itself. All this units perform different tasks and assist the human driver in driving. One may choose as many features as possible whereas the features may themselves display a range of intelligence,autonomy and monetary costs [Kal16] .

Digital image processing : is the use of computer algorithms to perform mathematical operations on digital images, in order to get an enhanced image or to extract some useful information from it.

Image Analysis : refers to processing an image where the ultimate goal is not to enhance or otherwise alter its appearance but instead to extract meaningful information about its contents.

Computer Vision : tackles the problem of engineering artificial visual systems capable of somehow comprehending and interpreting our real, three-dimensional world.

Machine Learning : It is a mechanism for pattern search and building intelligence into a machine to be able to learn, implying that it will be able to do better in the future from its own experience.

1.5 Outline

This thesis is divided into an introductory part, five main chapters listed bellow and a conclusion.

Chapter 2 : State of The Art

This chapter studies the previous researches in this particular field, i.e Road detection using computer vision techniques, and describes the environments we have used in this thesis and the acquired background which helped us to achieve our goals.

Chapter 3 : Method

The general methods followed in the Thesis as well as the theoretical foundations behind these methods are explained in this Chapter. We have first described the data we have used such as the image sequences fed to the classifier.

Chapter 4 : Implementation

This chapter describes the way the implementation method is done, starting from the Labelling process and the used tools , and ending with specifying the used toolboxes in the implementation of our method.

Chapter 5 : Evaluation

In this chapter, we have showed the results we have got after the construction of the feature vectors and the training of the built models, the results represent the performance measurements of the constructed models.

Chapter 6 : Future Applications and Developments

In this chapter, we describe the problems that occurred and we give as well a small description about possible solutions to overvome such problems in order to bild in the future better systems.

2 State of the Art

.

2.1 Computer Vision Systems

Computer vision is a strongly multidisciplinary research area, and computer vision-based systems are becoming ubiquitous , integrating into embedded systems world including, games devices, smart-phones and cameras, advanced driver assistance systems, robots used in manufacturing processes. Computer vision is the automatic analysis of images and videos by computers and has many applications in industry, for example, biometric security checks, automatic reading of license plate , guiding robots for complex product manufacturing and inspecting product or industrial process.

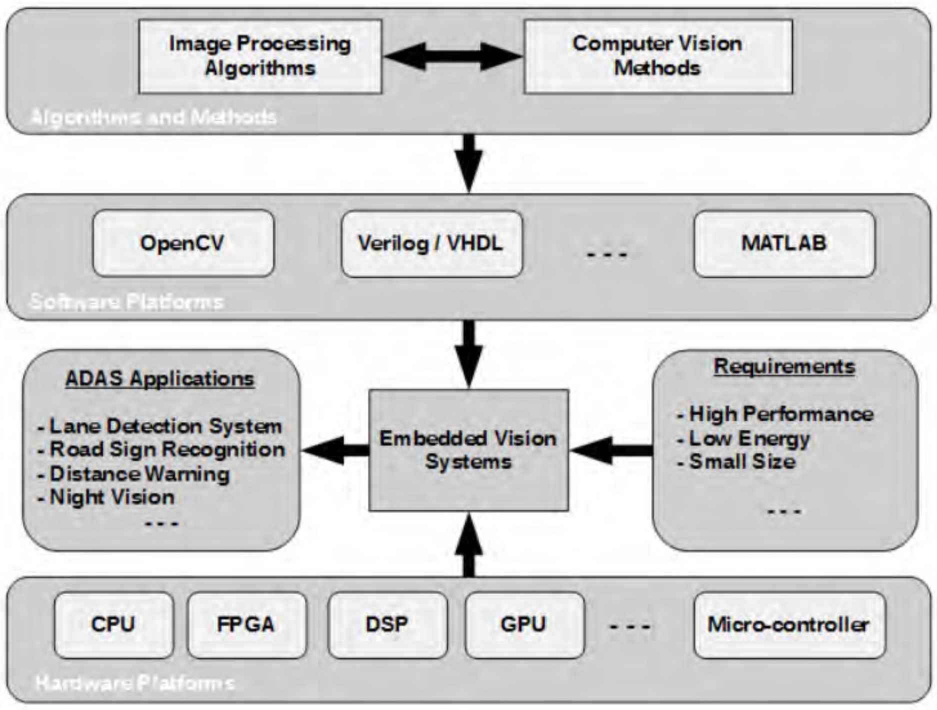


Figure 2.1: Embedded Vision System architecture [Kum16]

A basic computer vision system consists of a camera attached to a computer. The camera can be either color or grayscale depending on the intended use. Many vision systems utilize high-quality speciality cameras which have a bigger pixel density concentration in the images they produce, thereby capturing more detailed information. Other vision systems use cameras with low noise which produce a cleaner image, where all or most information in the image accurately represent the information in the world that they have captured. A high frame rate cameras are a type of cameras used also in vision systems and which are able to capture images at a very fast rate, often greater than 30 times per second.

The optical system used in a typical computer vision system captures images and transfers it to the computer as a series of pixels. Each pixel has a red, green, and blue value each ranging between 0 and 255 or between 0 and 65536 for the deep color images.

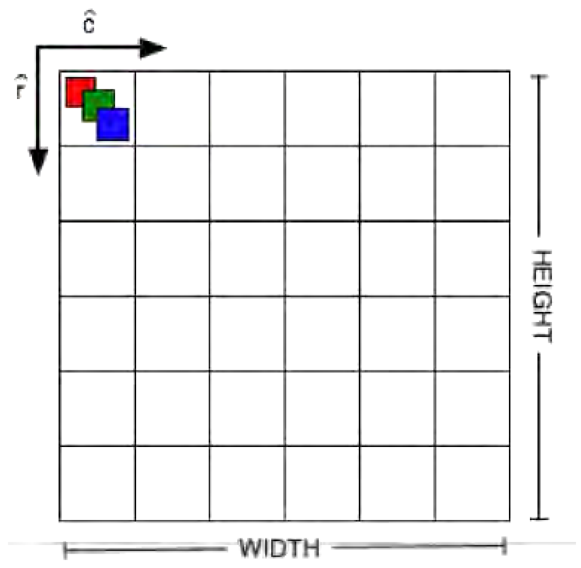


Figure 2.2: The Spatial Coordinate System

The Figure 2.2 shows an image frame from a camera. $\hat{r}$ and $\hat{c}$ are the axes of the image frame originating at the upper left corner of the image. Pixels in the image are most commonly indexed along the $\hat{r}$ and $\hat{c}$ axes corresponding to the row and column of the image.

Most of common computer vision applications consider a real-time vision pipeline. It covers inter alia : applications for autonomous navigation applied to vehicles in order to enhance their safety. This involves methods that imply the presence of a camera mounted on these cars to detect the road regions, traffic lights, road boundaries. The design and development methodology for such real-time embedded vision systems need to consider the involvement of software and hardware components, associated interconnection between input camera module to processing unit and memory size. The following paragraph gives a short overview of embedded systems real-time vision pipeline consideration and inter-relation between different components of embedded vision systems used in ADAS as our main focus in this thesis lies on this particular type of applications.

The strong and growing automotive market for ADAS using embedded computer vision systems comes after an advancement of powerful hardware namely in the form of fast and real-time processing, human machine interfaces and sufficiently large memory.

A general architecture of embedded vision-based advanced driver assistance applications can be described as shown in Figure(2.1). The components involved in a general architecture of embedded vision systems, shown in Figure(2.1), are designed taking into account the current literature research and review in the area of image processing, computer vision and embedded systems implemented on vision-based architecture and platform. In general image processing algorithms are used in combination of computer vision methods using different hardware and software modules including MATLAB, OpenCV and VHDL. These vision based applications have been designed and developed on different platforms including desktop CPU processor (ARM CPU processor), FPGA, GPU based systems, DSP (Blackfin 561 DSP Processor) and micro-controller based processor. The applications designed and developed under embedded vision systems methodologies has different requirement shown in the right side of the Figure(2.1) [Kum16].

2.2 Matlab

MATLAB, short for MAtrix LAborotory, is a numerical computing environment as well as a high-level programming language. It performs many computationally intensive tasks with considerable higher speed than other programming languages.

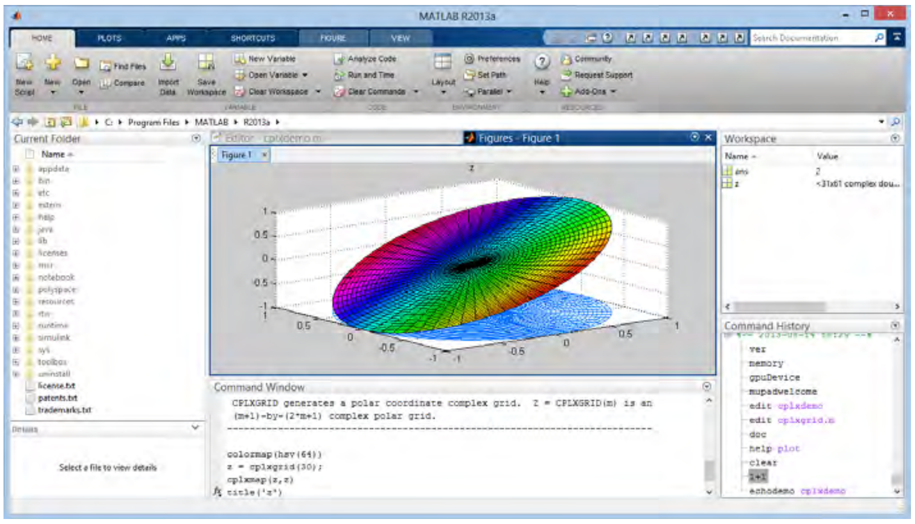
MATLAB is used in areas like signal and image processing, communications, control design, test and measurement, financial modeling and analysis and computational biology. Adds-on toolboxes (collection of spacial-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas. Figure(2.3) shows a screen-shot from the MATLAB environment.

The toolboxes of current interest for this thesis are Statistics and Machine Learning Toolbox and Neural Network Toolbox

2.2.1 Statistics and Machine Learning Toolbox

Overview

The Statistics and Machine Learning Toolbox provides functions and apps to describe, analyze, and model data. One can use descriptive statistics and plots for exploratory data analysis, fit probability distributions to data, generate random numbers for Monte



```
CPLXGRID generates a polar coordinate complex grid.  Z = CPLXGRID(m) is an
(m+1)-by-(2*m+1) complex polar grid.
--------------------------------------------------------------------------

colormap(hsv(64))
z = cplxgrid(30);
cplxmap(z,z)
title('z')
```

```
ver
memory
gpuDevice
mupadwelcome
edit cplxdemo
edit cplxgrid.m
doc
help plot
clear
1+1
echodemo cplxdemo
```

Figure 2.3: MATLAB Graphical User Interface

Carlo simulations, and perform hypothesis tests. Regression and classification algorithms are also included so that users can draw inferences from data and build predictive models using these algorithms [PT16].

For multidimensional data analysis, Statistics and Machine Learning Toolbox provides feature selection, stepwise regression, principal component analysis(PCA), regularization, and other dimensionality reduction methods that let users able to identify variables or features that impact a particular model.

The toolbox provides also supervised and unsupervised machine learning algorithms, including support vector machines (SVMs), boosted and bagged decision trees, k-nearest neighbor, k-means, k-medoids, hierarchical clustering, Gaussian mixture models, and hidden Markov models. Many of the statistics and machine learning algorithms can be used for computations on data sets that are too big to be stored in memory [PT16].

Key Features according to [PT16]

Exploratory Data Analysis: Statistics and Machine Learning Toolbox provides multiple ways to explore data: statistical plotting with interactive graphics, algorithms for cluster analysis, and descriptive statistics for large data sets.

Dimensionality Reduction: Statistics and Machine Learning Toolbox provides algorithms and functions for reducing the dimensionality of your data sets. Dimensionality reduction is an important step in your data analysis because it can help improve model accuracy and performance, improve interpretability, and prevent overfitting. You can perform feature transformation and feature selection, and explore relationships between variables using visualization techniques, such as scatter plot matrices and classical multidimensional scaling.

Machine Learning: Machine learning algorithms use computational methods to "learn" information directly from data without assuming a predetermined equation as a model. Statistics and Machine Learning Toolbox provides methods for performing supervised and unsupervised machine learning.

Regression and Analysis of variance: Using regression techniques, you can model a continuous response variable as a function of one or more predictors. Statistics and Machine Learning Toolbox offers a variety of regression algorithms, including linear regression, generalized linear models, nonlinear regression, and mixed-effects models. Analysis of variance (ANOVA) enables you to assign sample variance to different sources and determine whether the variation arises within or among different population groups. Statistics and Machine Learning Toolbox includes these ANOVA algorithms and related techniques

Probability Distributions: Statistics and Machine Learning Toolbox provides functions and an app to work with parametric and nonparametric probability distributions. With these tools, you can fit continuous and discrete distributions, use statistical plots to evaluate goodness-of-fit, compute probability density functions and cumulative distribution functions, and generate random and quasi-random numbers from probability distributions.

Big Data, Parallel Computing, and Code Generation: Statistics and Machine Learning Toolbox allows its users to perform computationally demanding and data-intensive statistical analysis and to generate portable and readable C code for select functions for classification, regression, clustering, descriptive statistics, and probability distributions

2.2.2 Neural Network Toolbox

Overview

Neural Network Toolbox provides algorithms, functions, and apps to create, train, visualize, and simulate neural networks. You can perform classification, regression, clustering, dimensionality reduction, time-series forecasting, and dynamic system modeling and control.

The toolbox includes convolutional neural network and autoencoder deep learning algorithms for image classification and feature learning tasks. To speed up training of large data sets, you can distribute computations and data across multicore processors, GPUs, and computer clusters using Parallel Computing Toolbox [DBH06].

Key Features according to [DBH06]

Classification, Regression, and Clustering: Neural Network Toolbox includes command line functions and apps for creating, training, and simulating neural networks. The apps make it easy to develop neural networks for tasks such as regression (including time-series regression), classification, and clustering. After creating your networks in these tools, you can automatically generate MATLAB code to capture your work and automate tasks.

Deep Learning: Deep learning algorithms can learn discriminative features directly from data such as images, text, and signals. These algorithms can be used to build highly accurate classifiers when trained on large labeled training datasets. Neural Network Toolbox supports training convolutional neural networks and autoencoder deep learning algorithms for image classification and feature learning tasks.

Network Architectures: Neural Network Toolbox supports a variety of supervised and unsupervised network architectures. With the toolbox's modular approach to building networks, you can develop custom network architectures for your specific problem. You can view the network architecture including all inputs, layers, outputs, and interconnections.

2.3 Color Analysis

2.3.1 Color Spaces

Each pixel of the image can be represented as a point in a 3D color space. Commonly used color spaces include RGB, Munsell, CIE L*a*b*, CIE L*u*v*, HSV (or HSL, HSB) and opponent color space. There is no argument in the scientific literature on which color space is the best [FR98].

RGB space is a broadly used color space for image display. It is composed of three color components red, green, and blue. These components are called "additive primaries" since a color in RGB space is produced by adding them together. In contrast, CMY space is a color space primarily used for printing. The three color components are cyan, magenta and yellow. These three components are called "subtractive primaries" since a color in CMY space is produced through light absorption.

The CIE L*a*b* and CIE L*u*v* spaces are spaces that consist of a luminance or lightness component (L) and two chromatic components a and b or u and v [FR98]. CIE L*a*b* is designed to deal with subtractive colorant mixtures, While CIE L*u*v* is designed to deal with additive colorant mixtures.

HSV (or HSL, or HSB) space is widely used in computer graphics and is a more intuitive way of describing color. The three color components are hue, saturation and value (or lightness, brightness). The hue is invariant to the changes in illumination and camera direction and hence more suited to object detection or retrieval. RGB coordinates can be easily translated to the HSV (or HLS, or HSB) coordinates using this following steps given by Travis[Tra91]. To convert from RGB to HSV (assuming normalised RGB values) one have to find first the maximum and minimum values from the RGB triplet. Saturation, S, is then:

$$S = \frac{max - min}{max} \tag{2.1}$$

and Value V, is:

$$V = max \tag{2.2}$$

The Hue, H, is then calculated as follows. First calculate R'G'B':

$$\begin{aligned} R' &= \frac{max - R}{max - min} \\ G' &= \frac{max - G}{max - min} \\ B' &= \frac{max - B}{max - min} \end{aligned} \tag{2.3}$$

Hue, H, is then converted to degrees by multiplying by 60 giving HSV with S and V between 0 and 1 and H between 0 and 360. More details about RGB and HSV color spaces will follow in the Chapter 4 as they are both chosen in the implementation of our method.

The Figure 2.4 and 2.5 illustrate the RGB and HSV color points geometric representations .

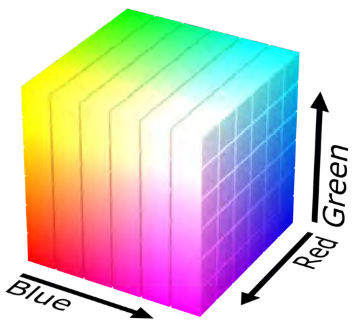


Figure 2.4: RGB cube

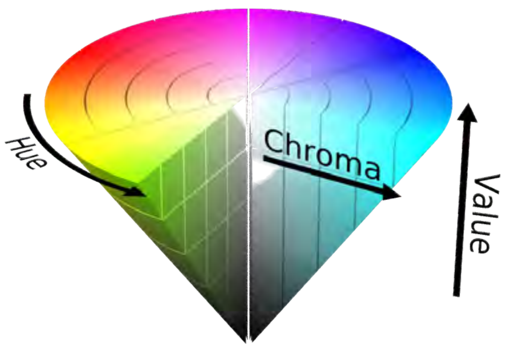


Figure 2.5: HSV cylindre

Another color space, called the opponent color [FSZ03] and that uses color axes (R-G, 2B-R-G, R+G+B) is also interesting and very used in the image processing literature. Its representation has the advantage of isolating the brightness information on the third axis. With this solution, the first two chromaticity axes, which are invariant to the changes in illumination intensity and shadows, can be down-sampled since humans are more sensitive to brightness than they are to chromatic information.

2.3.2 Color Features

Color moments

The mean, variance and standard deviation (Skewness) of an image or a window are known as color moments. Following equations define the mean, variance and standard deviation of an image or a window in an image that have a number of N pixels [FSZ03].

First order moment : Mean

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij} \tag{2.4}$$

Second order moment : Variance

$$\sigma_i = (\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - \mu_i)^2)^{1/2} \tag{2.5}$$

Third order moment : Skewness

$$s_i = (\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - \mu_i)^3)^{1/3} \tag{2.6}$$

Color Histogram

The color histogram serves as an effective representation of the color content of an image if the color pattern is unique compared with the rest of the data set. The color histogram is easy to compute and effective in characterizing both the global and local distributions of colors in an image. In addition, it is robust to translation and rotation about the viewing axis and changes only slowly with the scale, occlusion and viewing angle.

Since any pixel in the image can be described by three components in a certain color space (for instance, red, green, and blue components in RGB space or hue, saturation and value in HSV space), a histogram, i.e., the distribution of the number of pixels for each quantized bin, can be defined for each component. Clearly, the more bins a color histogram contains, the more discrimination power it has. However, a histogram with a large number of bins will not only increase the computational cost, but will also be inappropriate for building efficient indexes for image databases.

Furthermore, a very fine bin quantization does nor necessarily improve the retrieval performance in many applications. One way to reduce the number of bins is to use the opponent color space which enables the brightness of the histogram to be down sampled. Another way is to use clustering methods to determine the K best colors in a given space for a given set of images. Each of these best colors will be taken as a histogram bin. Since the clustering process takes the color distribution of images over the entire database into consideration, the likelihood of histogram bins in which no or very few pixels fall will be minimized. Another option is to use the bins that have the largest pixel numbers since a small number of histogram bins capture the majority of pixels of an image. Such a reduction does not degrade the performance of histogram matching, but may even enhance it since small histogram bins are likely to be noisy.

Color Histogram does not take the spatial information of pixels into consideration, thus very different images can have similar color distributions. To increase discrimination power, several improvements have been proposed to incorporate spatial information. A simple approach is to divide an image into sub-areas and calculate a histogram for each of those sub-areas. As introduced above, the division can be as simple as a rectangular partition, or as complex as a region or even object segmentation. Increasing the number of sub-areas increases increases the information about location, but also increases the memory and computational time.

Color Coherent Vector (CCV)

CCV addresses the problem of considering the spatial information of pixels that color Histogram doesn't take into consideration. In CCV each histogram bin is partitioned into two types: coherent and incoherent. If pixel value belongs to a large uniformly-colored region falls into coherent type. Otherwise it falls into incoherent [FSZ03].

Let α_i denote the number of coherent pixels in the ith color bin and β_i denote the number of incoherent pixels in an image. Then, the CCV of the image is defined as the vector $((\alpha_1, \beta_1), (\alpha_1, \beta_1), \cdots, (\alpha_n, \beta_n))$ with $(\alpha_1, \beta_1), (\alpha_1, \beta_1), \cdots, (\alpha_n, \beta_n)$ is the color histogram of the image. The CCV outperforms generally color histogram in Image Retrieval tasks [PZM96], especially for those images which have either mostly uniform color or mostly texture regions due to its additional spatial information but not very used in computer vision and pattern recognition tasks. In addition, for both the color histogram and color coherence vector representation, the HSV color space provides better results than CIE L*u*v* and CIE L*a*b* space.

2.4 Texture Analysis

Texture is a fundamental character of natural images. It plays an important role in visual perception and provides information for image understanding and scene interpretation.

Texture classification is an active research topic in computer vision and pattern recognition. Early texture classification methods focus on the statistical analysis of texture images

2.4.1 Local Texture Features:

Ojala et al. [OPH96] introduced the LBP operator which is one of the best performing texture operators, and which has been widely used in various applications such as face description and recognition. It is as well proven to be very discriminative and its key advantages, particularly, its invariance to monotonic gray-level changes and computational efficiency, make it suitable for many of current image analysis tasks. For a bibliography of LBP-related research.

Local Binary Pattern

The LBP operator is originally designed for texture description. The operator assigns a label to every pixel of an image by Thresholding the 3×3 - neighborhood of each pixel with the center pixel value and considering the result as a binary number. Then, The 256-bin LBP histogram computed over a region is used for texture description. Fig 2.6 illustrates the basic LBP operator.

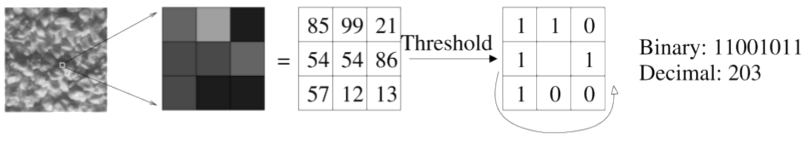


Figure 2.6: An example of the basic LBP operator.

Mathematically, Given a certain pixel, an LBP is computed by comparing it with its neighbors [GZZ10]

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,\ s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.7}$$

where g_c is the gray value of the central pixel, g_P is the value of its neighbors, P is the total number of involved neighbors, and R is the radius of the neighborhood.

To be able to deal with textures at different scales, the LBP operator was later extended to use neighborhoods of different sizes [OPM02]. Defining the local neighborhood as a set of sampling points evenly spaced on a circle centered at the pixel to be labeled allows any radius and number of sampling points.

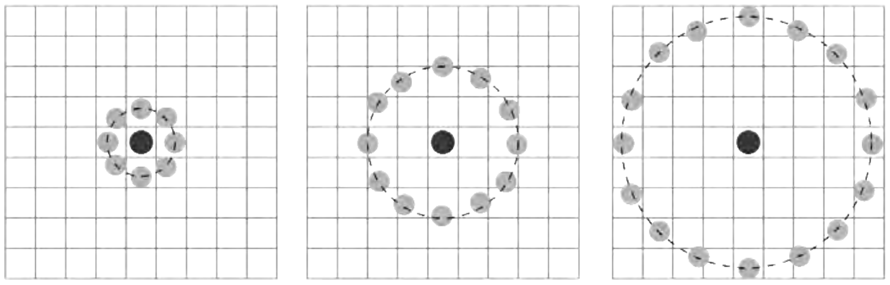


Figure 2.7: Examples of the extended LBP operator.the circular (8, 1), (16, 2) and (24, 3) neighborhoods

2.4.2 Scale-invariant feature transform

The SIFT [Low99] [Low01] is a method for extracting interest point features From images, that is, it not only detects interest point locations but also extracts features around the points that can be used to perform reliable matching between different views of an object or scene. The SIFT features are invariant to not only to image orientation but also image scale, and provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. For image matching, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors.

2.4.3 Gabor-Fisher

Gabor features [LW02] are very popular in the computer vision field, its effectiveness have been proved through many researches exploiting Gabor features and precisely in some particular tasks such as face recognition.

Before we explain how the representation of the Gabor feature is expressed, we should first define what Gabor Wavelets (kernels, filters) are. Gabor wavelets were introduced to image analysis due to their biological relevance and computational properties [Dau85], [JP87]. The Gabor wavelets, whose kernels are similar to the 2-D receptive field profiles of the mammalian cortical simple cells, exhibit desirable characteristics of spatial locality and orientation selectivity, and are optimally localized in the space and frequency domains. The Gabor wavelets (kernels, filters) can be defined as follows [Dau80]:

$$\Psi_{u,v}(z) = \frac{\|k_{u,v}\|^2}{\sigma^2} \exp(\frac{-\|k_{u,v}\|^2 \|z\|^2}{2\sigma^2}) \left(\exp(ik_{u,v}z) - \exp(\sigma^2/2)\right) \tag{2.8}$$

where μ and ν define the orientation and scale of the Gabor kernels, z = (x,y) , $\|.\|$

denotes the norm operator, and the wave vector $k_{\mu,\nu}$ is defined as follows:

$$k_{\mu,\nu} = k_\nu \exp(i\Phi_\mu) \tag{2.9}$$

where $k_\nu = k_{max}/f^\nu$ and $.\Phi_\mu = \pi\nu/8$. k_{max} is the maximum frequency, and is the spacing factor between kernels in the frequency domain [LW02].In most cases one would use Gabor wavelets of five different scales and eight orientations.

The Gabor wavelet representation of an image is the convolution of the image with a family of Gabor kernels as defined by (2.8). Let $I(x, y)$ be the gray level distribution of an image, the convolution of image and a Gabor kernel $\Psi_{u,v}$ is defined as follows:

$$O_{u,v}(z) = I(z) * \Psi_{u,v}(z) \tag{2.10}$$

where z = (x,y), $*$ denotes the convolution operator, $O_{u,v}$ and is the convolution result corresponding to the Gabor kernel at orientation μ and scale ν.

The Gabor filtering coefficient $O_{u,v}(z)$ is a complex number, which can be rewritten as $O_{\mu,\nu}(z) = M_{\mu,\nu}(z)\exp(i\theta_{\mu,\nu}(z))$ with $M_{\nu,\mu}(z)$ being the magnitude and $\theta_{\mu,\nu}(z)$ being the phase. It is known that magnitude information contains the variation of local energy in the image. In [LW02] , the augmented Gabor feature vector $\mathfrak{Feat}_{Gabor}$ is defined via uniform down-sampling, normalization and concatenation of the Gabor filtering coefficients

$$\mathfrak{Feat}_{Gabor} = \{O_{0,0}^{(\rho)t}, O_{0,1}^{(\rho)t}, \cdots, O_{4,7}^{(\rho)t}\}^t \tag{2.11}$$

where $O_{(\nu,\mu)}^{(\rho)}$ is the concatenated column vector from down-sampled magnitude matrix $M_{\mu,\nu}^{(\rho)}$ by a factor of ρ , and t is the transpose operator [YZ10] .

The application of Gabor Features is still limited to some of the object description applications [GA09], and precisely to face description applications , and that's because it generally outperforms other feature descriptors in this particular tasks .

2.4.4 Histogram of Oriented Gradients

HOG features has been introduced by Navneed Dalal and Bill Triggs [DT05], who have developed and tested several variants of HOG descriptors, with differing spatial organization, gradient computation and normalization methods.

Histogram of oriented gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing [CB10]. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI). In order to extract HOG from an image:

- Firstly gradient orientations at every pixel are calculated:

  The most common method is to apply the 1D centered point discrete derivative mask in both the horizontal and vertical directions. Specifically, this method requires filtering the grayscale image with the following filter kernels. being given an image I , we obtain the x and y derivatives using a convolution operation:

  $$I_x = I * D_x = I * \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$$
  $$I_y = I * D_y = I * \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

  The magnitude of the gradient is : $|G| = \sqrt{{I_x}^2 + {I_y}^2}$

  The orientation of the gradient is given by : $\theta = arctan(I_y/I_x)$

- Secondly a histogram of each orientation in a small rectangular region is calculated.

- Finally the HOG feature vector is created by concatenating the histograms of all small regions.

Histogram of Oriented Gradient (HOG) descriptors are proven to be effective at object detection, and in particular with human detection. For that reason, these features have been used to solve a variety of problems, including pedestrians recognition and tracking. On the other hand, Their use in road and lane detection is still limited.

2.4.5 The Discrete Cosine Transform

The discrete cosine transform (DCT) is a media processing technique for converting a signal into elementary frequency components. It has gained its popularity for its usefulness in various applications. For instance, it is widely used in data and image compression for its strong energy compaction property [ANR74] but it can also be used in Filtering, Coding, Pattern recognition and many other applications [RY90].

In this master thesis, We have used the Discrete Cosine Transform for the building of a feature vector used in the classification problem of the road and non-road region which is the main task of this master thesis. Therefore, a more detailed explanation of this Transform and the way the DCT feature is built, are given in the Chapter 4 of this thesis.

3 Method

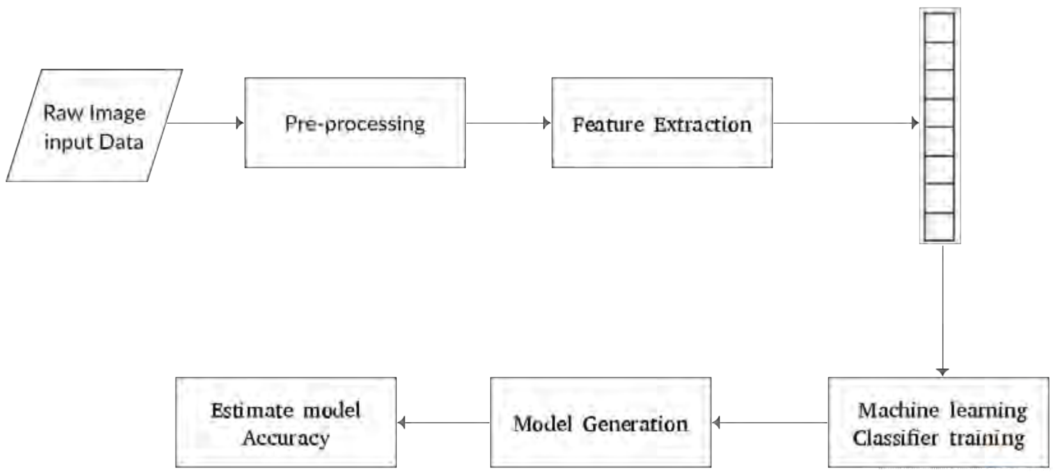


Figure 3.1: General approach followed in this thesis

3.1 Pre-processing

This Step is of an utmost importance as it will help us to reduce processing and computation time. The idea is to locate the Region of Interest(ROI) of each frame and eliminate all irrelevant data points (e.g. Sky region, Buildings...) from later precessing.

But first, We will try to adjust raw image sequence intensity values for a better visual displaying of it.

3.1.1 Raw Image Data Description

Image Data Semantic Meaning

The possessed raw data represent acquired images of road regions using a camera mounted on a car's front and in a fixed position and orientation.

The image stream contains different road conditions and environment situations e.g. Shadows, sun ... Likewise, the roadsides represent differences e.g pavement, sidewalks , grass, metal road barriers .. so that, our data span all possible situations . This criterion is proven to be of a great importance as it will allow the classification learners used later to achieve generalization performance and this way, our learning machine will be able to perform accurately on unseen new examples after having experienced our learning data set.

Figure 4.3 illustrates sample frame captures of the used data set.

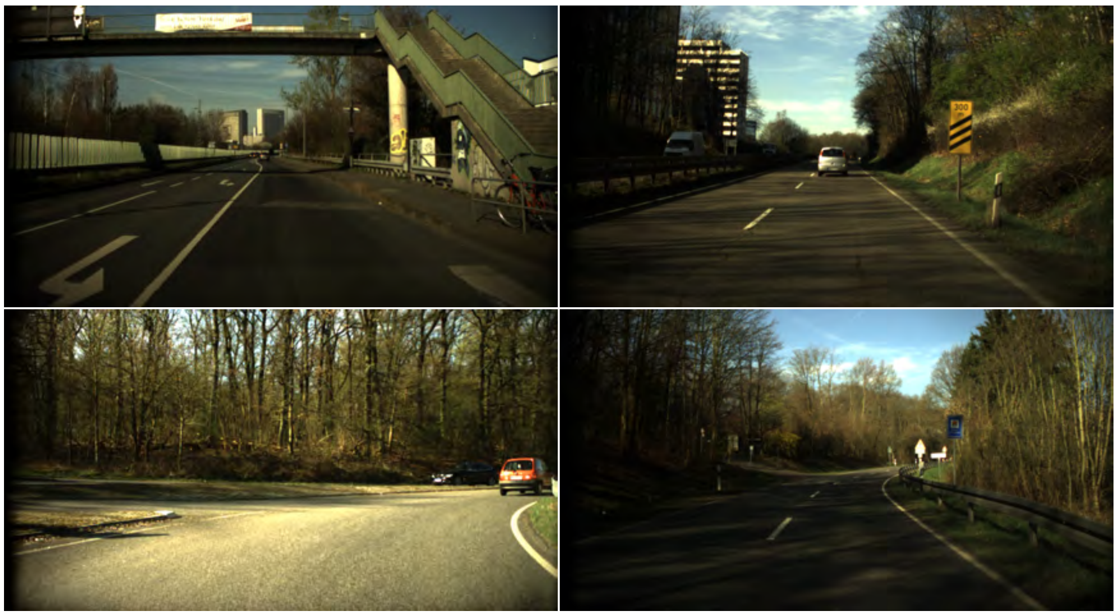


Figure 3.2: Sample Frames Displayed using ImageMagick Software

Color Depth

we use in this thesis sRGB Deep color images,i.e 16-bit color depth per channel, this color depth allows for more than a billion color variations (2^{48} possible color) and therefore, this will allow for a higher level of description of image information.

Image Stream File Format

To simplify the Input/Output operations, The image's sequence are concatenated together in a single binary PPM formatted stream file (Portable Pixel Map) respecting the format specification i,e no data, delimiters, or padding before, after, or between concatenated images.

This format which is characterised by simplicity and Human-readability consists of a header identifying the file type, the encoding and the dimensions of the images contained in the file, followed by the image data.

3.1.2 Image enhancement

We used in this thesis contrast stretching technique which is an image enhancement technique that attempts to improve the contrast of an image by stretching the range of intensity values it contains to make full use of possible values. Unlike histogram equalization, contrast stretching is restricted to a linear mapping of input to output values.

The first step to perform a contrast stretching of an image is to determine the limits over which image intensity values will be extended. These lower and upper limits of the new output will be fixed. Next, the histogram of the original image is examined to determine the value limits in the input image. If the original range covers the full possible set of values, straightforward contrast stretching will achieve nothing, but sometimes most of the image data is contained within a restricted range, this restricted range can be stretched linearly.This can be summarized using the function:

$$P_{out} = (P_{in} - c)(\frac{b-a}{d-c}) + a \tag{3.1}$$

with:
a (resp. b) : the lower (resp. upper) intensity value of the output image.
c (resp. d) : the lower (resp. upper) intensity value of the input image.
P_{in} (resp. P_{out}) : the original (resp. output) pixel value.

The problem using the last only mapping is that a single outlying pixel with either a very high or very low value can severely affect the value of c or d and this could lead to very unrepresentative scaling. Therefore a more robust approach is to first take a histogram of the image, and then select c and d at, say, the n^{th} percentile in the histogram (that is, n% of the pixel in the histogram will have values lower than c, and n% of the pixels will have values higher than d. This prevents outliers affecting the scaling so much.

We carry out a contrast stretching on the input images by adjusting its intensity values for a better visual displaying of it. This is done by mapping the intensity values to new values such that 1% of data is saturated at low and high intensities of the input images (i.e values are rejected and left outside the chosen limits for low and high intensities), this operation will increase the contrast of the output images.

This pre-processing step is optional and it has no effect on the final results of the proposed method, it is done only for displaying reasons.

3.1.3 Region of Interest

The importance of this step comes to the computational time that we will gain by performing it. The ROI determination is done using an empirical estimation of the vanishing point of the road region which is by definition The point at which the road region disappears or ceases to exist.

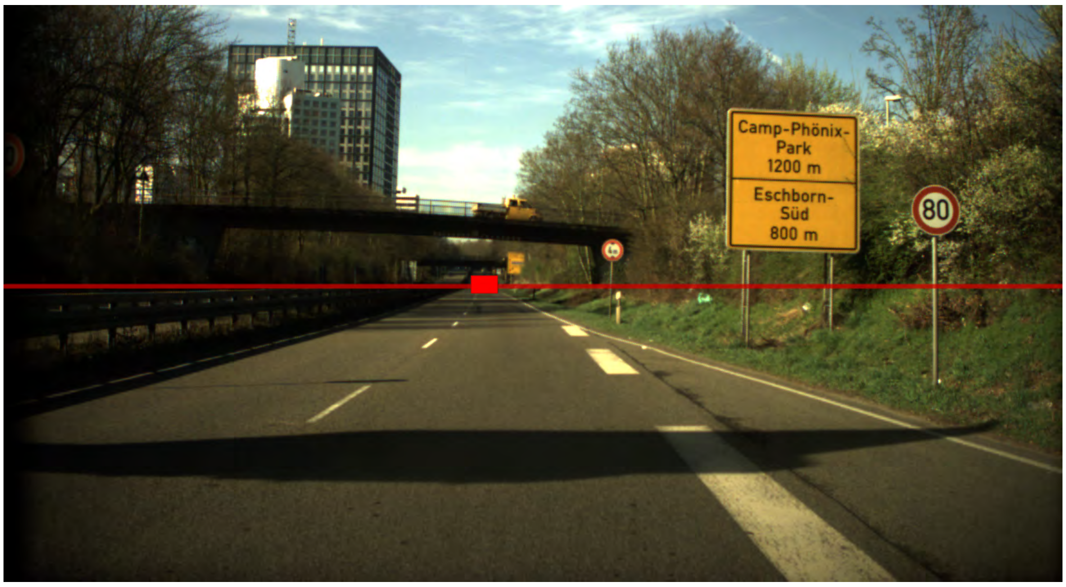


Figure 3.3: The vanishing point represented by the red square

The vanishing point detection was a case study in many previous works. In both Kong & al. paper [KAP09] and Bui & Nobuyama paper [BN12] , a soft-voting method was proposed to detect the vanishing point. But in our work, the vanishing point need not to be computed as we assume that the camera detecting the scenes is in a fixed position and that that the vanishing point does not vary between input sequence frames. An Empirical position is then chosen and the frames are cropped horizontally following the line passing by the vanishing point and parallel to the frames horizontal axes.

3.2 Feature Extraction

3.2.1 Block-based Feature extraction

Block processing techniques are used to maximize signal processing flexibility and independence of estimated parameters between successive blocks of sampled data, In other words, it is a way to collect information within the individual blocks without any compromise. the approach consists of dividing the image into blocks of uniform size and isolating the most relevant features of each block. Many previous studies used the Block-based Feature extraction techniques for image segmentation as well as for object detection tasks. In a study of Manikantan al. [KGVS12], a block-based DCT feature was calculated for the face recognition problem.

Block processing operations are performed by splitting the input frame into blocks of equal size and when the dimensions of this frame are not integral multiples of the block size chosen, the algorithm pads zeroes along the rows and/or columns to get the integral number of blocks. The block size is a parameter that depends on the application e.g a block size of 8x8 pixels is chosen for the JPEG compression. It is also possible to create an overlapping block border between neighbour blocks. The overlap size is also a parameter to be chosen depending on the application specifications. when the overlap size is 0x0 pixels. the block splitting is called a distinct block processing as there is no overlap and the neighbour blocks are totally independent.

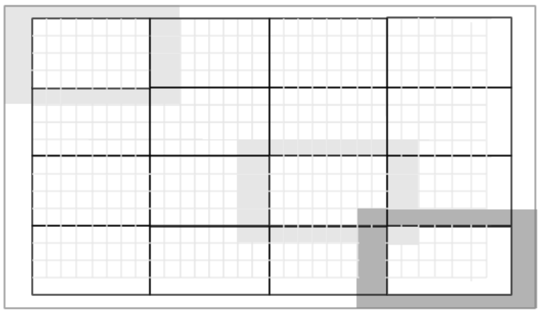


Figure 3.4: Block splitting with overlaps

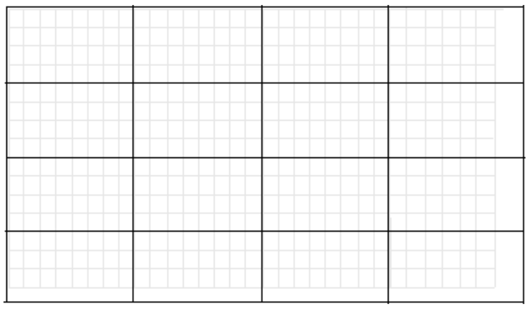


Figure 3.5: Distinct Block Splitting

In this Thesis, we have chosen to implement a block-based Feature extraction method. And for that aim, We have decided to split the input frames into blocks presenting overlaps.

3.2.2 Texture feature extraction

Texture Analysis

Texture based features provide measures of properties such as smoothness, regularity and coarseness of an input image. Three principal approaches are used to describe the texture of an image. Statistical approaches produce characterizations of textures as smooth, grainy, coarse. Structural techniques deal with the arrangement of image elements e.g the description of texture based on regularly spaced parallel lines. Spectral techniques are used to detect global periodicity in an image by identifying high energy, narrow peaks in the Fourier spectrum of the image.

Description of textures using a spectral approach is of a particular interest in this thesis. As we will use a Discrete cosine Transform based method to extract the image texture features.

Discrete Cosine Transform

The discrete cosine transform (DCT) is a media processing technique for converting a signal into elementary frequency components. It has gained its popularity for its usefulness in various applications. For instance, it is widely used in data and image compression for its strong energy compaction property [ANR74] but it can also be used in Filtering, Coding, Pattern recognition and many other applications [RY90].

The Discrete cosine transform operates on a signal to transform it from spatial domain to frequency domain. This will help to separate the input signal into spectral sub-bands of different importance. There exist four variants of the Discrete cosine Transform namely $DCT-I$, $DCT-II$, $DCT-III$, $DCT-IV$. The most commonly used type is the $DCT-II$, is often simply referred to as " the DCT " and was defined for the first time in the paper of N.Ahmed & al [ANR74] as follows: The $DCT-II$ of a one dimensional data sequence $X(m)$, $m = 0, 1, \cdots, (M-1)$ is defined as :

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=1}^{M-1} X(m)$$

$$G_x(u) = \frac{2}{M} \sum_{m=1}^{M-1} X(m) \cos\frac{(2m+1)u\pi}{2M}, u = 1, 2, \cdots, (M-1) \tag{3.2}$$

where $Gx(u)$ is the uth DCT coefficient. Similarly, the inverse transformation is defined as:

$$X(m) = \frac{1}{\sqrt{2}} G_x(0) + \sum_{k=1}^{M-1} G_x(u) \cos\frac{(2m+1)u\pi}{2M}, m = 1, 2, \cdots, (M-1) \tag{3.3}$$

When the data sequence is two dimensional as in the case of images, A two-dimensional $DCT-II$ is applied to to the data sequence to transform it to frequency domain. The 2-D $DCT-II$ for a data sequence $X(m,n)$, $m = 0, 1, \cdots, (M-1)$, $n = 0, 1, \cdots, (N-1)$ is just an extension of the 1-D $DCT-II$ defined in the equation (4.1) and is given by:

$$G_x(0,0) = \frac{2}{MN} \sum_{m=1}^{M-1} \sum_{n=1}^{N-1} X(m,n)$$

$$G_x(0,v) = \frac{4}{MN} \sum_{m=1}^{M-1} \sum_{n=1}^{N-1} X(m,n) cos \frac{(2n+1)v\pi}{2N},$$

$$G_x(u,0) = \frac{4}{MN} \sum_{m=1}^{M-1} \sum_{n=1}^{N-1} X(m,n) \cos\frac{(2m+1)u\pi}{2M},$$

$$G_x(u,0) = \frac{4}{MN}\sum_{m=1}^{M-1}\sum_{n=1}^{N-1} X(m,n)\cos\frac{(2m+1)u\pi}{2M}\cos\frac{(2n+1)v\pi}{2N}, \tag{3.4}$$

$$where\, u = 1,2,\cdots,(M-1),\;\; v = 1,2,\cdots,(N-1)$$

The inverse cosine transform will be defined in this case as follows:

$$X(m,n) = \frac{1}{2}G_x(0,0) + \sum_{u=1}^{M-1}\sum_{v=1}^{N-1} G_x(u,v)\cos\frac{(2m+1)u\pi}{2M}\cos\frac{(2m+1)v\pi}{2N}, \tag{3.5}$$

$$where\;\, m = 1,2,\cdots,(M-1),\, n = 1,2,\cdots,(N-1)$$

The first DCT coefficient $G_x(0,0)$ is a special case because it's proportional to the average horizontal pixel intensity change of all the input samples. It is for that reason sometimes called the "DC component" of the signal (DC as in direct current).

DCT Feature Extraction

In this section, a DCT feature extraction approach is proposed. This approach is composed of two stages. In the first stage, we choose a block size and an overlap parameter and then we apply the DCT to each block of the image. In the second stage, we construct the feature vector of each block as follows :

if we have already chosen the block size to be $[N,N]$ and the overlap size to be $[P,P]$. Then the $2D$ DCT type II will be applied to a block of a size $[N+2\ {*}P, N+2\ {*}P]$ resulting in an image that have the same size in which the pixel of the position (i,j) will represent the DCT coefficient $G_x(i,j)$. As long as the first coefficient $G_x(0,0)$ will be significantly affected by any illumination change as it represents the average intensity change of this block. we proceed by not counting the first coefficient in the construction of the feature vector to achieve robustness to illumination changes. Instead, we choose to divide the rest of the coefficients by this component. Thus, The feature vector will have a size of $2{*}(N+2{*}P)-1$ and will take this form :

$$DCT_{Feature}(Block) = \frac{1}{G_x(0,0)} \begin{pmatrix} G_x(0,1) \\ G_x(0,2) \\ G_x(0,3) \\ \vdots \\ G_x(0, N+2{*}P-1) \\ G_x(1,0) \\ G_x(1,1) \\ \vdots \\ \vdots \\ G_x(N+2{*}P-1, N+2{*}P-1) \end{pmatrix}$$

3.2.3 Color Feature Extraction

In this section, we will describe the approach that we have used to extract color characteristics of the image.

Adopted Color Spaces

the approach we took to get the color features of the input frames is to use the color histogram associated to each block of these frames which are divided in the same manner as for the last feature i.e same block-size and same overlap-size. For this reason, the choice of the color space become of ultimate importance as the color histogram constructed will depend directly of the color space chosen.

To examine as much possible situations as we can, we have chosen to use separately two color spaces , the standard RGB color space and then the HSV color space. Thereafter, we compare the results to conclude which color space will fit to our problem.

Color Feature Extraction

The step that comes after choosing the color space representation is the extraction of the color features. Therefore, a 3D color histogram is built for each block i.e each channel is represented by a 1D color histogram. And then a concatenation of the three color histogram distributions is done to get a color Feature vector.

Because he number of elements per histogram i.e the number of bins will determine the size of our feature Vector. choosing this parameter will be of a great importance in the construction of our feature vector

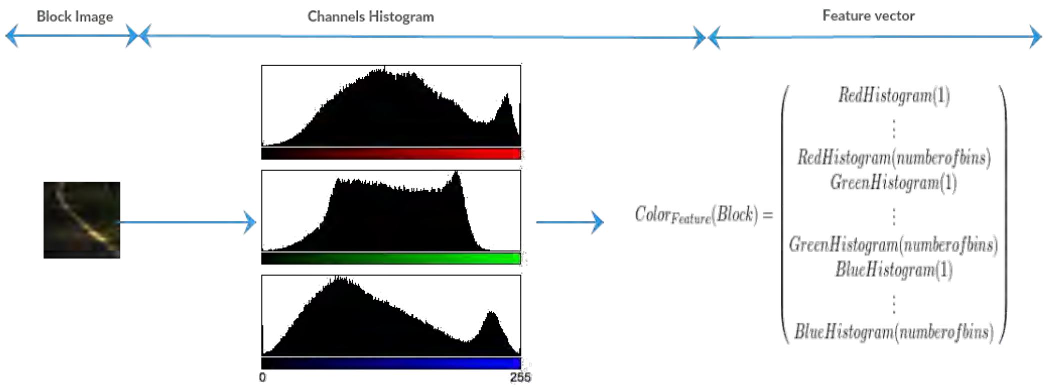

Block Image
Channels Histogram
Feature vector
0
255
$Color_{Feature}(Block) = \begin{pmatrix} RedHistogram(1) \\ \vdots \\ RedHistogram(numberofbins) \\ GreenHistogram(1) \\ \vdots \\ GreenHistogram(numberofbins) \\ BlueHistogram(1) \\ \vdots \\ BlueHistogram(numberofbins) \end{pmatrix}$

Figure 3.6: Color features building

3.2.4 Position Feature Extraction

The Last feature extracted from our images is the position feature which spots the block's position. The importance of recording such information comes from the fact that the camera installed in the front of the car has a fixed position and angles with respect to the ground and that the distribution of the road regions will have this way the tendency to occupy the low regions and the center of the image.

The feature vector that we have constructed is associated to each block and describes the latter by the position of its center pixel. If the block doesn't contain centers e.g a block of size 10x10, the pixel describing this block will be the pixel with the highest pixel coordinates in the smallest central square of size 2x2.

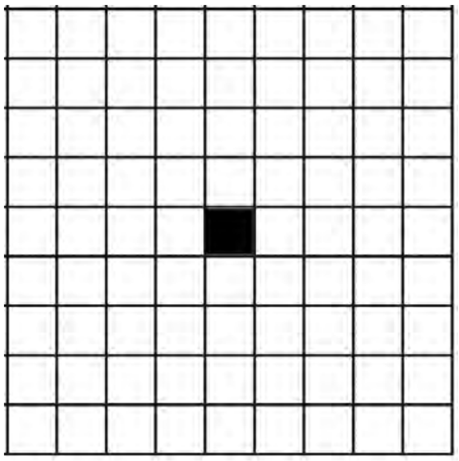


Figure 3.7: Center pixel of a 9x9 block

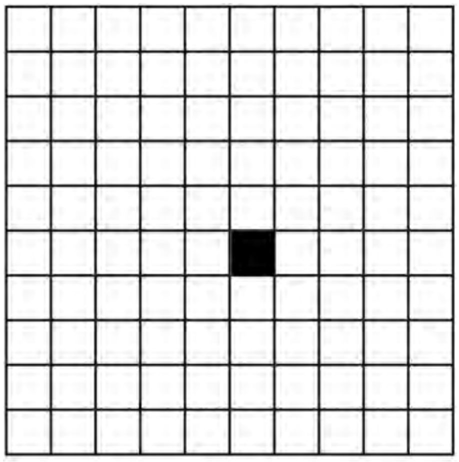


Figure 3.8: Center pixel of a 10x10 block

The position feature vector of a particular block is a two dimensional vector. The first coefficient and the second coefficient of this vector mark respectively the x-axis and the y-axis of the pixel chosen to describe this block.

$$Position_{Feature}(Block) = \begin{pmatrix} x - Coordinate\,of\,the\,central\,pixel \\ y - coordinate\,of\,the\,central\,pixel \end{pmatrix}$$

3.2.5 Feature vector

After extracting the DCT, Color and position features of each block. The next step is to combine these features in a single vector that will be a general descriptor of each associated block.

The global feature vector will take this form :

$$FeatureVector(Block) = \begin{pmatrix} Position_{Feature}(Block) \\ Color_{Feature}(Block) \\ DCT_{Feature}(Block) \end{pmatrix}$$

3.3 Classification Model

In this section, a general description of the approach used in building our learning model is presented.

3.3.1 Supervised Learning

In this Thesis, we have chosen a supervised approach in learning our data. In Supervised learning, the training sample consists of pairs, each contains an instance x,i.e a constructed feature, and a label y : $(x_i, y_i)_{i=1}^{n}$. Such (instance,label) are called $labeled\,data$

Supervised learning approaches try to learn from labeled training examples, i.e with ground-truth labels given in advance.

3.3.2 Support Vector Machine

Support Vector Machines were first introduced with a paper at the COLT 1992 conference as a machine learning method by Boser, Guyon and Vapnik (1992), Three years later, the soft margin classifier was introduced by Cortes and Vapnik (1995). SVM is a method for the classification of both linear and nonlinear data. The SVM algorithm uses generally (in the case the data is not linearly separable) a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (i.e a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by an hyperplane using margins and support vectors. A more detailed description of SVMs follows, based on Burges (1998) and Vapnik (1998).

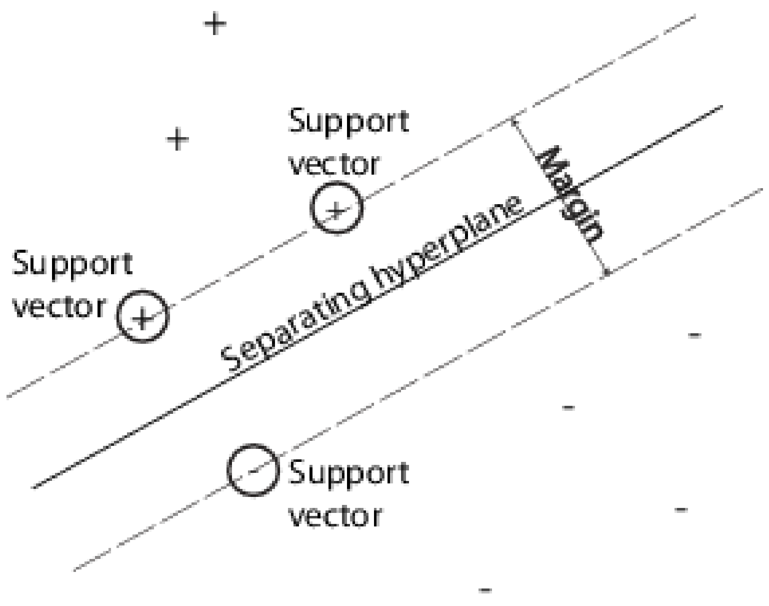


Figure 3.9: Optimal hyperplane selection via max-margin

Mathematical formulation

For a given dataset of n samples of the form $(x_1, y_1), \cdots, (x_n, y_n)$ where y_i are either 1 and 0, a constant denoting the class to which that point x_i belongs. Each x_i is p-dimensional real vector.

SV classifiers are based on the idea of the dividing (or separating) hyperplane $f(x)$ with normal $w \in R^p$, and an offset $b \in R$

$$f(x) = \langle w, x \rangle + b = 0 \ , \ x \in R^p \tag{3.6}$$

When the data is linearly separable, the best separating hyperplane (i.e decision boundary) is found by solving the optimization problem [JWHT14] [CST00]

$$w = \underset{w \in R^p}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 \ \ \textit{subject to} \ y_i \langle w_i, x_i \rangle \geq 1 \quad i = 1, \cdots, n \tag{3.7}$$

In the non-separable case, an introduction of a parameter ξ_i that measures the amount of violation in the constraints $y_i \langle w_i, x_i \rangle \geq 1$. The optimization problem takes in this case the form:

$$(w, \xi) = \underset{w \in R^p, \xi \in R^n}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i$$

Subject to $y_i \langle w_i, x_i \rangle \geq 1 - \xi_i \quad \xi_i \geq 1 \quad i = 1, \cdots, n$ (3.8)

The Dual problem for this optimization problem can be formulated this way:

$$\alpha = \max_{\alpha_i \geq 0} \sum_i \alpha_i + \sum_{jk} \alpha_j \alpha_k y_j y_k \langle x_j, x_k \rangle$$

Subj. $\alpha_i \geq 0 \quad \forall, \sum_i \alpha_i y_i = 0 \quad$ (3.9)

Learning in the Feature Space

When the data is not separable in the Input space, we map it into a Feature space where this data can be linearly separable. This solves the problem of expressing complex functions that separate between data.Besides, because of the computational complexity working with very large vectors when mapping into a higher dimensional feature space. we introduce the concept of Kernels. A kernel function can be interpreted as a kind of similarity measure between the input objects that defines implicitly a mapping ϕ to a higher dimensional space. This will allow us to reduce the computational time of defining and computing the mapping function. A mapping function ϕ kernel function K is defined as follow : For an input Space $\mathcal{X}$ and a feature space $\mathcal{F}$

$$\phi : \mathcal{X} \longrightarrow \mathcal{F}$$
$$K : \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{F}$$

In this Thesis we have built six models based on the support vector machine classifier. One model uses a linear Kernel and the five other models were built using these following kernel functions.

Type of Kernel	**Inner product kernel** $K(x, x_i), i = 1, 2, \cdots, N$
Quadratic Kernel	$K(x, x_i) = (x.x_i + \Theta)^2$ Θ is the width of the kernel
Cubic Kernel	$K(x, x_i) = (x.x_i + \Theta)^3$
Fine Gaussian Kernel	$K(x, x_i) = exp(-\frac{x.x_i}{2\sigma^2})$ with $\sigma = \sqrt{P} \times 4$
Medium Gaussian Kernel	$K(x, x_i) = exp(-\frac{x.x_i}{2\sigma^2})$ with $\sigma = \sqrt{P} \times 4$
Coarse Gaussian Kernel	$K(x, x_i) = exp(-\frac{x.x_i}{2\sigma^2})$ with $\sigma = \sqrt{P} \times 4$

3.3.3 Artificial Neural network

An artificial neural network is a classifier modeled based on operations of biological neural networks and hence can be defined as an emulation of biological neural systems.

A human brain contains an enormous amount of nerve cells called neurons. Each of these neurons are connected to many other similar neurons, creating a very complex network. Each cell collects inputs from all the neural cells it is connected to as a form of an electric signal, if this signal reaches a certain threshold, the neuron signals to all the cells it is connected to. The analogue of the neurons in an artificial neural network is called "Perceptron" and it operates the same way as its real analogs.

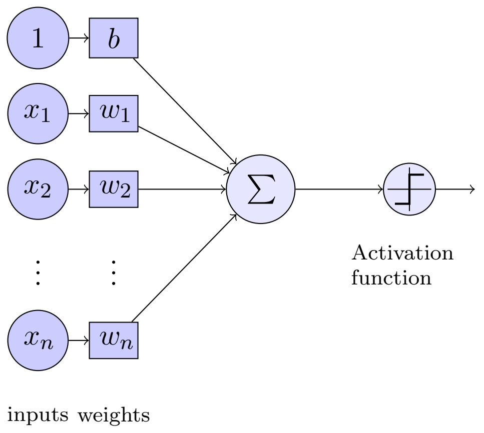


Figure 3.10: A graphical representation of a simple perceptron. Here n is the number of connections to the perceptron, w_i is the weight associated with the *ith* connection and x_i is the value of the *ith* connection. b represents the threshold.Picture by the author.

The perceptron can take several weighted inputs and summarize them,if the combined input exceeds a threshold it fires (i.e sends an output which is determined by the activation function and is often chosen to be between 0 and 1 or â1 and 1).An important thing to note here is that the derivative of the activation function should be easily calculated and it would be mathematically convenient if it is expressed in terms of the original activation function as this derivative is needed in the well-known training algorithm of neural networks **"Backpropagation algorithm"**.
The functioning of a Perceptron can be summed up in the following equation:

$$y = \Phi(\sum_{i=1}^{n} w_i x_i + b) \tag{3.10}$$

where y is the output signal, Φ is the activation function, n is the number of connections to the perceptron, w_i is the weight associated with the *ith* connection and x_i is the value of the ith connection, b represents the threshold. A graphical representation can be found in figure 3.10. Although the simplicity of the idea behind the conception of a perceptron, the strength of this latter is proven when several perceptrons are combined and work together. The perceptrons are often organized in layers, where each layer takes inputs from the previous one, applies weights and then signals to the next layer.

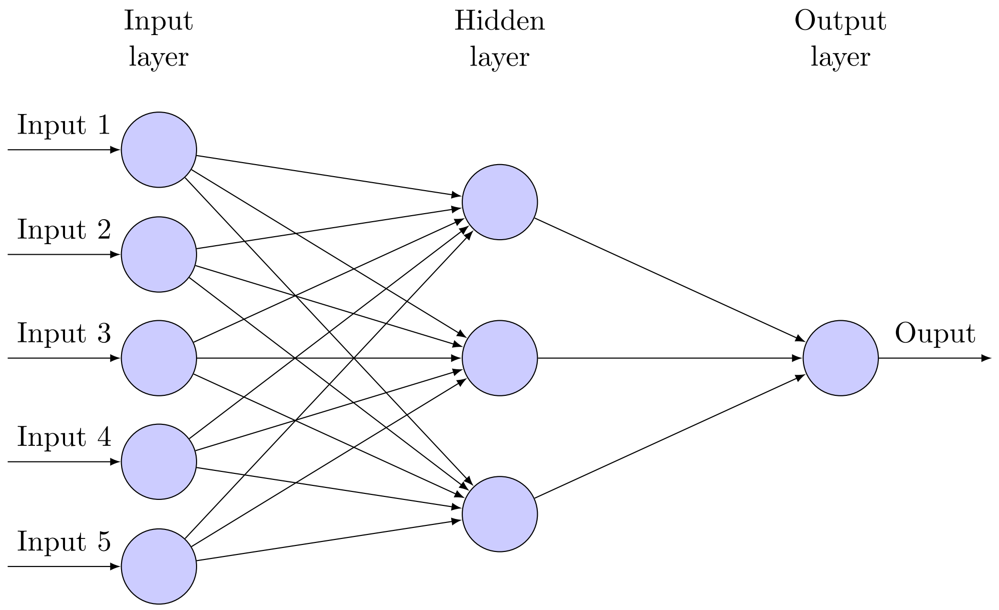


Figure 3.11: A graphical representation of an artificial neural network with one hidden layer. Picture by the author

A classifier must be able to learn from examples by adapting the weights on the incoming connections of these hidden units .

In an ANN, this is achieved by updating the weights of the connections between the layers.In fact, There are several ways of doing this where most of them involve initializing these weights and then feed an example to the network. The error made by the network at the output level is then calculated and feed backwards through “backpropagation”. This process is then used to update the weights. Hence, the network can learn to distinguish between several different classes by repeating the **Backpropagetion algorithm**.

Machine learning problems, particularly the ANN approach, face always the overfitting when the classifier becomes too good at recognizing the training examples, at the expense of not being able to recognize a general input. This can be avoided by **cross-validation**, where the network is trained on one set of data, and then evaluated on a separate one. When the error starts rising in the validation set, the network might be over-fitted. If previous networks' configurations are saved, the network can then be rolled back to the one which gave the smallest error.

4 Implementation

4.1 Environment

The following software, respectively operating systems, were used in This Thesis:

- Ubuntu 16.04 LTS 64 bit
- Matlab R2016a 64 bit
- EvaluationGui Labeling Tool

4.2 Feature vectors Labeling

Once the prepossessing of our input raw images sequence is done, a labeling process can be executed for supervised learning approaches.
The Figure 4.1 shows a sample preprocessed frame

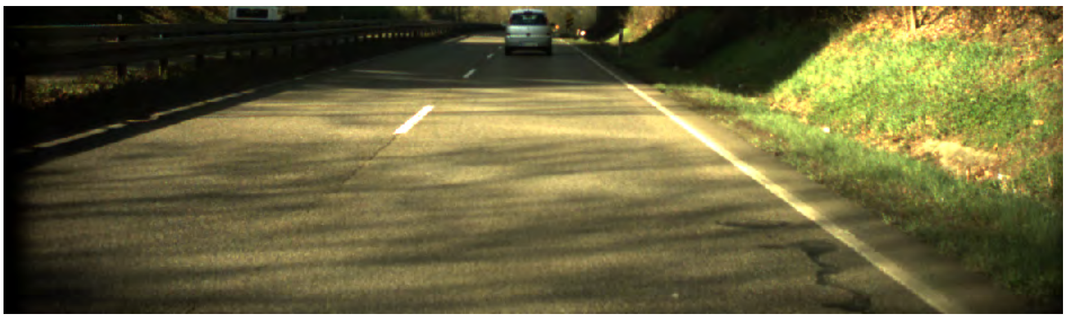
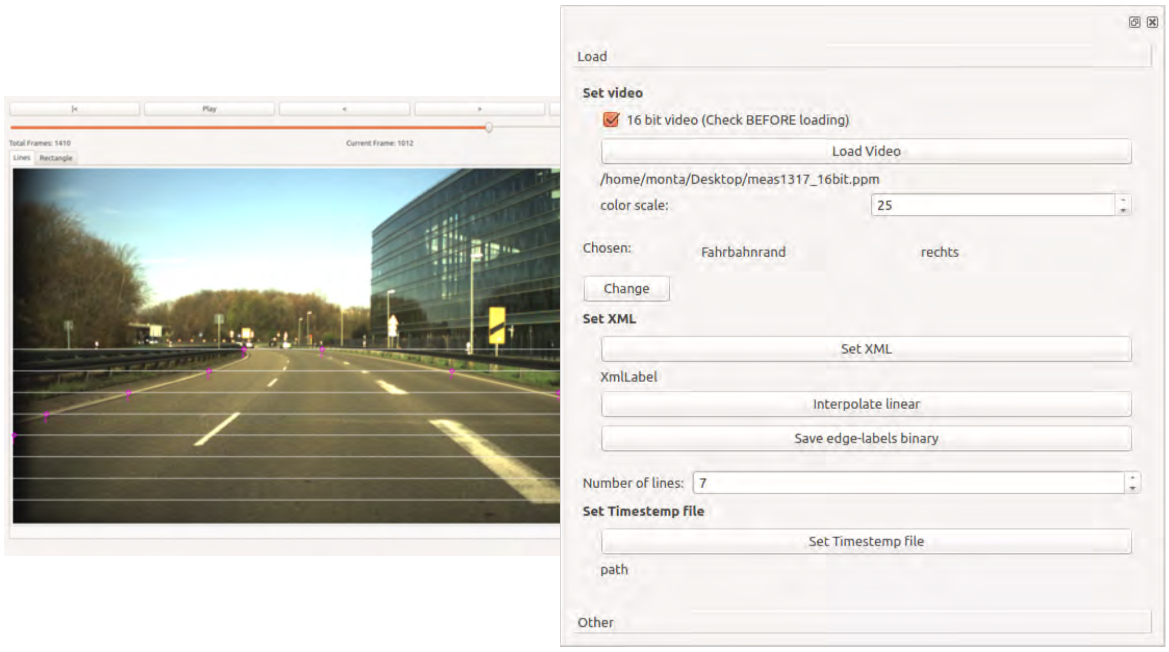


Figure 4.1: Sample preprocessed frame

For Labeling our Frames, we have used a graphical user interface tool named *Evaluation Gui* and developed at the university of Passau. Using this tool, we can load the road image sequence, select in an input frame the pixels that correspond to the road boarders and finally get as an output a binary file which represents a stream of road edge's pixel positions. Some other options are also available , such as an interpolation option of the selected pixels between different frames.

Each single frame treated by the *Evaluation Gui* is represented by its number in the image sequence followed by the selected pixel positions of the left side of the road and then the selected pixel positions of the right side.



Total Frames: 1410
Current Frame: 1012

[✓] 16 bit video (Check BEFORE loading)
/home/monta/Desktop/meas1317_16bit.ppm
color scale: 25
Chosen: Fahrbahnrand rechts
XmlLabel
Number of lines: 7
path

Figure 4.2: EvaluationGui interface

For associating a label to each block indicating whether this block belongs to a road region or not in the input images sequence . We start by extracting all the road edge's pixel positions from the binary file and then specify a polygonal region of interest by interpolating the pixel positions of each frame in this binary file, the region of interest will be the road area region. the next step consists of constructing a pixel-wise mask image in which the pixels that hold the same positions of the region of interest specified will have a label 1 and the rest will have the label 0. This way we constructed a pixel-wise mask image where each pixel of the treated image have an associated label in the mask image.

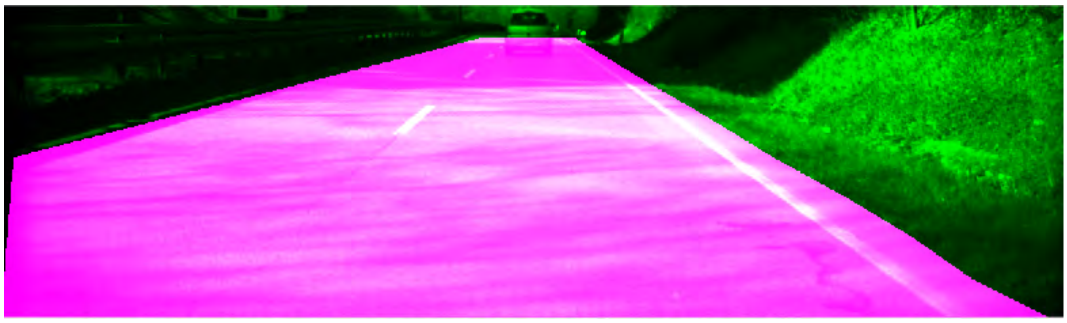


Figure 4.3: Pixel-wise constructed Mask

As we are going to carry out a block-wise classification, our aim in this step is to construct a block-wise mask in which each block is described by one label. For this reason, we have built a simple function that has as an input the pixel-wise mask and the block size chosen for extracting the features. a computation of the number of pixels labeled as 1 and those labeled as 0 is done in each block. And each block will have the label which is more present within its pixels.

The Figure 4.4 shows a Block-wise labeled sample frame

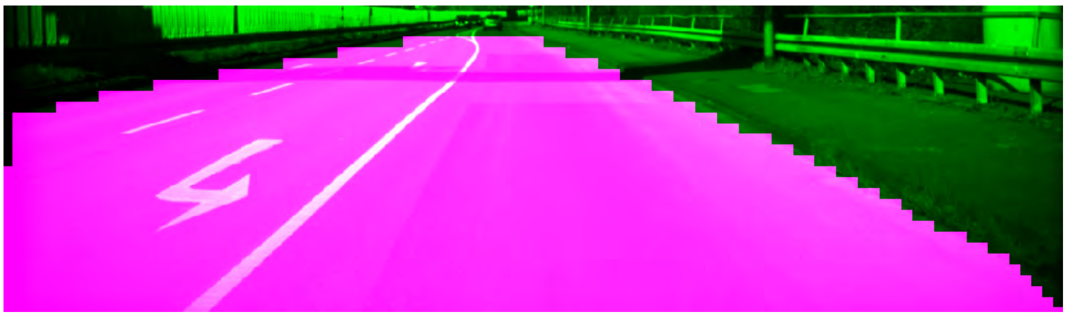


Figure 4.4: Block-wise Mask Image

A table is then constructed, it contains "a bag of words" feature vectors extracted from the processed frames, for each is associated an additional coefficient which represents the label of that vector. The table is used later by our Learner app as a labeled training data for the supervised machine learning model construction.

The following table illustrates an example of how a training data is built in the form of a table.

	Position Feature	Color Feature	DCT Feature	Label
Feature Vector$_1$	Position Feat$_1$ $\in R^2$	Color Feat$_1$	DCT Feat$_1$	Label$_1$ $\in [\|0,1\|]$
...		...		
Feature Vector$_N$	Position Feat$_N$ $\in R^2$	Color Feat$_N$	DCT Feat$_N$	Label$_N$ $\in [\|0,1\|]$

Table 4.1: Our constructed Training Data

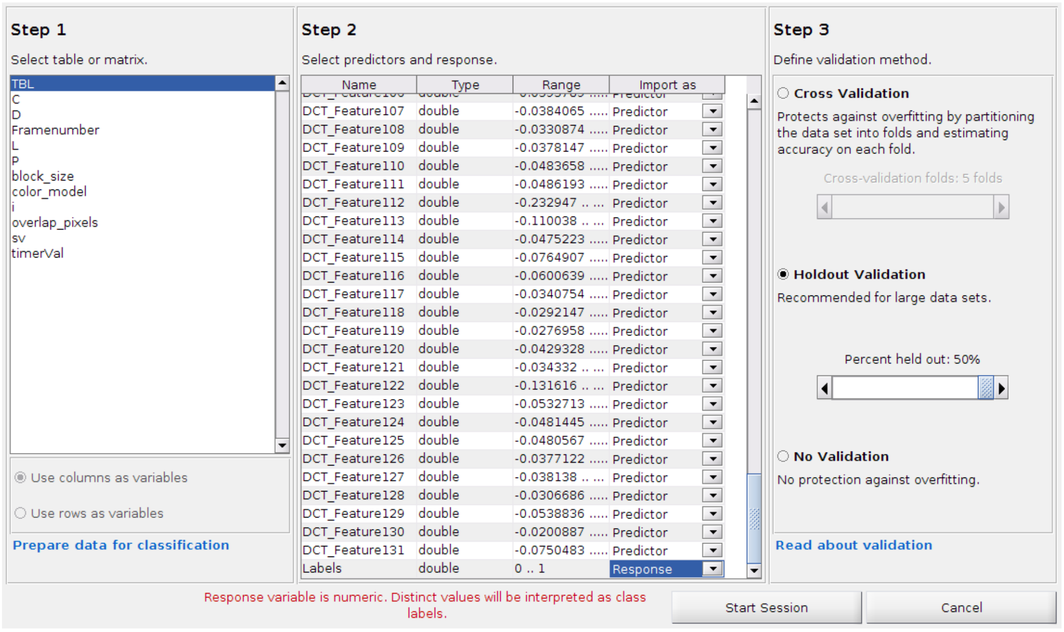
4.3 Matlab Classsification Learner App

We use for training the model the Matlab Classification Learner App. This App, introduced to Matlab in R2015a under the Statistics and Machine Learning Toolbox, allows us to explore supervised Machine learning using various classifiers. this means that it will be useful in searching for the best classification model type by comparing between classifiers performance.For this aim, we can choose to train several classification types including decision trees, support vector machines (SVM), and k-nearest neighbors, logistic regression and ensemble classification.

We can also use the Learner App in some other machine learning tasks such as selecting features, specifying validation schemes, and assessing results. Using the Classification learner App is simple and can be described in this steps

4.3.1 Data Importing and validation method

This first step consists of importing the table containing the labeled feature vectors and then selecting the validation scheme. In this App the feature vectors are called predictors, their labels are called the responses. we will use from now and over these appellations for explicitness and clarity reasons.



Step 1

Select table or matrix.

TBL
C
D
Framenumber
L
P
block_size
color_model
i
overlap_pixels
sv
timerVal

(● Use columns as variables) ○ Use rows as variables

Prepare data for classification

Step 2

Select predictors and response.

Name	Type	Range	Import as
DCT_Feature107	double	-0.0384065	Predictor
DCT_Feature108	double	-0.0330874	Predictor
DCT_Feature109	double	-0.0378147	Predictor
DCT_Feature110	double	-0.0483658	Predictor
DCT_Feature111	double	-0.0486193	Predictor
DCT_Feature112	double	-0.232947	Predictor
DCT_Feature113	double	-0.110038	Predictor
DCT_Feature114	double	-0.0475223	Predictor
DCT_Feature115	double	-0.0764907	Predictor
DCT_Feature116	double	-0.0600639	Predictor
DCT_Feature117	double	-0.0340754	Predictor
DCT_Feature118	double	-0.0292147	Predictor
DCT_Feature119	double	-0.0276958	Predictor
DCT_Feature120	double	-0.0429328	Predictor
DCT_Feature121	double	-0.034332	Predictor
DCT_Feature122	double	-0.131616	Predictor
DCT_Feature123	double	-0.0532713	Predictor
DCT_Feature124	double	-0.0481445	Predictor
DCT_Feature125	double	-0.0480567	Predictor
DCT_Feature126	double	-0.0377122	Predictor
DCT_Feature127	double	-0.038138	Predictor
DCT_Feature128	double	-0.0306686	Predictor
DCT_Feature129	double	-0.0538836	Predictor
DCT_Feature130	double	-0.0200887	Predictor
DCT_Feature131	double	-0.0750483	Predictor
Labels	double	0 .. 1	Response

Step 3

Define validation method.

○ **Cross Validation**

Protects against overfitting by partitioning the data set into folds and estimating accuracy on each fold.

Cross-validation folds: 5 folds

[✓ **Holdout Validation**]

Recommended for large data sets.

Percent held out: 50%

○ **No Validation**

No protection against overfitting.

Read about validation

Response variable is numeric. Distinct values will be interpreted as class labels.

Figure 4.5: Data importing interface

4.3.2 Classification algorithms

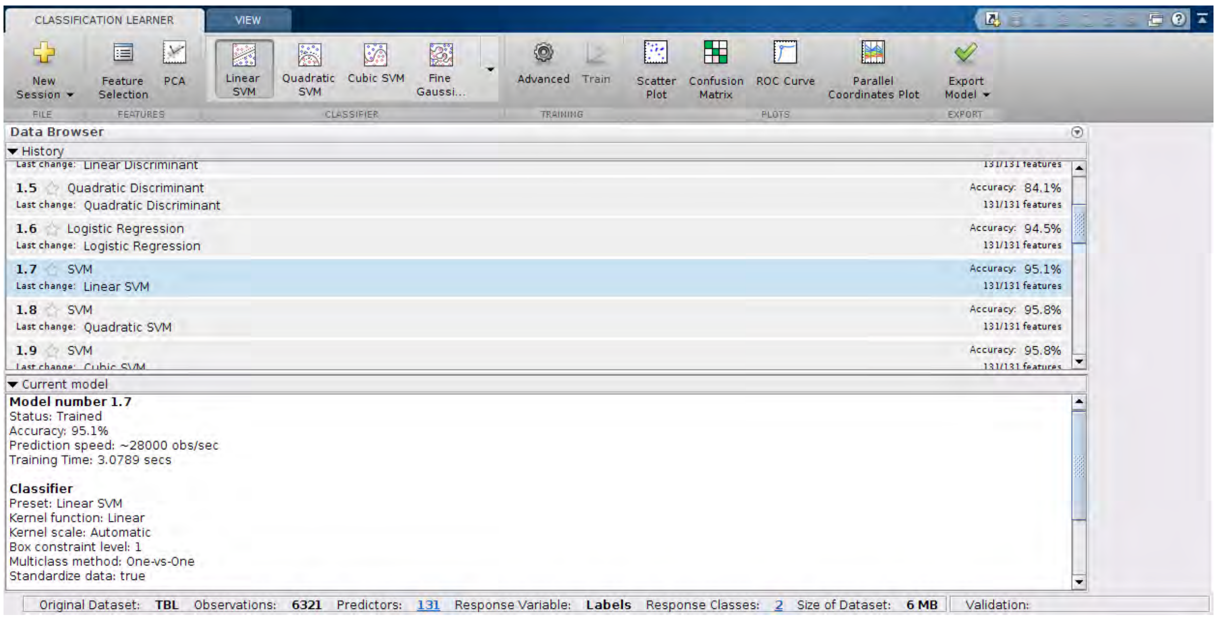
The Matlab classification learner App provides various classification algorithms. A complete list of all the included classifiers is illustrated in the following table.

Decision trees	Support vector machines	Nearest neighbor classifiers	Ensemble classifiers
Deep tree	Linear SVM	Fine KNN	Boosted trees
Medium tree	fine Gaussian SVM	medium KNN	bagged trees
Shallow tree	medium Gaussian	coarse KNN	subspace KNN
	coarse Gaussian SVM	cubic KNN	subspace discriminant
	quadratic SVM	cosine KNN	
	cubic SVM	weighted KNN	

Table 4.2: Included classification algorithms in the classification Learner App

4.3.3 Model Comparaison and Assessment

The Learner app allows us to assess classifier performance using confusion matrices, ROC curves, or scatter plots, compare model accuracy using the misclassification rate on the validation set.



Last change: Linear Discriminant — 131/131 features

1.5 Quadratic Discriminant — Accuracy: 84.1%
Last change: Quadratic Discriminant — 131/131 features

1.6 Logistic Regression — Accuracy: 94.5%
Last change: Logistic Regression — 131/131 features

1.7 SVM — Accuracy: 95.1%
Last change: Linear SVM — 131/131 features

1.8 SVM — Accuracy: 95.8%
Last change: Quadratic SVM — 131/131 features

1.9 SVM — Accuracy: 95.8%
Last change: Cubic SVM — 131/131 features

Model number 1.7
Status: Trained
Accuracy: 95.1%
Prediction speed: ~28000 obs/sec
Training Time: 3.0789 secs

Classifier
Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Original Dataset: **TBL** Observations: **6321** Predictors: 131 Response Variable: **Labels** Response Classes: 2 Size of Dataset: **6 MB** Validation:

Figure 4.6: Data Training

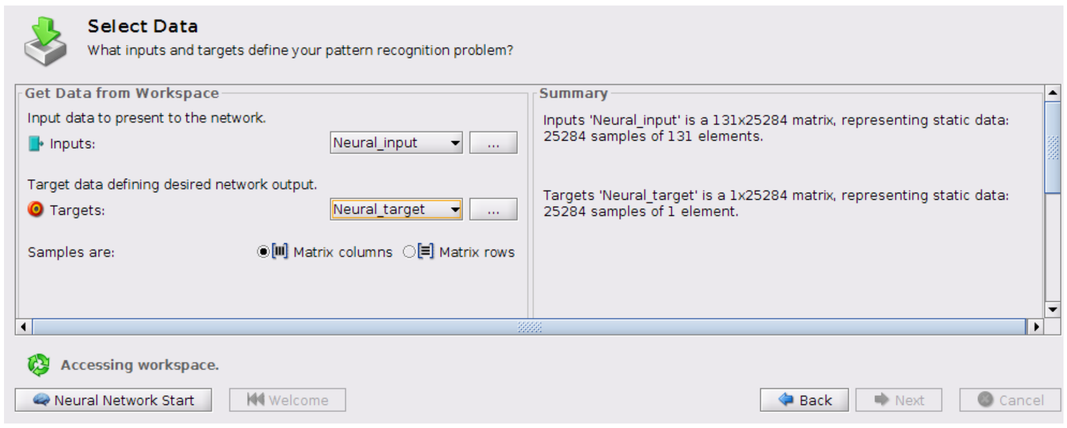
4.4 Matlab Neural Net Pattern Recognition App

In this thesis, we have used the Neural Net Pattern recognition App to build predictive models based on the neural networks. The App comes under the Neural Network Toolbox, allows us to create, train, visualize, and simulate neural networks. The following steps describe the way to train a neural network using the Neural Net Pattern

Recognition Tool.

4.4.1 Input And Target Data Importing

In this App the feature vectors are called input data, their labels are called the target data.The data are imported in a Matrix formatting.



Select Data

What inputs and targets define your pattern recognition problem?

Get Data from Workspace

Input data to present to the network.

Inputs: Neural_input

Target data defining desired network output.

Targets: Neural_target

Samples are: [✓ Matrix columns] Matrix rows

Summary

Inputs 'Neural_input' is a 131x25284 matrix, representing static data: 25284 samples of 131 elements.

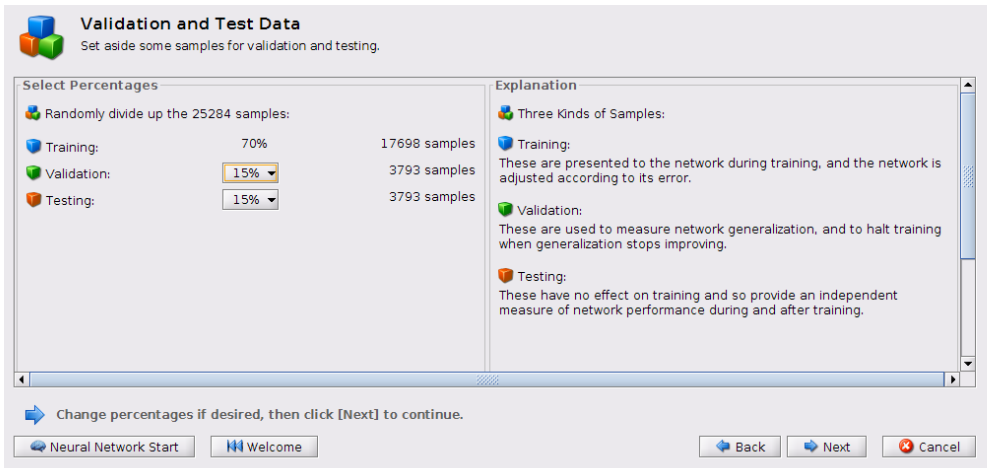
Targets 'Neural_target' is a 1x25284 matrix, representing static data: 25284 samples of 1 element.

Accessing workspace.

Figure 4.7: Data importing Interface

4.4.2 Validation method and parameters

The validation method and the parameters associated to this method can be chosen using this Interface.



Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 25284 samples:

Training:	70%	17698 samples
Validation:	15%	3793 samples
Testing:	15%	3793 samples

Explanation

Three Kinds of Samples:

Training:
These are presented to the network during training, and the network is adjusted according to its error.

Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

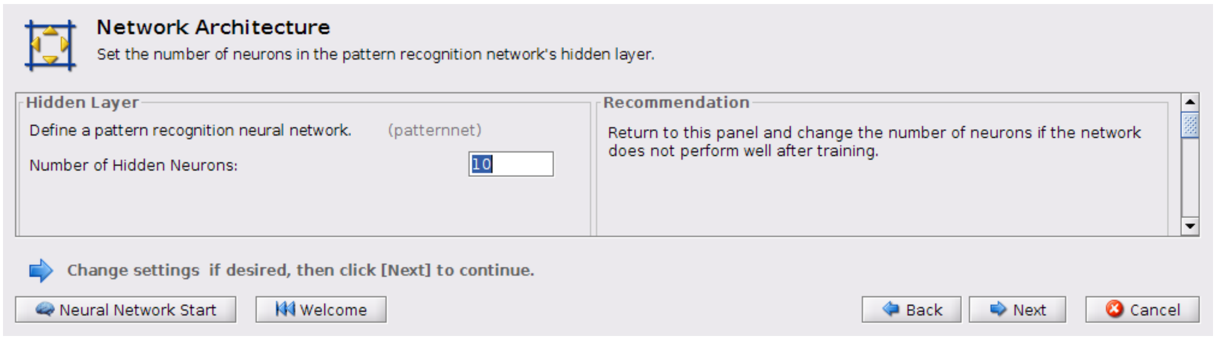
Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.

Change percentages if desired, then click [Next] to continue.

Figure 4.8: Validation parameters' Interface

4.4.3 Network Architecture

Using the Interface shown in Fig.(4.9), We can choose the network architecture, i.e the number of the hidden layers.



Network Architecture

Set the number of neurons in the pattern recognition network's hidden layer.

Hidden Layer

Define a pattern recognition neural network. (patternnet)

Number of Hidden Neurons: 10

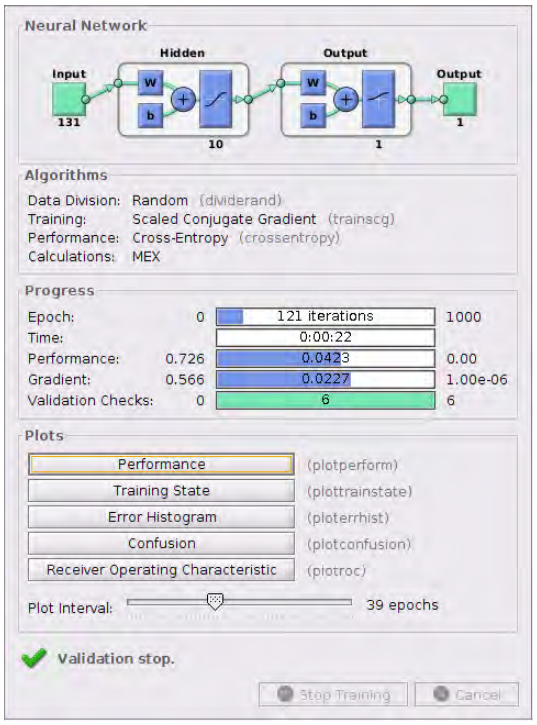
Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Change settings if desired, then click [Next] to continue.

Figure 4.9: Network Architecture Interface

4.4.4 Training The Network



Neural Network
Hidden
Output
Input
Output
W
b
W
b
131
10
1
1

Algorithms

Data Division:	Random (dividerand)
Training:	Scaled Conjugate Gradient (trainscg)
Performance:	Cross-Entropy (crossentropy)
Calculations:	MEX

Progress

Epoch:	0	121 iterations	1000
Time:		0:00:22	
Performance:	0.726	0.0423	0.00
Gradient:	0.566	0.0227	1.00e-06
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Confusion (plotconfusion)
Receiver Operating Characteristic (plotroc)

Plot Interval: 39 epochs

Validation stop.

Figure 4.10: The neural network training Interface

This last interface presents the sate of the network when the training is in process.

When the training of the model ends. we can get the performance of the model as well as different plots presenting the error histogram and the confusion matrix.

5 Evaluation

5.1 Chosen parameters

These parameters were used to build our learning models, all the performance results of these models were visualised under these conditions unless it is mentioned. We call them for now the default parameters.

- The block-size is chosen to have 8x8 pixel size :
- The overlap size between blocks is chosen to have 1x1 pixel size.
- The color space is chosen to be the RGB color space.
- The number of bins of the color histograms used to ectract the color features is chosen to be 10.

Using these parameters, we will extract from each preprocessed frame 6321 feature vectors, each one of it has 131 coefficient and a label associated to it. These configurations will lead to a large Training data set if the number of frames becomes bigger. The following table shows the size of the training data depending on the number of frames being in play.

Number of frames	Size of the training data
1 Frame	6 Megabytes
2 Frames	13 Megabytes
10 Frames	64 Megabytes
100 Frame	637 Megabytes
4388 Frames	30 Gigabytes

5.2 Validation Method

5.2.1 Hold-out Validation

Before building the Learning models, we have chosen **Hold-out validation** as a validation scheme which consists of selecting a percentage of the data to use as a test set. The model is trained on the training set and assesses the performance with the test set. In our approach, we have chosen the Training set and the test set to have equal sizes ,i.e 50% of our data is allocated to the training set and the rest to the test set.

This method is based on only a portion of the data. For that reason, it is recommended for large data sets which matchs up to our situation.

Total samples
Training set
Test set

Figure 5.1: Training and Test Set

5.2.2 Accuracy of a model

After building a model, a method for evaluating a model is to calculate its accuracy which represents the testing set examples correctly classified by the classifier. The accuracy is defined as follows

$$Accuracy = \frac{\sum true\,positive + \sum True\,negative}{\sum Total\,population} \tag{5.1}$$

True positive : the **correctly identified** set of vectors .
True negative : the **correctly rejected** set of vectors.
Total population : the whole test set.

5.3 Performance Measurements

5.3.1 Support Vector Machine Classifier

For the default parameters mentioned in 5.1, the model exploiting the SVM classifier reaches very good accuracy results. The results gathering the classification accuracy of the different SVM classification models, i.e using different Kernels, is shown in the following table.

	Number of the Total sample features			
	6321 ft.vect	**12642** ft.vect	**25284** ft.vect	**50586** ft.vect
Linear SVM	**94.2 %**	**95.3 %**	**95.0 %**	**95.0 %**
Quadratic SVM	**95.1 %**	**95.7 %**	**95.7 %**	**96.1 %**
Cubic SVM	**94.7 %**	**95.9 %**	**95.9 %**	**96.1 %**
Fine Gaussian SVM	**83.3 %**	**83.1 %**	**84.3 %**	**85.4 %**
Medium Gaussian SVM	**94.5 %**	**95.5 %**	**95.9 %**	**96.0 %**
Coarse Gaussian SVM	**90.4 %**	**92.6 %**	**94.6 %**	**94.4 %**

Table 5.1: The Predictive built models accuracy

Rather than the accuracy results, the distribution of the Support vectors ,i.e the position of these vectors in the frames from which the training data is constructed, is of an important interest for a precise evaluation of our SV models. As these support vectors are the definers of the separating hyperplane between road and non-road regions, studying their positions can give us more information about the behaviour of our predictive models. For that reason, we have represented some frames in a fusion with their associated support vectors represented in their positions under different conditions.

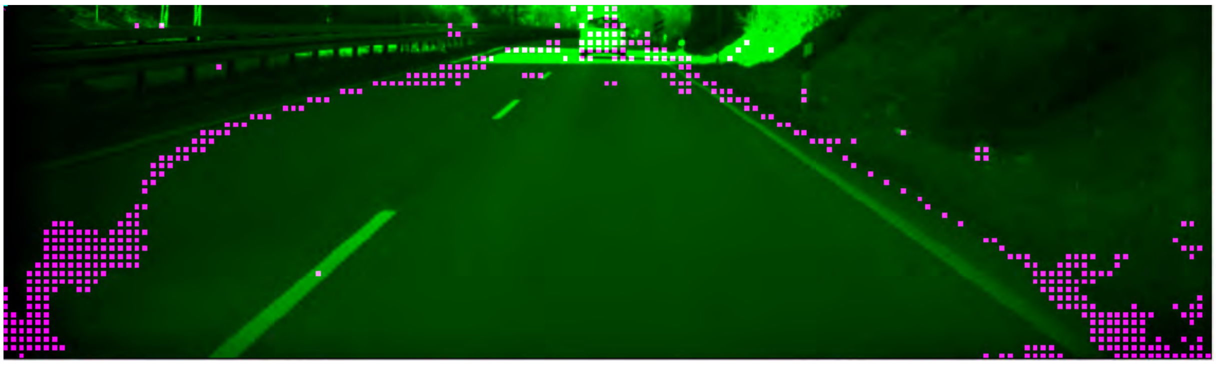


Figure 5.2: Support vectors distribution over an associated frame

As it is shown in the fig 5.3, the distribution of the support vectors follow the road edges as they are the natural delimiters of the road and non-road regions. Robustness against contrast changes is unfortunately poorly achieved. As we can see from the next figure, when the frame presents shadowed and non-shadowed road regions, the model finds a difficulty in detecting the road edges and the support vectors distribution become distributed in a pseudo-irregular manner.

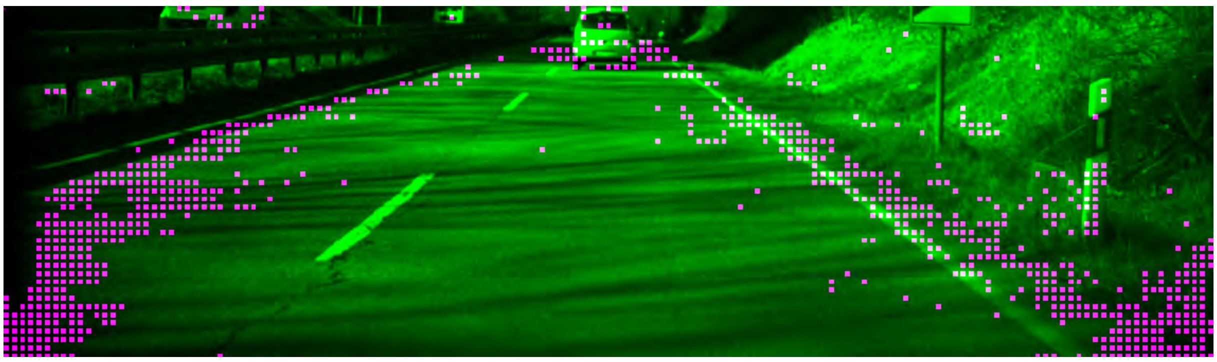


Figure 5.3: Support vectors distribution over an associated frame, Case of a shadowed frame

5.3.2 Other Classification Techniques performance

We have also created classification models using other machine learning algorithms and compared their performance with the Support vector machine performance. The types of the created models and their according accuracy results are summarised in the next table.

	Number of the Total sample features			
	6321 ft.vect	**12642** ft.vect	**25284** ft.vect	**50586** ft.vect
Linear discriminant	**80.4 %**	**79.0 %**	**79.7 %**	**79.0 %**
Quadratic discriminant	**81.6 %**	**82.1 %**	**81.6 %**	**81.2 %**
Simple Tree	**85.7 %**	**85.3 %**	**84.9 %**	**85.6 %**
Medium Tree	**94.4%**	**94.8 %**	**95.4 %**	**95.0 %**
Coomplex Tree	**95.3 %**	**97.1 %**	**97.6 %**	**98.3 %**
Fine KNN	**79.7** %	**80.5 %**	**82.7 %**	**84.3 %**
Medium KNN	**75.8** %	**75.1 %**	**78.1 %**	**78.5 %**
Coarse KNN	**70.2** %	**71.2 %**	**73.8 %**	**74.6 %**
Logistic Regression	**94.4 %**	**95.3 %**	**94.9 %**	**95.0 %**

Table 5.2: Other Predictive models accuracy

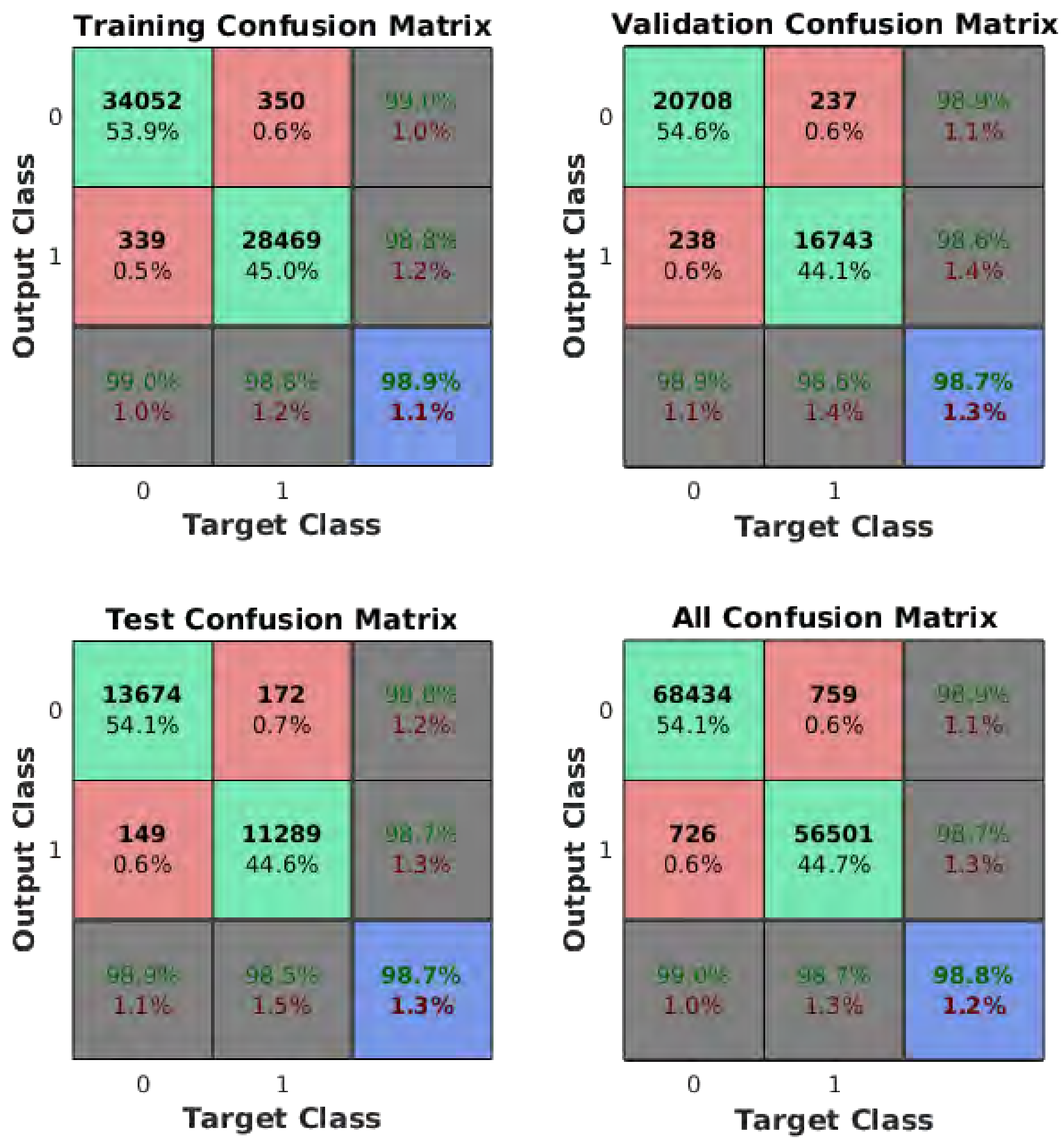
It can be observed from these results that the Support vector machine gives better results than most of the standard used classifiers. We can see clearly that **The nearest neighbours classifier(KNN)** with all its variants appears to do not fit this kind of tasks. But, we can also notice that the **Logistic Regression** classifier, the **The Decision Tree** with a medium number of leaves N , i.e N can reach 20, give good results that can be compared to the SVM performance and a **Complex Tree**, i.e a decision tree with a big flexibility with respect to the number of its leaves, can make astounding results when the number of samples gets bigger.

5.3.3 Neural Network Performance

For the default parameters mentioned above. We have constructed a **126420** data set of labeled samples. we have chosen for the validation set(respect. test set) to be **30%** of all the samples (respect. **20%**) and the rest was allocated to the training set. The training of a two-layer feed forward neural network with sigmoid hidden and softmax output neurons has given the following results summarized in the following confusion matrices Fig(5.4).

5.4 Conclusion

The implementation of this system using Matlab environment and its image processing and machine learning toolboxes, which followed, a step of conception consisting of constructing a suitable feature vector capable of describing as well as it can the information



Training Confusion Matrix

Output Class \ Target Class	0	1	
0	34052 53.9%	350 0.6%	99.0% 1.0%
1	339 0.5%	28469 45.0%	98.8% 1.2%
	99.0% 1.0%	98.8% 1.2%	98.9% 1.1%

Validation Confusion Matrix

Output Class \ Target Class	0	1	
0	20708 54.6%	237 0.6%	98.9% 1.1%
1	238 0.6%	16743 44.1%	98.6% 1.4%
	98.9% 1.1%	98.6% 1.4%	98.7% 1.3%

Test Confusion Matrix

Output Class \ Target Class	0	1	
0	13674 54.1%	172 0.7%	98.8% 1.2%
1	149 0.6%	11289 44.6%	98.7% 1.3%
	98.9% 1.1%	98.5% 1.5%	98.7% 1.3%

All Confusion Matrix

Output Class \ Target Class	0	1	
0	68434 54.1%	759 0.6%	98.9% 1.1%
1	726 0.6%	56501 44.7%	98.7% 1.3%
	99.0% 1.0%	98.7% 1.3%	98.8% 1.2%

Figure 5.4: Confusion Matrix of the Trained neural netwoek model

contained in the input images, has led us to elaborate the results of the last section and to affirm the capability of this system to detect the road regions. We can also note that that the real-time detection of the road regions can be easily reached due to optimized code and the rapidness of classification process of the built models which will allow us to conclude that the main goals of this thesis are reached

This system presents also some constraints, and one can say the shadows that happen to appear in the image sequences can make it harder to the classifier to find the boundary between the road and non-road regions . Nevertheless, the detection of the main part of the road is still present

If one want to build a larger system which is almost insensitive to such external interferences, It could be possible by the integration of feature vectors which are invariable to contrast, but that will definitely make us do some compromises in terms of computational time as the feature vectors will be bigger in size and the models will take longer to classify them. At this step, It could also be interesting to improve this by using some dimensionality reduction techniques such as Principal component analysis (PCA) to faster and lighten the processing and eventually to keep only the necessary components of our feature vector. If we would build a system like that, it would probably be better to present and more importantly, more robust to contrast variations as this code. This improvement that can be done will be discussed further in more details in the next chapter.

6 Future Applications and Developments

To produce better systems, capable of a better performance and a good robustness against the noise and the interferences that can encounter any sequence of images acquired by a camera mounted on a moving car and fed to model a system using machine learning classification techniques. there are certain changes that could and need to be done. We will begin to list some constraints of our system or that we may encounter, and afterwards describe possible solutions to overcome such problems.

- imperfect pixel-wise labelling for the supervised learning task.
- Intensity and contrast fluctuations and environment perturbations.
- long classification time usage.

6.1 Unsupervised learning

Unsupervised learning refers to the task of extracting patterns and structure from raw data without extra information, as opposed to supervised learning where labels are needed, e.g Clustering techniques such as K-means are unsupervised approaches of learning. This will enable us to overcome the imperfection of the labelling process as it is a purely manual task which will bring it necessarily to present errors.

6.2 New Descriptors

In the literature, we can find that descriptors used in this kind of tasks vary considerably from a method to another, this can give us an idea of how much interesting to combine other features that have proven to be effective with the ones we constructed. This can lead finally to overcome the intensity and contrast fluctuations and environment perturbations and certainly to increase the precision i.e. the number of correct answers.

6.3 More Powerful hardware equipments

The evolution of computational ability is proven to be exponential,i.e as it is shown by Moore's law, and every day witnesses the birth of more powerful processing components in terms of computational ability and which can be found easily in the market. We can benefit from this fact by using a more suitable and powerful hardware equipments which can enable us to accelerate as much as possible the processing time of our system and reach at the end a perfect real-time processing of our sequence of input images.

List of Acronyms

ADAS	Advanced Driver Assistance System
SVM	Support vector machine
CCV	Color Coherent Vector
LBP	Local Binary Pattern
SIFT	Scale-invariant feature transform
HOG	Histogram of oriented gradients
ROI	Region of Interest
DCT	Discrete cosine transform
PPM	Portable Pixel Map
JPEG	Joint Photographic Experts Group
ANN	Artificial Neural Network
KNN	k-nearest neighbors

Bibliography

[ANR74] Ahmed, N., T. Natarajan and K.R. Rao: *Discrete Cosine Transform.* Computers, IEEE Transactions on, C-23(1):90–93, 1974.

[BN12] Bui, TH and Eitaku Nobuyama: *A local soft voting method for texture-based vanishing point detection from unstructured road images.* SICE Annual Conference (SICE), 2012 ..., pages 396–401, 2012.

[CB10] Corvee, E. and F. Bremond: *Body Parts Detection for People Tracking Using Trees of Histogram of Oriented Gradient Descriptors.* In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 469–475, Aug 2010.

[CST00] Cristianini, Nello and John Shawe-Taylor: *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods.* Cambridge University Press, New York, NY, USA, 2000.

[Dau80] Daugman, John G.: *Two-dimensional spectral analysis of cortical receptive field profiles.* Vision Research, 20(10):847 – 856, 1980.

[Dau85] Daugman, John G.: *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters.* J. Opt. Soc. Am. A, 2(7):1160–1169, Jul 1985.

[DBH06] Demuth, H.B., M. Beale and M. Hagan: *Neural Network Toolbox for Use with MATLAB: User's Guide.* MathWorks, Incorporated, 2006.

[DT05] Dalal, N. and B. Triggs: *Histograms of oriented gradients for human detection.* In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[ETS14] ETSC: *Road safety planning Good practice examples from national road safety strategies in the EU.* (October):1–15, 2014.

[FR98] Ford, Adrian and Alan Roberts: *Colour space conversions.* 1998.

[FSZ03] Feng, D.D., W.C. Siu and H.J. Zhang: *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications.* Engineering online library. Springer, 2003.

[GA09] Gao, Feng and Haizhou Ai: *Face Age Classification on Consumer Images with Gabor Feature and Fuzzy LDA Method*, pages 132–141. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[GZZ10] Guo, Z., L. Zhang and D. Zhang: *A Completed Modeling of Local Binary Pattern Operator for Texture Classification.* IEEE Transactions on Image Processing, 19(6):1657–1663, June 2010.

[JP87] Jones, J. P. and L. A. Palmer: *An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex.* Journal of Neurophysiology, 58(6):1233–1258, 1987.

[JTA00] Jacobs, G, A Aeron Thomas and A Astrop: *Estimating Global Road Fatalities.* Transport Research Laboratory, (December):36 p., 2000.

[JWHT14] James, G., D. Witten, T. Hastie and R. Tibshirani: *An Introduction to Statistical Learning: with Applications in R.* Springer Texts in Statistics. Springer New York, 2014.

[Kal16] Kala, R.: *On-road Intelligent Vehicles: Motion Planning for Intelligent Transportation Systems.* Elsevier Science, 2016.

[KAP09] Kong, Hui, Jean Yves Audibert and Jean Ponce: *Vanishing point detection for road detection.* 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009, (3):96–103, 2009.

[KGVS12] K, MANIKANTAN, Vaishnavi Govindarajan, Sasi Kiran V V S and Ramachandran S: *Face Recognition using Block-Based DCT Feature Extraction.* Journal of Advanced Computer Science & Technology, 1(4):266–283, 2012.

[Kum16] Kumar, Surinder: *Lane Detection based on Contrast Analysis.* In *Master Thesis* [Kum16].

[Lev16] Levin, Sam: *https://www.theguardian.com/technology/2016/jul/20/elon-musk-tesla-master-plan-part-2-autopilot-feature*, jul 2016.

[Low99] Lowe, D. G.: *Object recognition from local scale-invariant features.* In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[Low01] Lowe, D. G.: *Local feature view clustering for 3D object recognition.* In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–682–I–688 vol.1, 2001.

[LW02] Liu, Chengjun and H. Wechsler: *Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition.* IEEE Transactions on Image Processing, 11(4):467–476, Apr 2002.

[MHC16] Model, Tesla, U S Highway and Freightliner Cascadia: *Preliminary report highway hwy16fh018.* pages 1–3, 2016.

[OPH96] Ojala, Timo, Matti Pietikäinen and David Harwood: *A comparative study of texture measures with classification based on featured distributions.* Pattern Recognition, 29(1):51–59, 1996.

[OPM02] Ojala, Timo, Matti Pietikainen and Topi Maenpaa: *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.* IEEE Transactions on pattern analysis and machine intelligence, 24(7):971–987, 2002.

[PT16] Paluszek, M. and S. Thomas: *MATLAB Machine Learning.* Apress, 2016.

[PZM96] Pass, Greg, Ramin Zabih and Justin Miller: *Comparing Images Using Color Coherence Vectors.* In *Proceedings of the Fourth ACM International Conference on Multimedia*, MULTIMEDIA '96, pages 65–73, New York, NY, USA, 1996. ACM.

[RY90] Rao, K. R. and P. Yip: *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[TPI13] Toroyan, Tami, Margie M Peden and Kacem Iaych: *Global status on road report 2015.* World Health Organization, 19(2):150, 2013.

[Tra91] Travis, D.: *Effective Color Displays: Theory and Practice.* Computers and people. Academic Press, 1991.

[YZ10] Yang, Meng and Lei Zhang: *Gabor Feature Based Sparse Representation for Face Recognition with Gabor Occlusion Dictionary*, pages 448–461. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.