Recognition of vehicle make from a frontal view.

Michal Čonos

October 14, 2007

Abstract

This thesis deals with a vehicle type recognition problem from a frontal view. The proposed solution is based on using a SIFT descriptors [25] for feature extract and using k-NN as a classifier. We also made and evaluated simplifications of this descriptor. These simplifications resulted in 3 different SIFT descriptor variants which are evaluated in this thesis using k-NN and Fisher's linear discriminant on two databases used for learning and testing. The first one contains car images and the second one contains truck images. Database with car images contains around 250 images. The database with truck images contains around 500 images.

Tato práce se zabývá rozpoznáváním typu auta z předního pohledu. Navrhované řešení je založeno na použití SIFT-deskriptoru [25] pro extrakci příznaků a klasifikátoru k-NN. Provedli jsme a vyhodnotili také modifikace tohoto popisu. Tyto vyústily na 3 různé varianty SIFT-deskriptoru, které jsou vyhodnoceny v této práci za použití klasifikátorů nejbližšího souseda a Fisherova lineárního diskriminantu na dvou databázích. Jedna obsahuje obrazy osobních aut, druhá obsahuje auta nákladní. Databáze s osobními auty obsahuje 250 obrazů. Databáze s obrazy nákladních aut obsahuje přibližně 500 obrazů.

Notation

Here we define some notation used in this text: matrices, vectors, sets and scalars are distinguished by font, matrices/vectors are typeset using **bold math font**, sets and scalars are typeset by using *normal math font*. Matrices and sets are typeset using capitalized letters. We do not really use vectors since we use matrix calculus. Vectors are viewed as a nx1 matrix, therefore if $\mathbf{w}$ is a vector then $\mathbf{w}^T\mathbf{w}$ is a scalar and $\mathbf{w}\mathbf{w}^T$ is a dxd matrix, where d is the vector dimension.

s	scalar s
$\|s\|$	absolute value of scalar s
$\mathbf{A}$	matrix $\mathbf{A}$
$\|\mathbf{A}\|$	determinant of matrix $\mathbf{A}$
$\mathbf{w}$	vector $\mathbf{w}$
D	set D
$\|D\|$	cardinality of set D
$\mathbf{w}^T$	transpose matrix/vector
$\|\|\mathbf{w}\|\|$	Euclidean norm of $\mathbf{w}$

Chapter 1

Introduction

In recent years the importance of various authentication and security methods increased. Vehicle type recognition, if solved accurately, is beneficial e.g. for building authentications, police camera control systems on crossings to match the number-plate against the car make (car thefts, insurance frauds, etc.) The aim is to obtain reliable classification of a vehicle in the image from a multitude of possible classes using a limited number of examples. Although classification of road going vehicles has been a subject of interest in the past, e.g. traffic control systems and toll-levy automation, vehicle type recognition has not been considered at this level of accuracy. Instead, most systems either detect (classify vehicle and non-vehicle) or classify vehicles in broad categories such as cars, buses, heavy goods vehicles.

1.1 Problem statement

In this thesis we are proposing a method for classification of vehicles into car makes such as Škoda 120, Škoda Favorit, etc. As the input we get an image which contains the car sample from frontal view alone or with other background scenery and other objects. In the second case we expect that a number-plate was correctly detected. As an output we classify the input image sample as a certain car make. We do not expect a high number of classes since there is a limited number of car makes which can observed on Czech roads. We expect the response within seconds, not longer (minutes, hours, ...). Since we expect relatively short response time we chose fast methods for each component of our system. We do not expect occlusion in images expect light occlusion like snowing, etc.

The recognition system proposed in this thesis is based on using a robust SIFT descriptor [25] and its variations that we introduced for feature extraction . Recognition process starts with the car localization in the image. We assume the car number-plate was detected or that the image contains the car image only. In the first case we use the scale and location of the number-plate to define Region of Interest in the image from which the features are extracted. Feature vectors are finally classified using nearest neighbour algorithm and Fisher's linear discriminant. Different system configurations are tested on two databases: database containing car images, database containing truck images.

1.2 Structure of thesis

Chapter 2 gives a short overview of the state of the art, we study the existing methods for vehicle type recognition. We observe two major approaches: model based approach and appearance based approach. It is not our intent to give a complete overview of all of these methods, we rather focus on articles which influenced this work.

Chapter 3 describes the proposed method. We can read here how the region of interest is extracted from the image, how the image is geometrically and photometrically normalized, how the feature vector is extracted from the sample image and what classifiers we used.

Chapter 4 describes the experiments we performed. The chapter starts with database description and continues with a description of parameters used in experiments, i.e. SIFT representations, classifiers and error functions used.

Chapter 5 validates the recognition system we developed with all of its parameters. The recognition system is evaluated against image blur, image added noise and learning set reduction.

Chapter 6 presents the conclusions of our work.

Chapter 2

State of the art

In general, we can distinguish between two approaches: model based (using stereo vision cameras) and appearance based. We choose the latter one approach.

The classical, model-based approaches to object recognition start from an explicit 3D model of an object's shape. 2D features or primitives such as lines, holes, circular segments, etc. are extracted from the image and matched to the 3D model of the object, taking into account the projection from 3D to 2D. Alternatively, 3D volumetric primitives can be extracted directly from the image to be matched with various models. Often, graphs are used to represent the spatial relationship between features. Recognition in those cases is a question of (sub)graph matching. The main drawback of the model-based approaches is that they firmly rely on the feature extraction, which is often vulnerable, error-prone step. Also, this approach is only feasible for special object classes, composed of primitives that allow easy and robust detection.

While the model-based approaches can be considered as being more object-centered, the appearance-based approaches can be thought of as being more viewer-oriented. Another difference is that, in general, the appearance-based techniques exploit to a far greater extent the available photometric information (sometimes even exclusively) while the model-based approaches focus on the geometric entities present in the image. As a last distinction, one can point out that the appearance-based methods are traditionaly more empirical, while the model-based approaches try to analytically model the relation between 3D object features and their projections in the image[10].

2.1 Appearance Based Methods

2.1.1 Approach of T. Kato, Y. Ninonmiya and I. Masaki [17]

In this paper the *multi-clustered modified quadratic discriminant function*(*MC-MQDF*) is used to recognize cars and non-cars images. However the proposed method is capable to learn and classify into various kinds of vehicles: passenger vehicles, commercial vehicles, etc. The MC-MQDF is capable of estimating the complex distribution due to the variety of different possible appearances for preceding vehicles. In a complex distribution test including a variety of vehicles the classification rate for MC-MQDF was approximately 98% while using of the ordinary MQDF resulted in 93% success.

The recognition process can be described as follows:

Learning process.

- *Preparation of Training Samples:* Vehicle and non-vehicle training samples are extracted from a selection of images in order to provide learning classifiers for vehicle class and non-vehicle class images. The vehicle samples are images cropped by the rectangular windows, which are tangetial to the outlines of vehicle images. The non-vehicle samples are cropped by the rectangular windows which are not tangetial to the outlines of vehicles.
- *Feature Extraction:* First, the resolutions of all the samples are normalized, since the sizes of the vehicles are not identical to one another. After normalization of the size, all the feature vectors are projected into feature space.
- *Learning Classifiers:* A classifier is estimated from the distribution of the training samples for each of the classes. Each classifier has a distribution model that containes normal distribution.

Recognition Process.

- *Extraction of an image:* An image is cropped from a window in an image of a road scene. The window moves around in the image in order to find a vehicle.
- *Feature Extraction:* Each cropped image is converted to a feature vector, which is the same as in the learning process.
- *Classification:* Each feature vector is classified as a vehicle or a non-vehicle by the classified from the learning process. If an image is classified as a vehicle, a vehicle must exist at the corresponding window in the road scene.

Feature Extraction Feature extraction was done as follows: all images were normalized to 16x16 image. The feature vector was constructed from the grayscale image using the intensities.

Classification.

In *MC-MQDF* the clustering starts with the dimension parameter $k = 0$, and it is repeated until the goal dimension by increasing k sequentially. For each dimension, the *k-means clustering* is applied to the training samples with a dynamically changing *MQDF* metric, which is changed by recomputing of the mean and covariance matrix from the new members of each cluster. The *MC-MQDF model estimation* procedure can be described as follows.

Step 1. Obtain n initial cluster centroids. (These are chosen at random from training samples in the paper).

Step 2. Start the clutering with the dimension parameter $k = 0$.

Step 3. Compute the mean and covariance matrix for each cluster from the corresponding cluster members.

Step 4. Increase the dimension k by an arbitrary step Δk

Step 5. Re-assign each sample to clusters with the MQDF metric.

Step 6. Re-compute the mean and the covariance matrix for each cluster from its new members.

Step 7. Repeat from *Step 5–7* until all cluster members finish changing.

Step 8. If dimension k reaches to goal dimension, the MC-MQDF model estimation is completed. Otherwise return to *Step 4*.

Database Description.

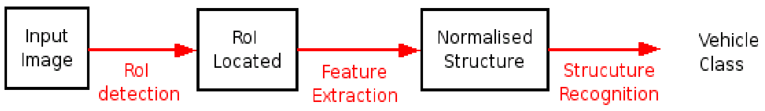

Figure 2.1: Recognition process presented in [19]

The database used in this paper had 5000 entries. These images were obtained from videotapes, which were recorded while a vehicle containing an onboard camera drove on highways and on roads in urban areas during daylight hours. The camera was 1/2 CCD monochrome camera with a 16mm-focal length lens, and the captured images consisted of 320x240 pixels with a 256-level gray scale. The evaluation images consist of three types: i) passenger vehicles (sedans, coupes, station wagons and vans); 2) commercial vehicles (buses, trucks, and dump trucks); and 3) motorcycles; In some scenes more then two vehicles were present. The number of motorcycles was very small in comparison to the others. We can see the concept of the proposed method:

Evaluation Method.

In order to reduce the computation time, the evaluations were carried out using test samples consisting of vehicle samples and non-vehicle samples, instead of classifying with a combination of all locations and sizes. Scenes to crop the training samples and the test samples were chosen such that they did not overlap. The training samples for the vehicle were drawn at random from the samples collected. The test samples for the vehicle class were drawn from the remaining images. For the vehicle class, ten training samples were cropped from each vehicle. The test samples were cropped from windows for which approximate locations were given. The training and test samples for the non-vehicle class were cropped at random from the corresponding road scenes, from which the training and test samples for the vehicle were cropped. The sizes of the samples were normalized to 16x16 after cropping the samples from scenes.

Obtained Results.

The maximum classification rate for MC-MQDF for passenger vehicles, commercial vehicles and motorcycles were 99.3%, 97.2%, and 95.6% respectively. The best classification rate for the MC-MQDF was 97.7%.

2.1.2 Approach of V.S. Petrović, T.F. Cootes [19]

This paper is closer to our work because it tries to classify into car makes. Car make here is e.g.: Peugot 406, Ford Puma, Mercedes Class A, etc.

Recognition Process Description.

The system proposed in this paper is based on the principle of locating, extracting and recognising normalized structure samples taken from a reference image patch on the front of the vehicle (figure 2.1 on page 10). The process starts with locating a reference segment on the object (in this case number-plate) and defining a *Region of Interest* (*RoI*) relative to it. The RoI is processed by the feature extraction element to define a normalized sample of the structure within it. The structure is expressed in a feature vector of pre-defined length that is representative for the vehicle identity. Finally, simple nearest neighbour classification is used to determine the vehicle type associated with each vector.

RoI Detection.

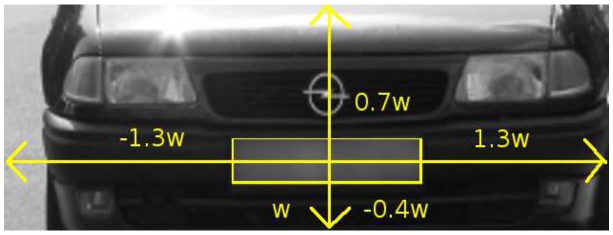

Figure 2.2: The image normalization in [19] was done using this geometry (the image is from our database)

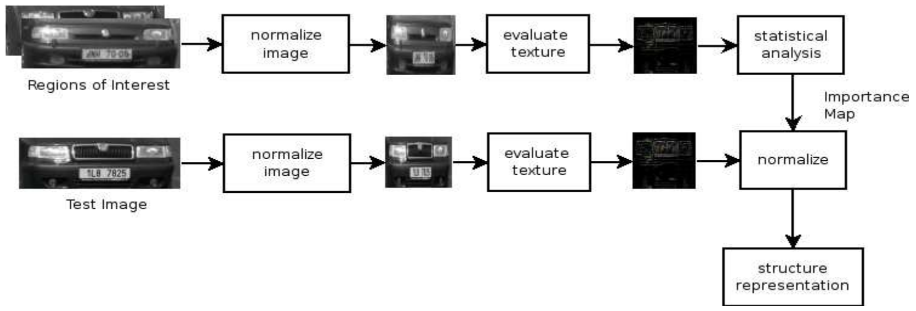

Figure 2.3: Feature Extraction in [19]

The location and scale of a reference structure on the object defines a reference frame for the region of interest to be sampled. An RoI defined relative to the number-plate is thus independent on the actual location and scale of the vehicle in the image. Number-plates are assumed to be highly regular rectangles. To locate then the system in this paper finds all possible right-angle corners suing suitably tuned, separable gradient filters. A hierarchical algorithm for aggregation of corner points into valid rectangular constallations is used to generate hypotheses for the plate location in the image. A number of scale and aspect constraints are used to remove unsuitable candidates, many caused by the characters on the plate and regular vehicle structure features, and of the remaining candidates the one with best corner structure to fit to each of its corners is chosen. The *Region of Interest*(*RoI*) is defined relative to the number-plate coordinates. the region of $\pm 1.3w$ in width and $-0.7w, 0.4w$ from the number-plate center is used, where w is the number-plate width (figure 2.2 on page 11).

Feature Extraction. Feature extraction from the Region of Interest provides a structure representation used to recognise the object, The approach used in vehicle type recognition is illustrated in figure 2.3 on page 11. Initially the RoI is down-sampled to a desired fixed resolution NxM (preceded by suitable smoothing). For vehicle images, horizontal resolution is more severily reduced as the structure is signifficantly more redundant in that direction.

Quite a lot of feaure extraction algorithms were used leading to various image representations. All were obtained by performing the given transformation at each pixel except spectrum phase, which is the phase of FFT.

- raw image

- Sobel edge response (s_x, s_y)
- edge orientation $(\arctan \frac{s_x}{s_y})$
- direct normalized gradients $(\frac{s_x}{\sqrt{s_x^2+s_y^2}}, \frac{s_y}{\sqrt{s_x^2+s_y^2}})$
- locally normalized gradients
- square mapped gradients $(\frac{s_x^2-s_y^2}{s_x^2+s_y^2}, \frac{2s_xs_y}{s_x^2+s_y^2})$
- Harris corner response
- spectrum phase(FFT) $\phi = \arg FFT(I')$

Gradients are normalized between $(-\pi, +\pi)$. Further robustness was obtained by restricting the orientation between 0 and π. As an alternative to direct mapping PCA was considered to determine a low dimensional, optimal structure representation. In order to improve recognition performance additional normalization of structure samples is used. The goal is to emphasize areas of the rigid structure that exhibits the greatest variation between different classes.

Classification.

Two distance measures were investigated to compare test and registration samples, the *dot product* $d = 1 - \mathbf{f}_1^T\mathbf{f}_2$ and the *Euclidean distance* $d = |\mathbf{f}_1 - \mathbf{f}_2|$. The identity of the test sample was then determined using the nearest neighbour rule. These two measures gave similar results, the dot product slightly outperformed the Euclidean distance.

Database Description.

The database contains over 1000 images and 77 different classes. The images are 640x480 color pixels. The distance is in average ≈1.2m. The camera was not fixed and there is a significant variation in both scale and in-plane rotation $(\pm 5°)$

Obtained Results.

The best classification performance with probability of right identification 97.7% was obtained using square mapped gradients. These results were collected using manual RoI detection. When used with automatic car detection system, 93.3% were classified correctly.

Another approach is presented in [20]. Car detection technique based on multi-cues in still outdoor images is presented here. On the bottom level, two area templates based on edge cue and interest points cue are first designed, which can reject most of non-car sub-windows. On the top level, both global strcuture cue and local texture cue are considered. To character the global structure property the odd *Gabor moments* are introduced and trained by *Support Vector Machines* (*SVM*). The multi channels even Gabor based local texture property extracted from corner area is modeled as a Gaussians distribution. The final experiment results show that the integration of global structure property and local texture property si more powerful in discrimination between car and non-car objects and a detection rate of 93% was obtained. The database contains 1000 negative, 550 positive samples and 170 test images.

In [35] an approach is presented to vehicle-class recognition from a video clip. Two concepts are introduced: probes consisting of local 3D curve-groups which when projected into video frames are features for recognizing vehicle classes in video clips; and Bayesian recognition based class probability densities for groups of 3D distances between pairs of 3D probes. The most stable image features for vehicle class recognition appear to be image curves associate with 3D ridges on the vehicle surface. These ridges are mostly those occurring at metal/glass interfaces, two-surface intersections such as back and side, and self occluding contours such as wheel wells or vehicle-body apparent contours, i.e., silhouettes. There are other detectable surface curves, but most do not provide useful discrminatory features, and many of these are clutter, i.e., due to reflections from the somewhat shiny vehicle surface. Models are built and used for the considerable variability that exists in the features used. A Bayesian recognizer is then used for vehicle class recognition from a sequence of frames. The ultimate goal is a recognizer to deal with essentially all classes of civilian vehicless seen from arbitrary directions, at a broad range of distances and under the broad range of lighting ranging from sunny to cloudy. Experiments are run with a small set of classes to prove feasibility. This work uses estimated knowledge of the motion ans position of the vehicle. One way of inferring that information which uses 1D projectivity invariance is indicated. Recognition rate achieved is 88% which is comparable to [20].

In [36] a method for recognizing vehicle classes using computer craphics (CG) is described. In previous work this group developed a vehicle recognition system based on local-feature configuration, which is a generalization of the eigen-window method. This system could recognize one vehicle class very accurately, but there have been limitations in recognizing several classes, when they were quite similar to each other. In this paper[36] the improvements of this recognition system are descibed to distinguish four classes, namely sedan, wagon, mini-van and hatchback. The system requires training images of all target vehicle classes. These training images are easily created using a 3-dimensional computer graphics (3D CG) tool. CG was used as training images since it dispenses with much of the trouble of collecting real training images. Outdoor experimental results have shown that this recognition system can classify in real images with an accuracy of 83% but quite small training and testing set are used: 50 images for training set and 16 images for testing.

The paper [32] describes a method for recognizing the classes of street-parking vehicles. They combine two already developed systems: vehicle recognition system based on local feature configuration, and the other is detecting street-parking vehicles from side-view range images. These two systems are combined here to develop a new system with which it is possible not only count the number of street-parking cars but also to recognize their class of vehicle type such as sedam, wagon, mini-van or so. This system can recognize four vehicle classes: sedan, wagon, mini-van and hatchback. Outdoor experimental results have shown the accuracy of 79%. The database was extended in comparison to [36] to 34 testing images.

In [34] an approach for learning to detect objects in still gray images that is based on a sparse, part-based representation of objects is presented. A vocabulary of information-rich object parts is automatically constructed from a set of sample images of the object class of interest. Images are then represented using parts from this vocabulary, along with spatial relations observed among them. Based on this representation, a learning algorithm was used to learn to detect instances of the object class. The framework developed here can be applied to any object with distinguishable parts in relatively fixed spatial configuration. Experiments shown here reports robustness to partial occlusion and backgound variation. In addition, solutions to several methodological issues that are significant for the research community to be able to evaluate object detection approaches are discussed and offered here.

2.2 Model Based Methods

In paper [44] the classifiaction into car types is solved. Car types are : sedan, pick-up, hatchback, wagon and van. The database had 393 vehicles and only 32 vehicles were misclassified which yields the classification rate of 91%. The paper proposes a segmentation algorithm using deformable template models to segment a vehicle of interest both from the stationary background and other moving vehicles in an image sequence. A polygonal template is defined to characterize a general model of a vehicle and a priory probability density function is derived to constrain the template to be deformed within a set of allowed shapes. The segmentation problem is reduced to a minimization problem and solved by a *Metropolis algorithm*.

2.3 Summary

We saw that appearance-based approaches start from what can be actually seen in the picture, they are not limited to the specific object classes and that they can deal with very complex objects like faces or a cars. Model-based approaches use some additional information such as shapes which were pre-defined for the recognition task.

In the above articles various methods were presented. We saw that almost every method proposed above contains the following procedure: locating the object in an image, then using feature extraction on this so called Region of Interest (RoI), and finding the approprite class from the extracted features.

The database contained at least tens of images and at most thousands of images. Also detection from videoclip was used. In [27] was presented how to use the boosting methods to detect a pedestrian to create huge database used for other related work.

We can see from the articles briefly described above that we should expect our classification rate into car makes between 80–97%.

Chapter 3

Proposed method

In Chapter 2 we saw that we can expect a classification rate between 80–97%. We will try to classify vehicles according to car make like Skoda Fabia, Skoda Felicie, Daf, etc.

The recognition process can be described as follows (figure 3.1 on page 15):

- *Extraction and Geometrical Normalization:* Vehicle samples are extracted from images in order to provide learning classifiers for vehicle car makes. The Region of Interest (RoI) is extracted from every image sample and it is normalized to MxN resolution.
- *Feature Extraction:* Every RoI is processed by feature extraction algorithm. Feature vectors are projected into feature space.
- *Learning Classifiers:* We used k-NN, and FLD as classifiers and two different metrics for k-NN

3.1 Structure of Proposed Method

3.1.1 Frontal Mask Localization and Geometric Image Normalization

We have two types of images:

- images with frontal mask only

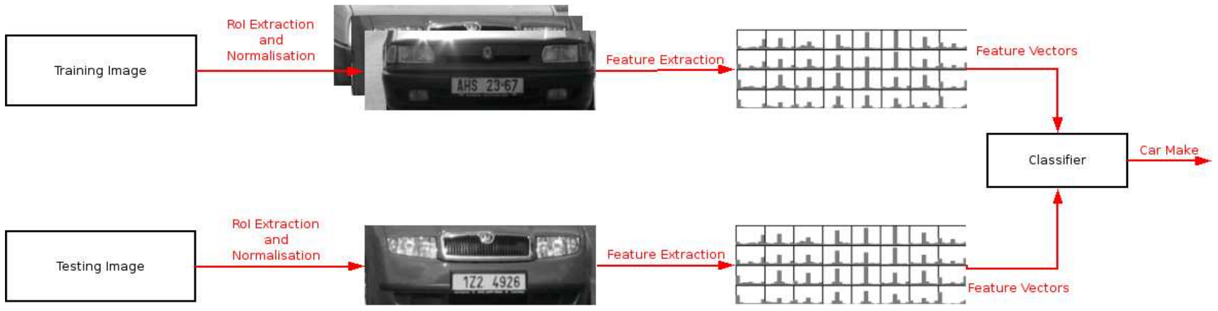

Figure 3.1: Proposed Classification Process

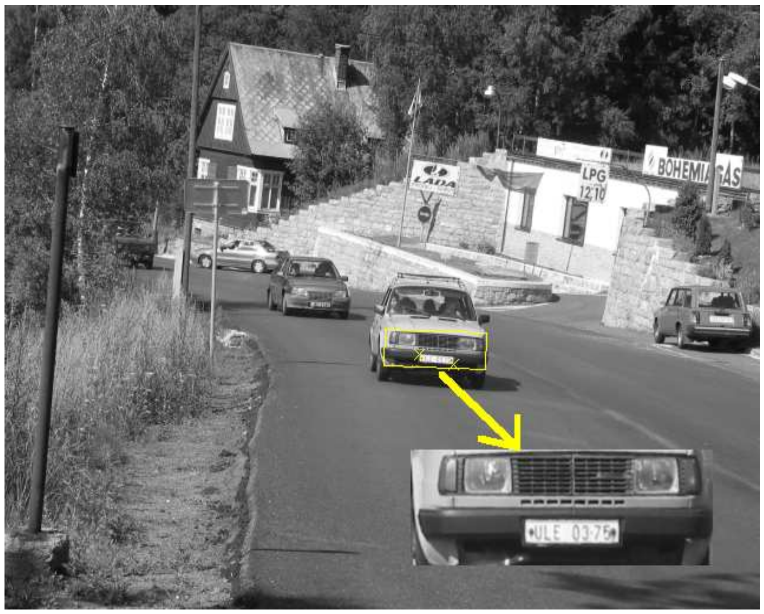

Figure 3.2: The number-plate is detected first. Then the surrounding rectangle is cut from the image

- images with scenery containing cars

The first case is easy, the only process here is to normalize an image. This is the case of truck images. For car images we need to identify the Region of Interest, given coordinates of number-plate.

Locating Car Frontal Mask

For database with car pictures we expect that the number-plate is correctly detected. We have two coordinates for every image with top-left corner of number-plate and bottom-right corner of number-plate.

Then we can establish orthogonal geometry with the origin in the top-left corner of number-plate. Then we cut a region surrounding the number-plate as depicted on figure 3.2 on page 16.

Geometric Image Normalization

All images are geometrically normalized to 256 pixels in width and 92 pixels in height. The geometry in the car image is measured via basis vectors relating to the number-plate as we can see in figure 3.3 on page 17. The license plate has a width of 81 pixels and a height of 21 pixels in the image referred as w or h respectively.

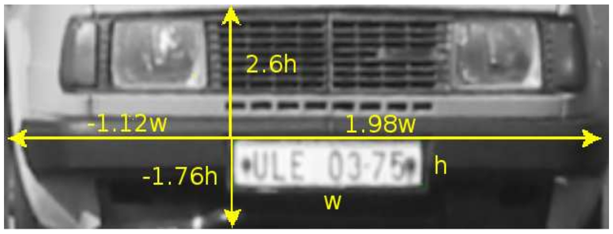

Figure 3.3: Car geometry is measured in the space of number-plate

3.1.2 Feature Extraction and Photometric Normalization

Now we need to do the feature extraction from the frontal car image. In paper [14] the performance of interest point descriptors is evaluated. Many different descriptors have been proposed in the literature. However, it is unclear which descriptors are more appropriate and how their performance depends on the interest point detector. The descriptors should be distinctive and at the same time robust to changes in viewing conditions as well as to errors of the point detector. The following descriptors are compared: *SIFT* descriptors[25], *steerable filters*[40], *differencial invariants*[41], *complex filters* [42], *moment invariants*[43] and cross-correlation for different of interest points. The evaluation in [14] uses as a criterion detection rate with respect to false positive rate and is carried out for different image transformations. This is the standard *Receiver Operating Characteristics* (*ROC*). Two points **a** and **b** here are similar if the distance between their descriptors is below an arbitrary threshold $d_M(D_a - D_b) < t$, the value of t is varied to obtain the ROC curves. Given two images representing the same scene the detection rate is the number of correctly matched points with respect to the number of possible matches (*true positive rate*):

$$p_{correct} = \frac{\#correct\ matches}{\#possible\ matches}$$

The *false positive rate* is the probability of a false match in a database of descriptors. Each descriptor of the query image is compared with each descriptor of the database and the number of false matches is counted. The probability of false positives is the total number of false matches with respect to the product of the number of database points and the number of image points:

$$p_{false} = \frac{\#false\ matches}{(\#database\ points)(\#query\ image\ points)}$$

The test covered several experiments: rotation, scale changes, affine transformations and illumination changes. In this evaluation, it was observed that the ranking of the descriptors does not depend on the point detector and that SIFT descriptors perform the best in all tests except for lighting illumination change where steerable filters performed better (but they both had a very good and comparable quality). Steerable filters in overall summary come second; they can be considered a good choice given the low dimensionality. Based on this article we chose SIFT. We modified the SIFT for 3 types of representation which will be described later. We simplified the SIFT feature extraction and we will show that the results are comparable. However the easiest SIFT did not perform as good as the SIFT proposed in [25] the algorithm simplification lead to a faster feature extraction. SIFT with overlapping regions was also tried and it performed best. Features similar to those displayed on figure 3.5 on page 18 are obtained. These features serve as an input to the classification learning process. The image is normalized photometrically since part of SIFT feature extraction algorithm is the vector normalization to unit length.

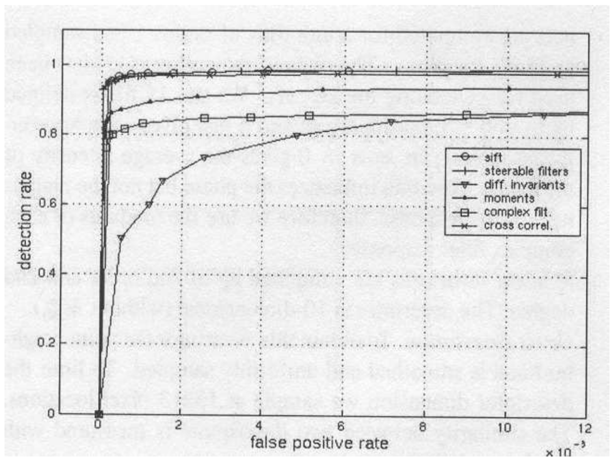

Figure 3.4: ROC curve for evaluated descriptors in [14], here for an rotated image

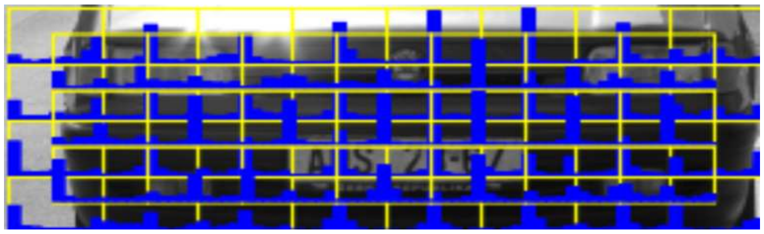

Figure 3.5: Feature representation used for learning

3.1.3 Classification

The vehicle classification process is not an easy problem. Truck vehicles are very similar to each other and moreover they contain a lot of confusing information (like company name, driver name, etc.). Images of trucks also have a signifficant shift in position and they were taken under difficult weather conditions like snowing or during night and with lights on. Images of cars have a lot of reflections from metal surface and quite big variance within class.

Statistical pattern recognition approaches for pattern detection generally fall into two categories: *generative methods* or *discriminative methods* depending on the estimation criteria used for adjusting the model parameters and/or structure.

Generative approaches such as hidden Markov Model, Markov random field or naive Bayes classifier relies on probability distribution estimation over examples using maximum likelihood or maximum a posterior methods. Discriminative methods such as neural networks, LDA or SVM are trying to find a decision boundary between classes. Since generative mixture methods such as a mixture of Gaussians and a mixture of factor analyzers rely on a joint probability distribution over examples it carries some advantages like handling incomplete examples, the typical estimation criterion is nevertheless optimal from the classification viewpoint. Furthermore generative approaches usually require larger data set in comparison to discriminative methods since most of them are focusing on covariance matrix estimation. Discriminative methods that focus directly on the parametric decision boundary, e.g. *Fisher's linear discriminant(FLD)* or SVM, typically yield a better classification results[28].

We decied to try these classification algorithms:

- k-NN (Nearest Neighbor)
- Fisher's Linear Discriminant (FLD)

The first one, *k-NN* (*Nearest Neighbor*), serves us as a baseline, it is easy to implement and the results are quite good. We will tune the SIFT parameters using this algorithm. We will also show how our different SIFT representations can affect the classification rank. *Fisher's Linear Discriminant* is a robust method with quite a small amount of parameters which usually gives good results.

We investigated two distance measures to compare testing and training samples, the Euclidean *Euclidean distance* and additionally also *Earth Mover's Distance* (*EMD*).

3.2 Detailed description of components

3.2.1 Distinctive Image Features from Scale-Invariant Keypoints — SIFT

The technique we are going to describe was proposed by D. Lowe in [25]. The following are the major stages of computation in image feature extraction:

- **Scale-space peak selection:** identify points of interest
- **Keypoints localization:** keypoints are selected based on measures of their stability
- **Orientation assignment:** one or more orientations are assigned to each keypoint based on local image properties
- **Keypoint descriptor:** local image gradients are measured at the selected scale in the region around each keypoint and transformed into a representation that allows for local shape distortion and change in illumination.

We made a simplification of this method. We treat every region of interest as a keypoint.

Features extraction

Orientation and gradient assignment

We start with orientation $\phi(x, y)$ and gradient $g(x, y)$ assignment for every pixel (x, y) in sample image I. Let $I(x, y)$ is a pixel intensity at position (x, y) in image I:

- $\forall(x, y) \in I$:

$$d_y = (I_{x-1,y-1} + 2I_{x,y-1} + I_{x+1,y-1}) - (I_{x-1,y+1} + 2I_{x,y+1} + I_{x+1,y+1})$$
$$d_x = (I_{x+1,y-1} + 2I_{x+1,y} + I_{x+1,y+1}) - (I_{x-1,y-1} + 2I_{x-1,y} + I_{x-1,y+1})$$
$$\phi^*(x, y) = \arctan d_y/d_x$$
$$g(x, y) = \sqrt{d_y^2 + d_x^2}$$

We would like to obtain the same features for image (a) and (b) in figure 3.6 on page 20. We need this this result since we want to cope with various metal surface colors from dark colors to bright colors. This is the reason we assign to $\phi(x, y)$ the orientation modulo π:

$$\phi(x, y) = \pi + \phi^*(x, y) \pmod{\pi}$$

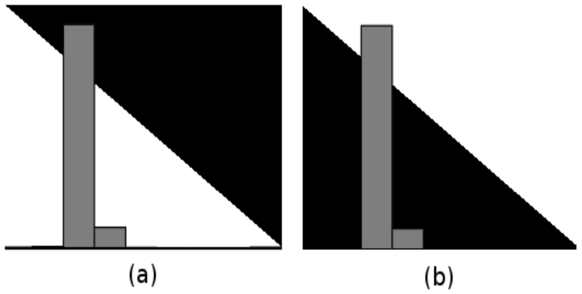

Figure 3.6: obtained orientation is modulo π to obtain the same results for (a) and (b)

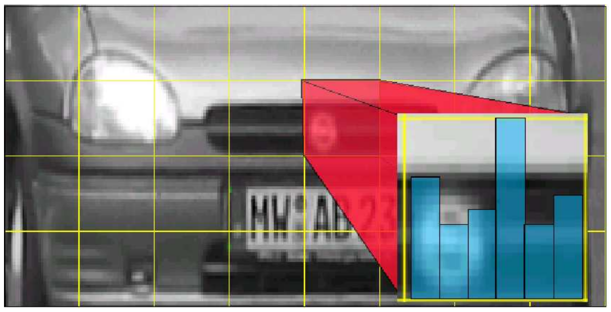

Figure 3.7: Features Extraction

The local image descriptor representations

The previous orientations have assigned an orientation and gradient for every pixel in the image. The next step is to compute the descriptor which is distinctive yet is as invariant as possible to parameters such as change in illumination, shift in vertical or horizontal direction or viewpoint change. One usual approach would be to sample local image intensities around the keypoint at appropriate scale and to match these using a normalized correlation measure. However, simple correlation of image patches is highly sensitive to changes that cause misregistration of samples, such as affine or 3D viewpoint change.

As we can see on figure 3.7 on page 20 the image is divided into several square patches (tiles). Every square patch is divided into certain number of bins. Every bin represents a certain interval of orientations which are assigned to it. We call this interval a direction. We can number bins from 1 to b, where b is the number of bins for each tile (i.e. number of directions for each tile). Similarly we can number tiles from 1 to t, where t is the number of tiles.

How the values are assigned to every bin depends on the SIFT representation. We proposed 3 variants:

SIFT-Ori

This is the easiest representation. Every orientation has the same vote, which equals to one. Given the i-th tile T_i we can compute values for every j-th bin as:

$$\forall (x, y) \in T : u_{ij} = |\{(x, y) : (x, y) \in T_i \land \phi(x, y) \in (j * \pi)/b\}|$$

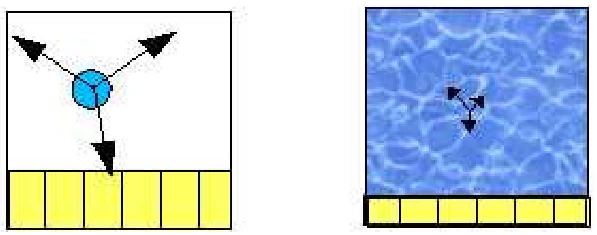

Figure 3.8: Possible distinction between random texture and circle-like shapes

where i ranges from 0 to $b-1$.

With this approach we can have problems when there is the same probability for the direction in the tile. This can apply for random texture but also for circle-like shapes. If we want to distinguish between these two cases we should use *votes based on gradient magnitude (SIFT-Grad).*

SIFT-Grad

This can be more distinctive between the *random textures* and *circle-like shapes* (figure 3.8 on page 21). Random textures have usually small local gradients. We can use this information to achieve the similar result as displayed in the picture. We do not want the histograms to be the same (as they would be when each direction has the same vote weight). We can choose the vote weight correlated to the gradient magnitude. Given a tile T_i we have:

$$u_{ij} = \sum_{\{(x,y):(x,y)\in T_i \wedge \phi(x,y)\in(i*\pi)/b\}} g(x,y)$$

SIFT-GradWei

This representation can deal better with shifts of the tile. Given the i-th tile T_i and c_x, c_y is the center of the tile:

$$u_{ij} = \sum_{\{(x,y):(x,y)\in T_i \wedge \phi(x,y)\in(i*\pi)/b\}} \frac{1}{d(x,y)} g(x,y)$$

where $d(x,y)$ is computed as:

$$d(x,y) = \sqrt{(x-c_x)^2 + (y-c_y)^2}$$

$d(x,y)$ is the distance of point (x,y) from the center.

Overlapping tiles

This lead to significant improvement in classification process. The square patches in the SIFT descriptor described above did not have the overlap with other tiles. The areas occupied by the square patches were disjunctive. The overlapping tiles means we have additional layer of square patches which are laied over the original layer.

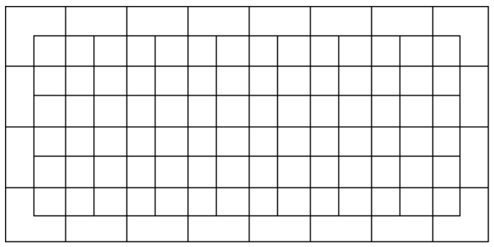

Figure 3.9: Overlapping Tiles

Feature vector construction

We obtained u_{ij} for each i-th tile and j-th. Let the number of bins is b and the number of tiles is t. Then our feature vector μ of image I is:

$$\mu = (\mu_1, \mu_2, \dots, \mu_{b*t})$$

where μ_k is computed as:

$$\mu_k = u_{\lfloor k/b \rfloor, k \pmod b}$$

The dimension of vector μ depends on number of tiles t and number of bins b of the SIFT descriptor and is $b * t$.

3.2.2 Error function

We need to somehow measure the similarity or dissimilarity of two vectors in feature space. In the previous section we defined a feature vector. By comparing the distance of two feature vectors we will have one possible solution for measurement.

The choice of *error function* could have significant impact on the classification rate. We will utilize *Euclidean distance* error function and also show *Earth Mover's Distance* algorithm.

Error function is a transformation from a vector space to real numbers:

$$T : V^n \times V^n \to \Re \tag{3.1}$$

where V^n is a vector space of dimension n.

Euclidean distance

When we want to measure the dissimilarity of two vectors we can use the Euclidean distance of these. The error is then computed as the distance of two vectors we want to compare, let $\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$ are such vectors then the error function is computed as their Euclidean distance:

$$e_{\mu_1,\mu_2} = ||\boldsymbol{\mu_1} - \boldsymbol{\mu_2}||$$

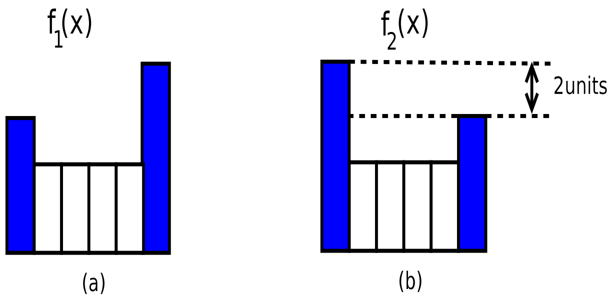

Figure 3.10: These two distributions are very close to each other, but their Euclidean distance is quite large.

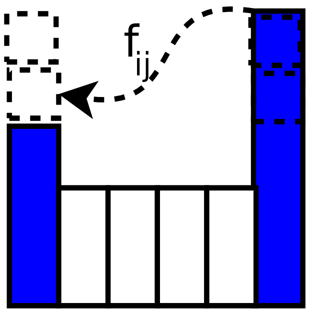

Figure 3.11: Here is depicted how mass units are transformed from distribution $f_2(x)$ to $f_1(x)$.

Earth Mover's Distance — EMD

The *Earth Mover's Distance*(*EMD*) was introduced in [8]. The problem of *Euclidean distance* is that when we have features such as SIFT, then the two *direction distributions* are almost the same or at least very similar to each other (see figure 3.10 on page 23).

The *Earth Mover's Distance* is the minimum amount of work needed to transform one distribution to another one. We assume that both distributions have the same mass, in our case:

$$\int_{-\infty}^{+\infty} f(x)dx = 1$$

In the case on figure 3.10 on page 23 the Euclidean distance would be $2^2+0+0+0+0+2^2 = 4$, the EMD is only 2 (see figure 3.11 on page 23).

What we need to solve is *Transportation Problem*. We have a bipartite graph (figure 3.12 on page 24), one side of nodes represents consumers (C), the second one producers (P). We want to find such a flow f_{ij} which minimizes the overall cost c_{ij} between transporting mass from producers to consumers:

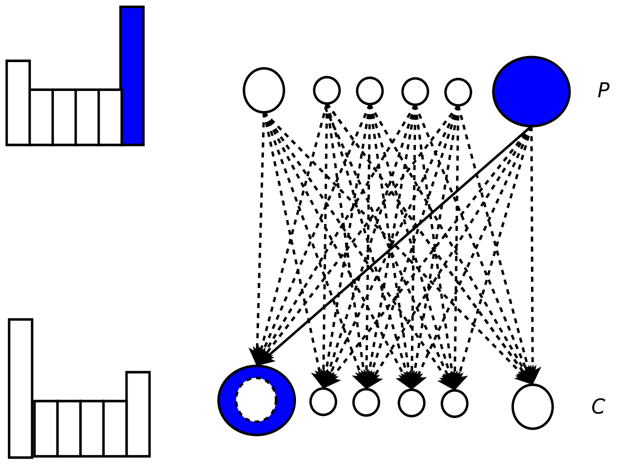

Figure 3.12: Transportation Problem relation to EMD. The solid line shows the minimum workflow of 2 mass units

$$\sum_{i \in P} \sum_{j \in C} f_{ij} c_{ij} \tag{3.2}$$

In our case c_{ij} is defined as:

$$c_{ij} \equiv \begin{cases} i - j + N & \pmod N \quad , i - j \leq 0 \\ i - j & \pmod N \quad , i - j > 0 \end{cases} \tag{3.3}$$

which reflects that the very left and the very right bins are neighbors.

3.2.3 Classifiers used

k-NN — The Nearest Neighbor

Let us assume we have sets D_i, these represent c classes, $k > 0$ and test sample $\mathbf{x}$. We want to classify $\mathbf{x}$ as a member of one of classes D_i, *k-NN* does this very simply (figure 3.13 on page 25):

- find k closest vectors to test vector $\mathbf{x}$, let these are $\mathbf{r_1}, \ldots, \mathbf{r_k}$
- make a hypercycle C_n around $\mathbf{x}$ with radius r, $r = \max_{i=1,\ldots,k} |\mathbf{r_i}|$
- classify $\mathbf{x}$ as D_i:

  $S_i \subseteq D_i$

  $S_i = argmax_{R_i} |R_i|$

  $R_i = \{\mathbf{r_i} : ||\mathbf{r_i} - \mathbf{x}|| \leq r \wedge \mathbf{r_i} \in D_i\}$

The last step says: classify input vector $\mathbf{x}$ as the member of the class which has the majority in hypercycle C_n.

The special case of *k-NN* is *1-NN*, where we are classifying sample $\mathbf{x}$ as the closest vector to $\mathbf{x}$.

The very important thing here is the metric used to find the *closest vectors*. This was discussed in section *Error Function*.

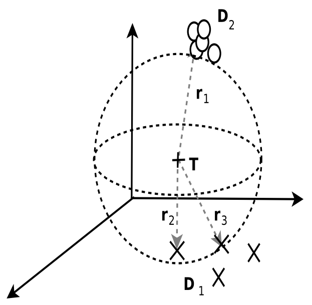

Figure 3.13: Classification using k-NN, $k = 3$. The test sample T is being classified as $\times$, because in the hypercycle surrounding T are 2 elements from $\times$ and only one from $\circ$.

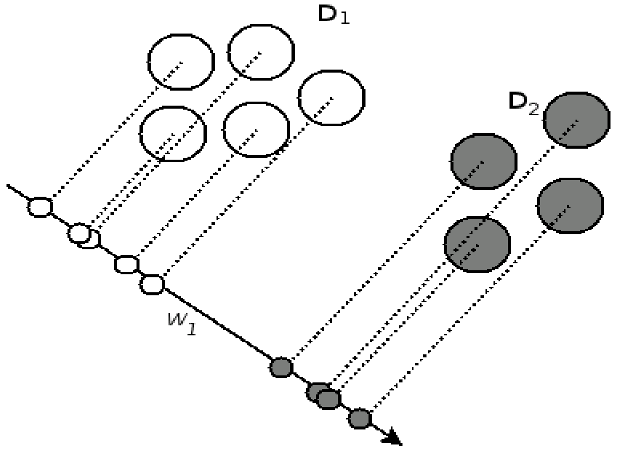

Figure 3.14: Projection from 2D to 1D — we should be able to find some threshold for classification

FLD - Fisher Linear Discriminant

Let us assume we have sets D_i, these represent c classes, each containing n_i elements ($|D_i| = n_i$). We are trying to reduce the dimensionality of input data vectors $\mathbf{x} \in D_i$ is $d, d = dim(x)$ to dimensionality $c - 1$, where we believe the classification will be more straightforward, maybe possible using linear classifier. We want to do the following transformation:

$$T : \Re^d \to \Re^{c-1} \tag{3.4}$$

Fisher's Linear Discriminant is trying to maximize the ratio of *scatter between classes* to the *scatter within classes.*

Two Classes Classification

First let us illustrate this problem in two classes problem. We can see on figure 3.14 on page 25 that the projection on vector w_1 gives us a chance to find some threshold suitable for distinguishing whether a potential test sample belongs to class D_1 or D_2. This could be possible because the projection of D_1 and D_2 on w_1 results in two separate sets. While on figure 3.15 on page 26 this is not possible because the projected class points from D_1 overlap with projected class points from D_2. We can see that some projections on vector 1D space are better here and some are worth.

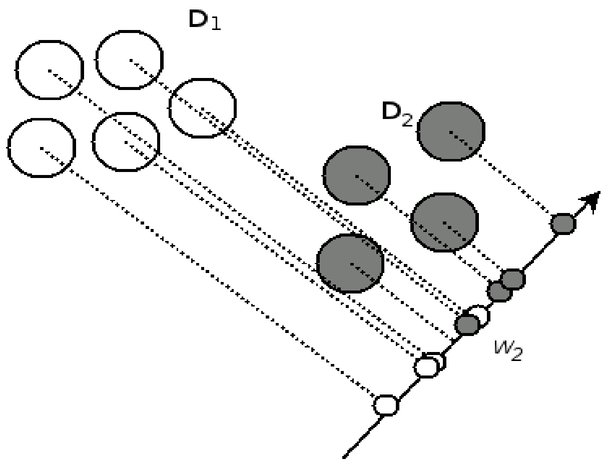

Figure 3.15: Projection from 2D to 1D — not a very good projection for classification purposes

Now take a look at this in more detail. Assume we are trying to project sample $\mathbf{x}$, to scalar value y which corresponds to the metrics of vector $\mathbf{w}$, we are projecting on. This is done by using the following scalar product:

$$y = \mathbf{w}^T \mathbf{x} \tag{3.5}$$

It is straightforward that if $\mathbf{m_{D_1}}$ is the mean vector for class D_1, then its projection on vector $\mathbf{w}$ is:

$$\tilde{m}_{D_1} = \mathbf{w}^T \mathbf{m_{D_1}}$$

The same rule is valid for the scatter. if $\mathbf{s_{D_1}}$ is the scatter matrix for class D_1, then $\tilde{s}_{D_1}$ is scatter projected on $\mathbf{w}$:

$$\tilde{s}_{D_1} = \mathbf{w}^T \mathbf{s_{D_1}}$$

What we want to do is:

- maximize the distance of means in the projected space
- minimize all of the scatters in the projected space

We can see figure 3.16 on page 27 that $\mathbf{w_1}$ does this for us:

We can define the criterion for vector $\mathbf{w}$:

$$J(\mathbf{w}) = \frac{|\tilde{m}_{D_1} - \tilde{m}_{D_2}|^2}{\tilde{s}^2_{D_1} + \tilde{s}^2_{D_2}} \tag{3.6}$$

We want such a $\mathbf{w}$ to maximize the function $J(\mathbf{w})$.

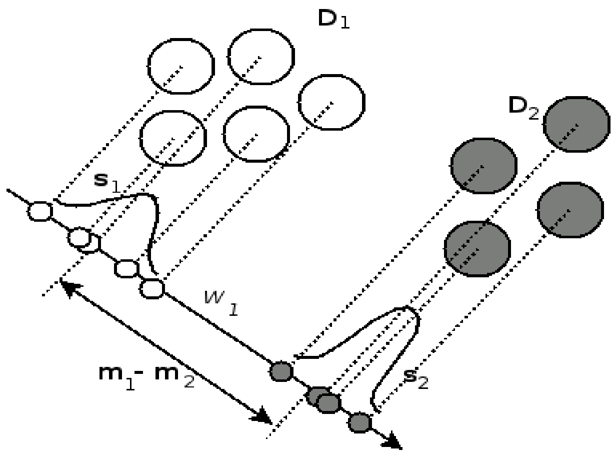

Figure 3.16: The best is to separate the classes — projected distances of m_1 and m_2 should be maximized and projected scatters s_1 and $s2$ should be minimized.

Now we can define *scatter matrices.* First define scatter within classes matrix:

$$\mathbf{S_W} = \sum S_{D_i} \tag{3.7}$$

where $\mathbf{S_{D_i}}$ is defined as:

$$\mathbf{S_{D_i}} = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m_{D_i}})(\mathbf{x} - \mathbf{m_{D_i}})^T$$

We can easily see that:

$$\tilde{s}^2_{D_i} = \mathbf{w}^T \mathbf{S_{D_i}} \mathbf{w}$$

We define *scatter within classes* as the sum of all class scatters. We are trying to find a projection where these scatters are as small as possible (see figure 3.16 on page 27). This is done by $\mathbf{w}^T \mathbf{S_W} \mathbf{w}$.

We define *scatter between classes* for two classes (we will change its definition later in *Multiple Classes Classification* as:

$$\mathbf{S_B} = (\mathbf{m_{D_1}} - \mathbf{m_{D_2}})(\mathbf{m_{D_1}} - \mathbf{m_{D_2}})^T$$

This defines the distance between means of class D_1 and class D_2. We are trying to find a projection where this is as distant as possible (see figure 3.16 on page 27). This is done by $\mathbf{w}^T \mathbf{S_B} \mathbf{w}$.

The maximization of mean distances and minimization of class scatters is done by maximizing $J(\mathbf{w})$ which we can now rewrite as:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S_B} \mathbf{w}}{\mathbf{w}^T \mathbf{S_W} \mathbf{w}} \tag{3.8}$$

In fact equations (3.8) and (3.6) are the same.

Multiple Classes Classification

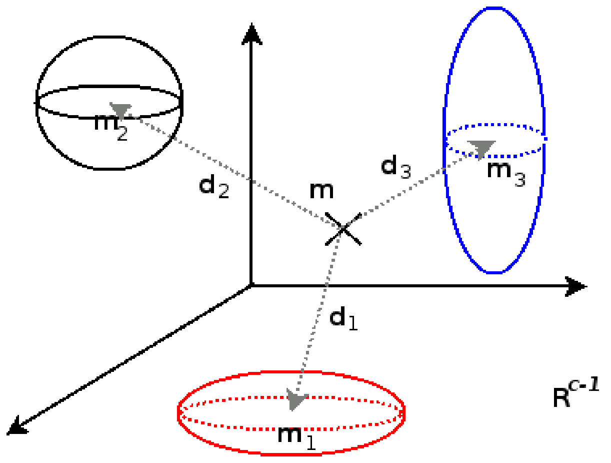

Figure 3.17: The LDA transformation is trying to maximize $d_1^2 + \ldots + d_c^2$ in the space $\Re^{c-1}$

As stated in (3.4) FLD does the dimensionality reduction for us. In multiple classes classification we want to project to space $\Re^{c-1}$, where c is the number of classes we have: $D_1, D_2, \ldots, D_c$.

We will leave the definition of S_W unchanged but we have to change the definition of S_B. We will define *total mean* as:

$$\mathbf{m} = \frac{1}{N} \sum_{i=1}^{c} n_i \mathbf{m_{D_i}} \tag{3.9}$$

which is in fact *center of gravity* of all class means $\mathbf{m_{D_i}}$ and if the mean is computed using the proper statistic[1] $\mathbf{m}$ is the *center of gravity* of all $\mathbf{x} \in D_1, D_2, \ldots, D_c$

Then we can define *scatter between classes* as:

$$\mathbf{S_B} = \sum_{i=1}^{c} n_i (\mathbf{m_{D_i}} - \mathbf{m})(\mathbf{m_{D_i}} - \mathbf{m})^T \tag{3.10}$$

Now what we are going to do is:

$$\mathbf{y} = \mathbf{W^T x} \tag{3.11}$$

where $\mathbf{y}$ is a projection of $\mathbf{x}$ in $\Re^{c-1}$ space, $dim(\mathbf{y}) = c - 1$ and $\mathbf{W}$ is matrix of $(c-1)$ x d.

We are trying to maximize $J(\mathbf{W})$ defined very similarly to (3.8):

$$J(\mathbf{w}) = \frac{\mathbf{W}^T \mathbf{S_B} \mathbf{W}}{\mathbf{W}^T \mathbf{S_W} \mathbf{W}} \tag{3.12}$$

As we can see on figure 3.17 on page 28 LDA transformation is trying to spread classes across $\Re^{c-1}$.

[1]Usualy we assume Gaussian distribution here, so mean for D_i is computed as $\mathbf{m_{D_i}} = \frac{1}{N} \sum_{\mathbf{x} \in D_i} \mathbf{x}$

Solving Fisher's Discriminator

We can mark $\mathbf{w_i}$ as columns of $\mathbf{W}$ in (3.12). Then we need to solve the *generalized eigenvalue problem*:

$$(\mathbf{S_B} - \lambda \mathbf{S_W})\mathbf{w_i} = 0 \tag{3.13}$$

The is the same as:

$$(\mathbf{S_W^{-1}} \mathbf{S_B} - \lambda \mathbf{E})\mathbf{w_i} = 0 \tag{3.14}$$

We can use this by *Singular Value Decomposition* which is implemented in many mathematical programs and libraries.

Once we have the *projection matrix* $\mathbf{W}$ we can do the classification in space of dimension $\leq c-1$ by projecting test sample to this space and finding the maximum for $p(x|D_i)P(D_i)$. Another approach can be using *Nearest Neighbor* in this reduced space.

Implementation

We can use `svd(inv(Sw)*Sb)` function implemented in Matlab or PDL::MatrixOps or ccmath. When using Matlab we can also use `eig(Sw,Sb)` function which can solve the *generalized eigenvalue problem*.

Possible Issues

The main issue here is that $\mathbf{S_W^{-1}}$ can be singular when we have a high-dimensional data (which is usually true for images) and not enough train data supplied. E.g. in our case we have 32 SIFT tiles each has 9 bins which is in result a vector of dimension 288. Thus $\mathbf{S_W}$ and $\mathbf{S_B}$ are matrices 288x288. In an ideal case we would need 288^2 of train pictures[6] to not have $\mathbf{S_W^{-1}}$ degenerated.

The possible solution is to use *Principal Component Analysis*(*PCA*) first[5, 6, 7], but the problem is that vectors containing a very important information could be part of PCA null space[2][5, 6].

[2] also known as transformation kernel

Chapter 4

Experiments

4.1 Database description

We have two databases, the first one contains car pictures and the second one contains truck pictures.

4.1.1 Car images

We collected around 250 images of cars (figure 4.1 on page 31) but we could only re-use around 100 of them since not all images were suitable for car make recognition (e.g. bad resolution, unique cars we cannot match against its peer, etc). These were shot using small digital compact camera by hand in the north of the Czech Rebublic on road E65. Images were taken in various confitions: at dusk, dawn, during bright clear days and overcast conditions. Metal surfaces of cars are smooth and highly reflective. The lens zoom was not fixed so a perspective distortion is present in all photos (focal length between 20mm-105mm) . In-plane rotation is between $0-10^{\circ}$. The original image size was 1600 pixels in width and 1200 pixels in height. We classifed into the following makes: Octavie, Opel, Opel2, Skoda 120, Felicie, Skoda 105, Favorit2, Fiat Uno, Favorit, Fabie.

4.1.2 Truck images

We obtained a lot of truck images(figure 4.2 on page 31) from a border crossing. The camera was mounted on a roof and was using fixed lens zoom. We used about 500 images from the truck database. The pictures from this database were also obtained in various confitions: during bright days, night, overcast days and also during snowing which causes an partial occlussion. We classifed into the following car makes:Daf, Iveco, Man, Mercedes, Renault, Scania and Volvo. All truck images are 1280 pixels in width and 512 pixels in height.

4.2 Parameters

These parameters are important for the classification process:

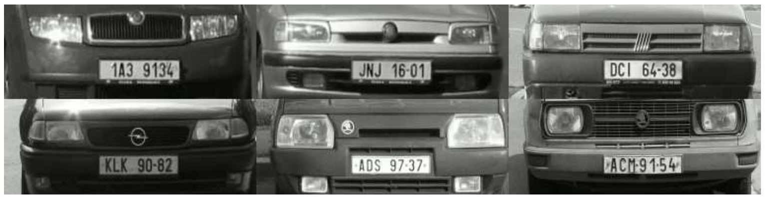

Figure 4.1: Sample car images in our database

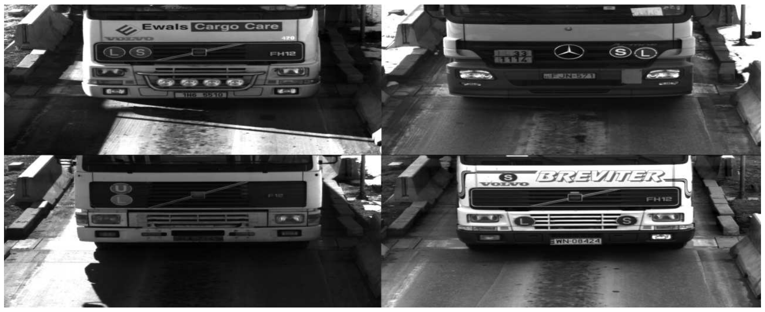

Figure 4.2: Sample truck images in our database

- database (cars, trucks)
- classifier (k-NN, FLD)
- error function (Euclidean, EMD)
- SIFT overlapping
- SIFT representation
  - SIFT-Ori
  - SIFT-Grad
  - SIFT-Wei

4.2.1 SIFT representation

We tried these SIFTs:

SIFT-Ori: Here every direction and gradient votes with equal weight. This is not very discriminative since it cannot distinguish between the black circle on white background and between the random texture where all directions have the same probability.

SIFT-Grad: The gradient magnitude is computed and its direction gets votes related to this magnitude. This turned out to have more discriminative power.

SIFT-GradWei: This one is very similar to the SIFT-Grad but the gradient's magnitudes are multiplied with a coefficient. This coefficient decreases with the increasing distance from the center of square patch.

Overlapping SIFTs: each of SIFTs above can have an overlapping square patches over itself.

These parameters are important for every experiment with SIFTs:

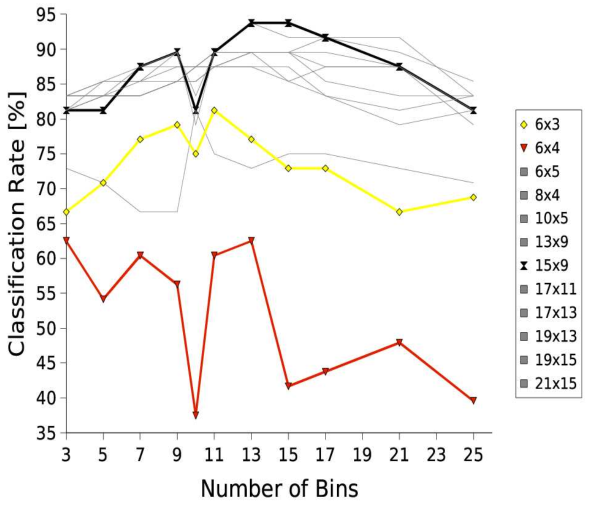

Figure 4.3: Optimal Number of Bins

- number of square patches in horizontal direction (H)
- number of square patches in vertical direction (V)
- number of bins (B)

Will use the term *SIFT topology* by which we mean: number of square patches in horizontal direction, number of patches in vertical direction and number of bins per square patch. We will write this as: HxVxB or HxV if we do not focus on number of bins per square patch.

4.2.2 Used classifiers

Parameters for every classification method:

- k-NN
  - k
  - metric (Euclidean, EMD)
- FLD
  - reduced sub-space dimension
  - classification in projected feature space (1-NN, nearest μ)

4.3 k-NN performed on car database

4.3.1 Optimal SIFT topology

Here we try to detect the peak of classification rate in relation to SIFT topology and number of bins.

	SIFT Topology											
# of bins	**6x3**	**6x4**	**6x5**	**8x4**	**10x5**	**13x9**	**15x9**	**17x11**	**17x13**	**19x13**	**19x15**	**21x15**
3	66.67	62.5	72.92	81.25	83.33	81.25	81.25	83.33	83.33	81.25	83.33	83.33
5	70.83	54.17	70.83	85.42	83.33	83.33	81.25	83.33	83.33	83.33	85.42	85.42
7	77.08	60.42	66.67	87.5	83.33	85.42	87.5	83.33	83.33	87.5	85.42	85.42
9	79.17	56.25	66.67	89.58	85.42	89.58	89.58	85.42	85.42	87.5	85.42	87.5
10	75	37.5	81.25	79.17	87.5	83.33	81.25	85.42	85.42	87.5	87.5	87.5
11	81.25	60.42	75	89.58	87.5	89.58	89.58	87.5	87.5	89.58	89.58	87.5
13	77.08	62.5	72.92	89.58	87.5	89.58	93.75	87.5	89.58	89.58	93.75	89.58
15	72.92	41.67	75	85.42	87.5	89.58	93.75	85.42	89.58	89.58	91.67	89.58
17	72.92	43.75	75	83.33	83.33	87.5	91.67	87.5	85.42	89.58	91.67	91.67
21	66.67	47.92	72.92	79.17	81.25	87.5	87.5	87.5	83.33	87.5	91.67	89.58
25	68.75	39.58	70.83	81.25	83.33	81.25	81.25	81.25	83.33	79.17	83.33	85.42

Table 4.1: Optimal Number of Bins (see figure 4.3 on page 32) and its relation to the classification rate [%]

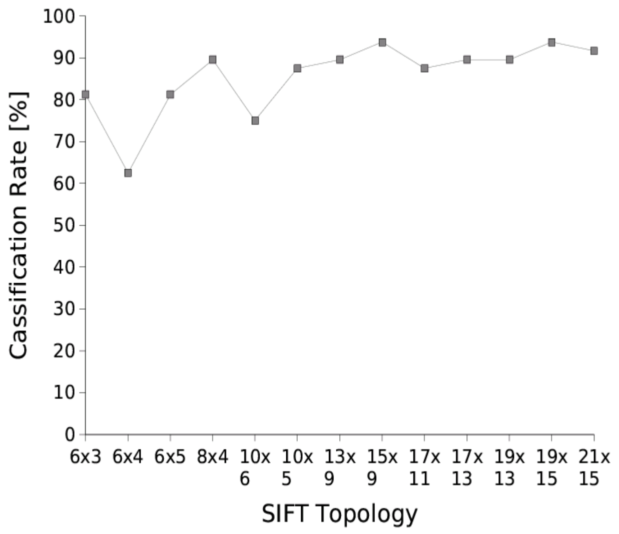

Figure 4.4: SIFT Topology and how this influences the classification rate

We can see that the best classification rate was achieved when the number of bins was between 11–15. The worst case was usually when number of bins was 10 (figure 4.3 on page 32).

Another interesting thing is the overall weakness of classificator with number of bins 10. The surrounding bins 9 and 11 have usually better power.

Another interesting thing is that SIFTs with topology 6x3 and 6x5 are both stronger classificators than the one with topology 6x4 eventhough the classification is better with more detailed classificators (so 6x3 should be worse than 6x4)

We found that the classification rate is getting better as the detail of SIFT topology is increasing up to some level. Then the classification rate is decreasing (figure 4.4 on page 33).

SIFT Topology	**Classification Rate [%]**
6x3	81.25
6x4	62.50
6x5	81.25
8x4	89.58
10x6	75.00
10x5	87.50
13x9	89.58
15x9	93.75
17x11	87.50
17x13	89.58
19x13	89.58
19x15	93.75
21x15	91.67

Table 4.2: SIFT Topology and how this influences the classification rate(see figure 4.4 on page 33)

# of bins	Euclidean Distance (success [%])	EMD (success [%])
5	85.42	72.93
7	87.50	79.17
9	89.60	81.25
11	89.60	77.08
15	85.42	79.17
20	77.08	79.17
30	75.00	85.42
45	75.00	75.00

Table 4.3: Euclidean Distance error function and # of bins in SIFT(see figure 4.6 on page 35) and how this affects the classification rate

4.3.2 SIFT representation

We have 3 SIFT variants and each of them can have overlapping and non-overlapping topology (described earlier). We can observe that the overlapping topology is doing better in almost all cases except the simplest one.

The differencies between SIFT representations were described earlier in Chapter 4.

4.3.3 Euclidean versus EMD error function

We tried to find out the relation between *number of bins* and error function. We observed an interesting behaviour (figure 4.6 on page 35). For Euclidean distance error function we had the best classification rate 90% for 9 bins per box. For EMD the peak was for 30 bins, the classification rate was 85.5%. The lack of discrimination with more bins is being solved via EMD. On the other hand with less bins these are discriminative by itself.

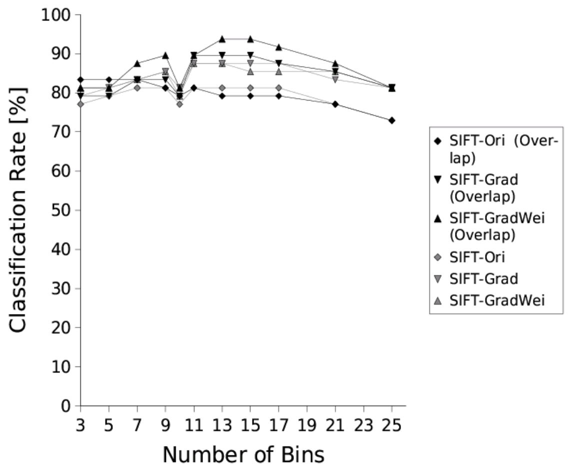

Figure 4.5: SIFT Type and how this influences the classification rate

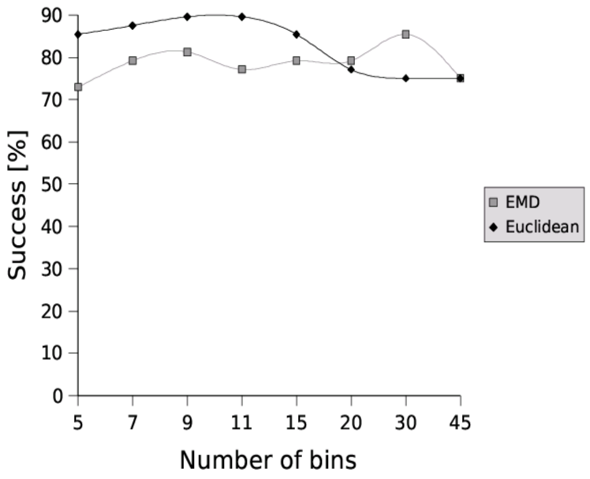

Figure 4.6: EMD vs. Euclidean Distance Function

k	Sift Topology:	6x3	6x5	8x4	15x9
1		54.17	70.83	89.58	93.75
3		66.67	83.33	72.92	93.75
5		60.42	81.25	70.83	81.25
7		72.92	75	68.75	85.42
9		70.83	66.67	68.75	70.83
11		66.67	56.25	62.5	58.33
13		60.42	54.17	64.58	56.25
15		56.25	54.17	60.42	52.08
17		43.75	47.92	52.08	37.5
19		29.17	41.67	43.75	25
21		25	43.75	39.58	20.83
23		18.75	33.33	31.25	20.83
25		20.83	29.17	29.17	22.92
27		20.83	31.25	25	20.83

Table 4.4: Best k for k-NN and how this influence the classification rate(see figure 4.7 on page 37)

4.3.4 Best k for k-NN

Here we can see how the k influence the classification itself. We can see that with increasing k the classification rate is decreasing (figure 4.7 on page 37) and that there exist some k for which the classification is the best. This is not necessarily the minimum one as we can see that for SIFTs with worse discrimination power (number of square patches: 6x3, 6x4) the peak is for the $k > 1$. For SIFTs with number of square patches 8x4 and 15x9 the peak is for $k = 1$ and $k = 3$ respectively.

Let us show what is happening on the picture figure 4.8 on page 37. Let C is a test car and $C_1, \ldots, C_N$ are car classes from the learning set. The classification error for (C, C_1) is e_1, for (C, C_i) is e_i. Leat us assume there is an $E, E > 0$ which denotes a threshold for which the assignment between $(C, C_i) = E$ is pointless. In another words C has nothing to do with C_k. If we choose some k we are trying to find a class which has a majority between $C_1, \ldots, C_k$ observations. The problem is that if from some $j, j < k$ the error $e_j, e_j > E$ exists we are taking into account car class $C_j, \ldots, C_k$ which have noting to do with C.

The graph on figure 4.7 on page 37 is showing the classification is going worse from $k \geq 5$. The problem was if we had e.g. 1x Fiat Uno with 2 total occurences in class and 4x Skoda Felicie with total number of occurences > 8. Therefore the relative majority is for Fiat Uno since it has relative occurence of 0.5 whereas Felicie would have at most $4/9 < 0.5$.

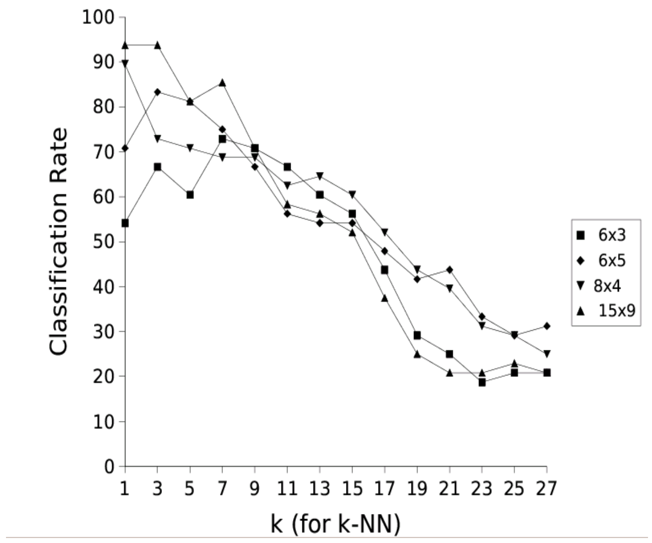

Figure 4.7: The best k for k-NN algorithm

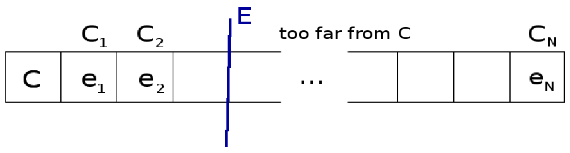

Figure 4.8: The explanation to figure 4.7 on page 37

4.4 k-NN Performed on truck database

4.4.1 SIFT topology

The best classification rate approx. 89% was achieved for SIFT with topology 15x9x25. However SIFT with topology 17x13 showed the similar classification power (85–88%) with number of bins between 15–30 . We saw that every SIFT had its maximum for certain number of bins and then the classification rate was lower for higher number of bins. This peak was for SIFT with higher number of tiles around 25 bins per tile, for SIFTs with lower number of tiles it was round 15 bins per tile (figure 4.9 on page 38). Another interesting point was that the classification power achieved its peak for SIFT 15x9 and for other SIFTs it was not outperformed. The classification power stagnates from certain number of tiles (figure 4.11 on page 39).

Another interesting observation is that the classification rate for number of bins per tile equal 10 is very low, it is lower then for surrounding number of bins: 9 and 11. The possible explanation gives us the picture figure 4.10 on page 39.

SIFT Topology and its relation to the classification rate [%]

# of bins	6x3	6x4	8x4	10x5	13x9	15x9	17x13	19x15	21x17
3	28.26	36.96	44.57	30.43	50	41.3	50	-	-
5	36.96	47.83	52.17	41.3	54.35	55.43	55.43	-	-
7	45.65	56.52	63.04	45.65	60.87	71.74	-	-	-
9	53.26	58.7	64.13	50	67.39	76.09	70.65	-	-
10	46.74	45.65	53.26	51.09	64.13	60.87	67.39	-	-
11	50	54.35	60.87	60.87	75	80.43	78.26	-	-
13	55.43	61.96	65.22	59.78	80.43	81.52	82.61	-	-
15	55.43	64.13	67.39	63.04	77.17	83.7	85.87	82.61	86.96
17	53.26	70.65	66.3	64.13	79.35	84.78	86.96	83.7	85.87
21	53.26	66.3	65.22	67.39	81.52	85.87	88.04	84.78	88.04
25	48.91	66.3	68.48	75	83.7	89.13	88.04	88.04	86.96
30	-	-	-	73	80	81.25	86.96	79.35	77.17

Table 4.5: The SIFT topology and how it influence the classification rate (see figure 4.9 on page 38)

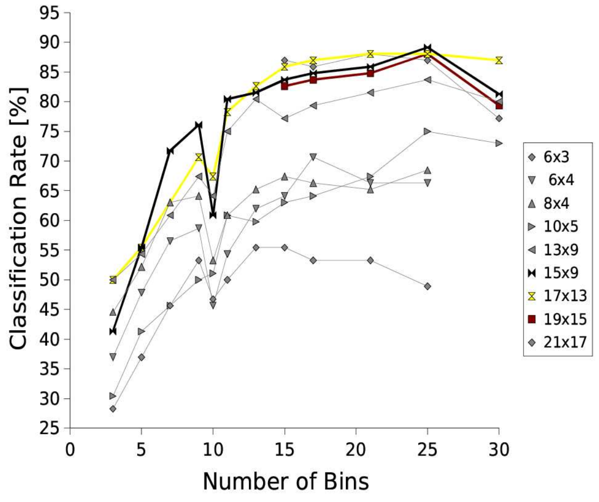

Figure 4.9: We can see that SIFT with topology 15x9x25 performed best

Number of tiles its relation to the classification rate [%]

SIFT Topology	Classification Rate [%]
6x3	55.43
6x4	70.65
8x4	68.48
10x5	75
13x9	83.7
15x9	89.13
17x13	88.04
19x15	88.04
21x17	88.04

Table 4.6: Increasing of SIFT tiles increases the classification rate figure 4.4 on page 33)

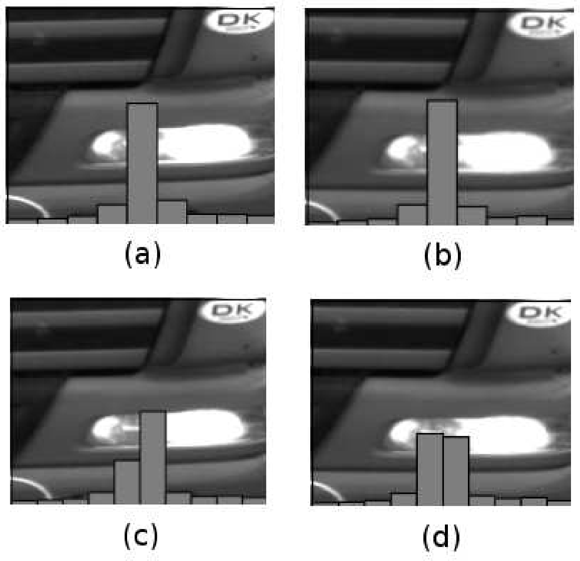

Figure 4.10: Rotation and stability: (a) sample rotated $\approx 1^\circ$ to the left, number of bins equals to 9, (b) sample rotated $\approx 2^\circ$ to the right, number of bins equals to 9, (c) sample rotated $\approx 1^\circ$ to the left, number of bins equals to 10, (d) sample rotated $\approx 2^\circ$ to the right, number of bins equals to 10

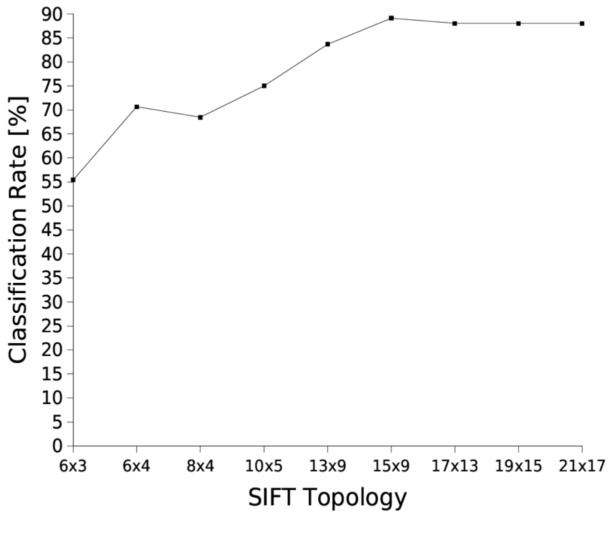

Figure 4.11: SIFTs with higher number of tiles have better classification power but from certain point the classification rate stagnates

SIFT Representation	Classification Rate [%]
SIFT-Ori SIFT	84.78
SIFT-Grad	86.96
SIFT-GradWei	89.13
SIFT-Ori (overlapping)	82.6
SIFT-Grad (overlapping)	88.04
SIFT-GradWei (overlapping)	90.22

Table 4.7: SIFT representation comparison(see figure 4.12 on page 40)

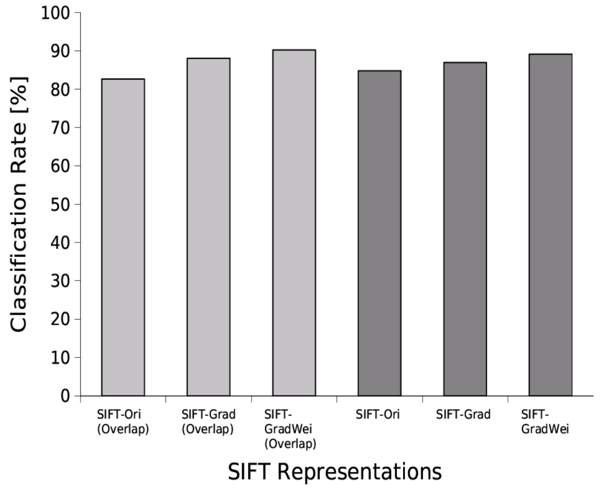

Figure 4.12: Here we can see that overlapping SIFTs performed better except the simplest case

4.4.2 SIFT representation comparison

We can see that the overall quality is almost the same and it is between 80–90%. The best classifier was overlapping SIFT with topology 15x9x25 with the classification rate approximately 90%. However, even the simplest SIFT had the classification rate around 82%.

4.4.3 EMD vs. Euclidean distance

We had to use a smaller subset of truck images since the EMD algorithm is a time consuming computation. We reduced our database to 112 images in total. We observed that EMD and Euclidean distance measures performed almost with the same quality but Euclidean distance outperformed EMD (figure 4.13 on page 41). The best classification rate for EMD was 40.9% and for Euclidean distance measures it was 50%. The highest values were achieved for 25 bins per tile. The classification rate decreased with number of bins.

Distance Measure and Classification Rate [%]

Number of Bins	EMD	Euclidean distance
25	40.90	50
21	34.09	43.18
17	36.36	45.45
15	43.18	47.72
13	45.45	45.45
11	40.90	47.72
10	22.73	36.36

Table 4.8: Distance Measures Comparison(see figure 4.13 on page 41)

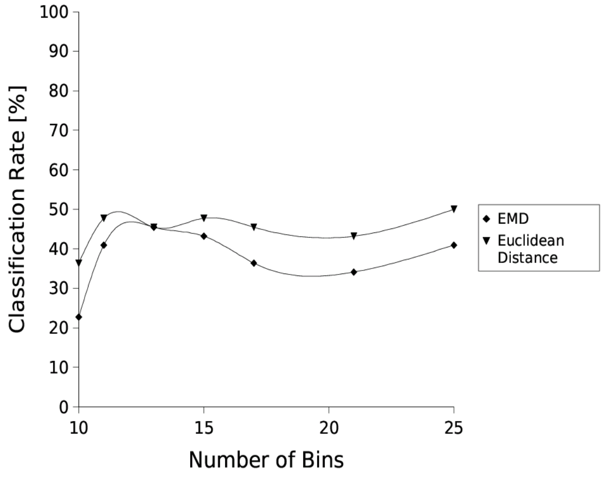

Figure 4.13: We can see that EMD and Euclidean distance measures performed almost the same

k for k-NN	Classification Rate [%]
1	92.39
3	86.95
5	90.21
7	85.86
9	88.04
11	85.86
13	84.78
15	84.78
17	84.78
19	83.69
21	85.86
23	83.69

Table 4.9: Best k for k-NN and how this influence the classification rate(see figure 4.14 on page 42)

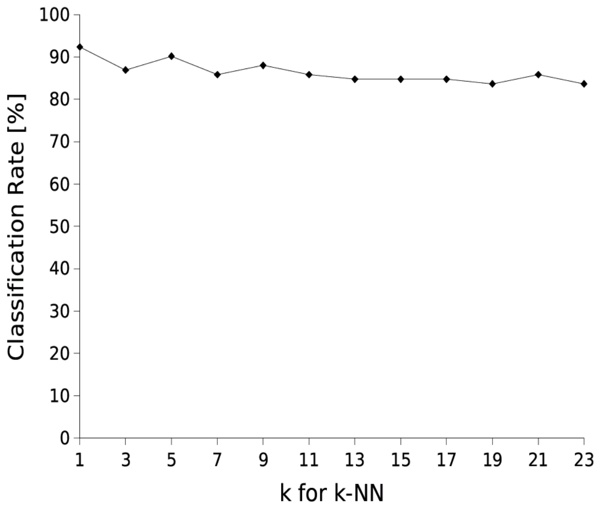

Figure 4.14: The influence of k on classification rate

4.4.4 Best k for k-NN

We tried this experiment for SIFT with topology: 15x9x25. This one was the best so far. Here we can see how the k parameter influences the classification itself. We can see that with increasing k the classification rate is decreasing (figure 4.14 on page 42). The best classification rate was obtained for $k = 1$ however the classification rate remains almost the same for all tried k.

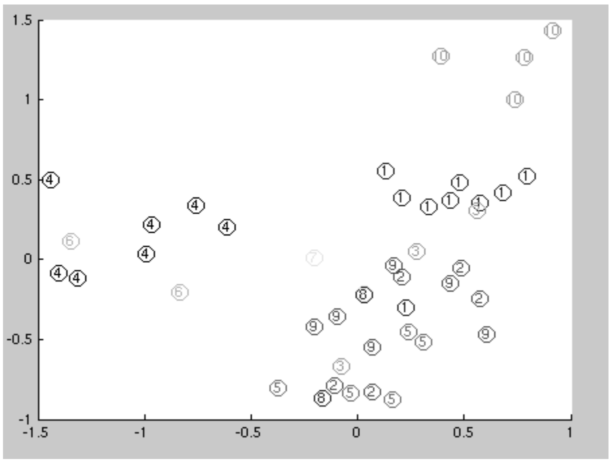

Figure 4.15: The FLD transformation performed on features extracted from car images

Car Class	Car Label in the picture
octavie	1
opel	2
opel2	3
skoda120	4
felicie	5
skoda105	6
favorit2	7
fiat-uno	8
favorit	9
fabie	10

Table 4.10: Classes to numbers assignment(see figure 4.15 on page 43)

4.5 Fisher's linear discriminator (FLD) performed on car images

Experiments for Car Images using FLD were done on SIFT with topology 15x9x15 and overlapping tiles since this was the best combination obtained so far for car images. For truck images the optimal SIFT was 15x9x25 and with overlapping tiles, however the computation was so complex that we never obtained the projection matrix within 12 hours. We decided to use SIFT with topology 15x9x13 without overlap and compare this to the k-NN with the same SIFT topology.

Here we can see the FLD projection from SIFT feature space to training set projection space. We can see in figure 4.15 on page 43 the distribution of classes and in figure 4.16 on page 44 the distribution of truck classes when using FLD.

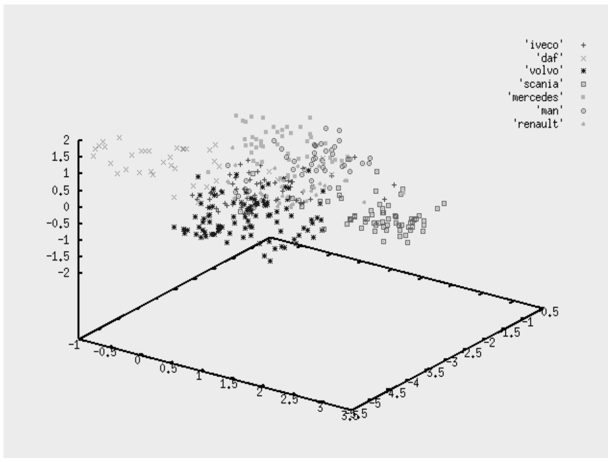

Figure 4.16: The FLD transformation performed on features extracted from car images

	Classification Rate [%]	
Feature Space Dimension	**1-NN**	**nearest μ**
1	41.67	35.41
2	54.17	45.83
3	64.58	50.00
4	60.42	50.00
5	72.92	66.67
6	77.08	75.00
7	77.08	75.00
8	77.17	83.33
9	79.17	83.33

Table 4.11: We applied the FLD projection to various number of dimensions, using 1-NN and nearest class mean as classifiers in projected space.

4.5.1 Optimal dimension

4.6 Fisher's linear discriminator (FLD) performed on truck images

4.6.1 Optimal dimension

In this experiment we take SIFT of certain topology (15x9x13) and variant and do the classification using the FLD and 1-NN in projection space.

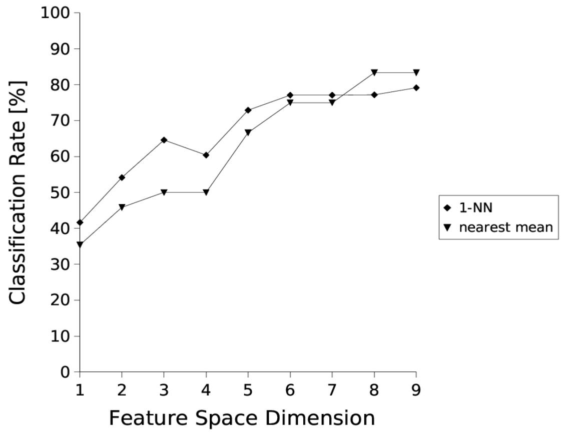

Figure 4.17: The optimal feature space dimension selection for car images

	Classification Rate [%]	
Feature Space Dimension	**1-NN**	**nearest μ**
1	31.52	31.52
2	48.91	47.82
3	66.30	60.87
4	66.30	71.73
5	81.52	80.04
6	81.52	83.69

Table 4.12: We applied the FLD projection to various number of dimensions, using 1-NN as a classifier in that projected space we obtained these results

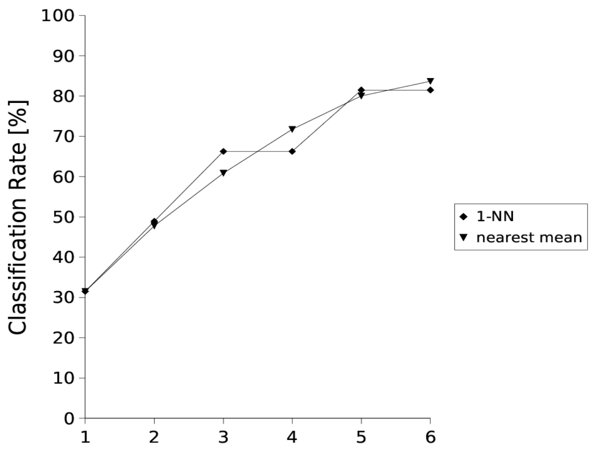

Figure 4.18: The optimal feature space dimension selection for car images

Chapter 5

Experimental validation

In *Chapter 4* we obtained the best recognition system configuration as:

Car Images:

- Number of horizontal square patches: 15
- Number of vertical square patches: 9
- Number of bin per square patch: 15
- Tiles Overlapping: yes
- SIFT representation: SIFT-GradWei
- Classifier: 1-NN
- Error function: Euclidean distance

Truck Images:

- Number of horizontal square patches: 15
- Number of vertical square patches: 9
- Number of bin per square patch: 25
- Tiles Overlapping: yes
- SIFT representation: SIFT-GradWei
- Classifier: 1-NN
- Error function: Euclidean distance

We will evaluate the stability of these classifiers in three different areas:

- sensitivity to learning set reduction
- sensitivity to image blur
- sensitivity to image noise

5.1 Sensitivity to learning set reduction

In this experiment we measure the classifier classification rate change when several images are removed from the learning set. We would assume that a highly stable classifier would not change its classification ability rapidly if only a small subset of images is removed from the learning set.

The experiment is performed as follows:

- *Step 1.* Identify the rank n of the learning set D, $n = |D|$, choose a decreasing constant Δn
- *Step 2.* Do the classification for the learning set D and testing set T. Obtain classification rate in percents.
- *Step 3.* Randomly generate a set S of rank Δn by selecting members from D. Then substract S from D
- *Step 4.* Goto *Step 2* if $|D| > 0$
- *Step 5.* Plot a 2-D graph for every obtained rank of D at any time and corresponding classification rate.

5.2 Sensitivity to image noise

To quantitatively check the robustness of our method to image noise, we added multiplicative noise to every pixel's intensity value:

$$I_{noise} = I_{orig} + \sqrt{v} * rand(0,1) \tag{5.1}$$

We assume I_{orig} between 0 and 1. We were increasing v from 0.01 to 0.3 for trucks and from 0.01 to 0.1 for car images. Truck images have higher resolution, thus the variance v could be bigger.

This experiment is implemented as follows:

- *Step 1.* Identify learning set D and testing set T.
- *Step 2.* Initialize V with chosen variance: $V = \{0.01, 0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.3\}$. Let v is the first member of V;
- *Step 3.* $T_v = \{\}$. For every image I_{orig} in testing set T, add multiplicative noise of variance v to every pixel according to the equation 5.1 and construct the new noisy image I_{noise}. Add this image I_{noise} to newly created testing set T_v.
- *Step 4.* Do the classification with noisyfied testing images T_v and learning set D.
- *Step 5.* Let v is the next member of V.
- *Step 6.* Goto *Step 3.*
- *Step 7.* Plot a 2-D graph for every variance v and corresponding classification rate.

Number of elements in training set [%]	Classes Rate [%]
100	92.39
80	89.13
60	84.78

Table 5.1: Sensitivity to Training Set Redution (see figure 5.2 on page 49)

5.3 Sensitivity to image blur

We also tried the robustness on smoothed images, simulating badly focused cameras or image blur. We convolved the original image with the Gaussian kernels of increasing size:

$$I_{blur}(x,y) = I_{orig}(x,y) * G(x,y,\sigma) \tag{5.2}$$

where $G(x,y,\sigma)$ is defined as:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

We were increasing σ from 0.5 to 3.0 for car images and and from 0.5 to 8.0 for truck images. Experiment is performed as follows:

- *Step 1.* Identify learning set D and testing set T.
- *Step 2.* Initialize Σ with chosen values: $\Sigma = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. Let σ is the first member of Σ;
- *Step 3.* $T_\sigma = \{\}$. For every image I_{orig} in testing set T, do a convolution with Gaussian kernel according to the equation 5.2 and construct the new blurred image I_{blur}. Add this image I_{blur} to newly created testing set T_σ.
- *Step 4.* Do the classification with noisyfied testing images T_σ and learning set D.
- *Step 5.* Let σ is the next member of Σ.
- *Step 6.* Goto *Step 3.*
- *Step 7.* Plot a 2-D graph for every Gaussian kernel size σ and corresponding classification rate.

5.4 k-NN stability on truck images

5.4.1 Sensitivity to learning set reduction

We reduced the training set to 80% and then to 60% and we observed what will happen to the classification rate. We observed that the classification rate was decreasing. But even when the training set was reduced to 60% of its original size the classification rate decreasing only to 84%.

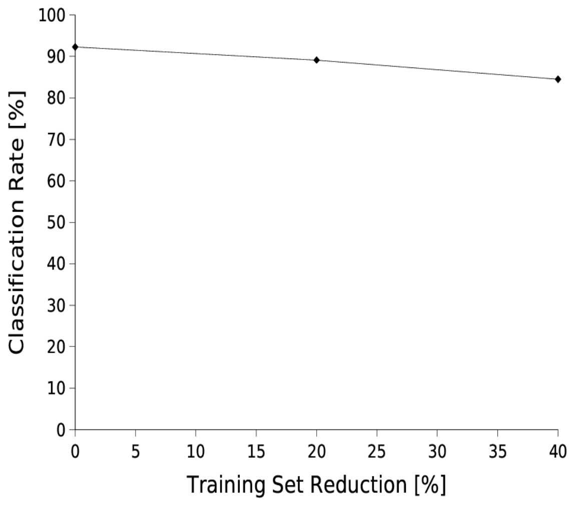

Figure 5.1: The training set was reduced to 80% and 60% of its original size.

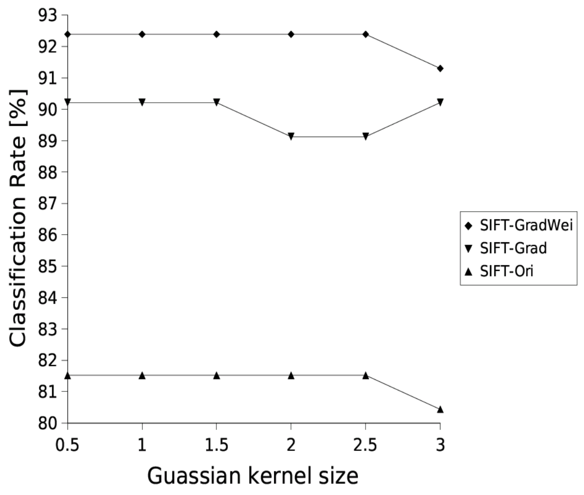

Figure 5.2: k-NN classifier sensitivity to image blur (truck images)

5.4.2 Sensitivity to image blur

It can be observed from the chart (figure 5.2 on page 49) that SIFT was almost invariant to blur. The possible explanation is that SIFT shows almost no change in feature vector in blurred image. (figure 5.3 on page 50). The classification rate was around 92% for the SIFT-GradWei and around 82% for SIFT-Ori.

5.4.3 Sensitivity to image added noise

The SIFT descriptor is sensitive to the image added noise, as we can see in figure 5.6 on page 52. The classification rate was quite stable up to certain v, $v = 0.1$. At this point the clasification

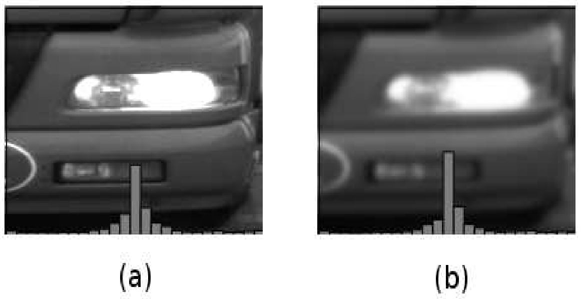

Figure 5.3: SIFT is almost invariant to blur (a) original image, (b) Gaussian with 6x6 region size applied on image

Gaussian kernel size (σ)	SIFT-GradWei	SIFT-Grad	SIFT-Ori
0.5	92.39	90.22	81.52
1	92.39	90.22	81.52
1.5	92.39	90.22	81.52
2	92.39	89.13	81.52
2.5	92.39	89.13	81.52
3	91.3	90.22	80.43
5	90.3	-	-

Table 5.2: Sensitivity to the image blur(see figure 5.2 on page 49)

rate rapidly decreased and simpler SIFTs achieved better classification rates for $v = 0.3$ in comparison to SIFT-GradWei SIFT. The initial classification rate was around 90% for $v = 0.02$, for $v = 0.1$ it was 88%, but for $v = 0.3$ it was only 66%.

noise (v)	SIFT-GradWei	SIFT-Grad	SIFT-Ori
0.02	91.3	89.13	82.61
0.04	92.39	89.13	82.61
0.06	89.13	83.7	80.43
0.08	88.04	83.7	78.26
0.1	88.04	83.7	75
0.2	69.57	70.65	69.57
0.3	66.3	69.57	56.52

Table 5.3: Sensitivity to the image added noise(see figure 5.5 on page 51)

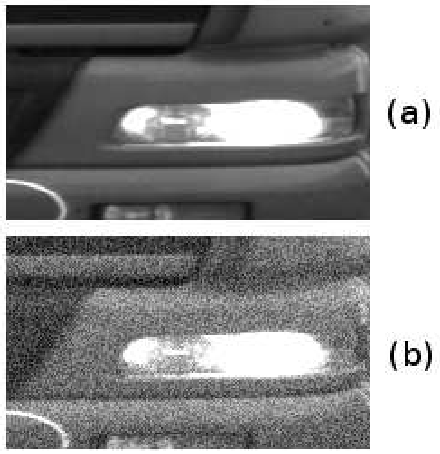

Figure 5.4: Images after noise addition. (a) $v = 0.02$, (b) $v = 0.3$

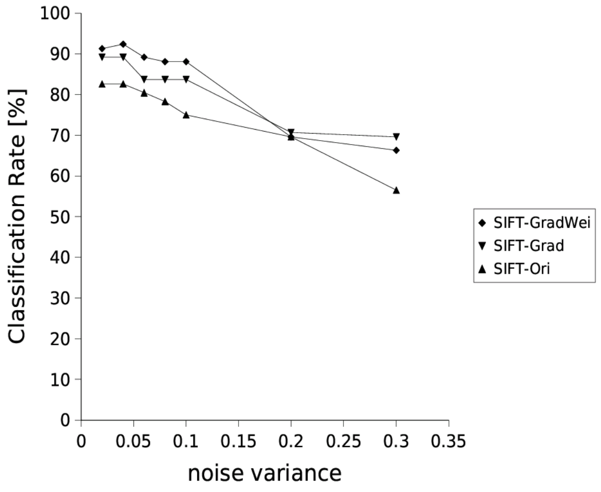

Figure 5.5: k-NN classifier sensitivity to image added noise

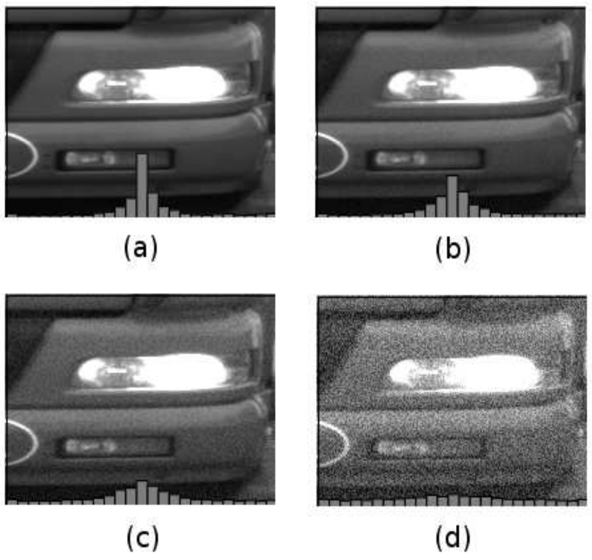

Figure 5.6: In this picture we can see that SIFT discrimination power decreases with increasing amount of noise. For v=0.3 the distribution is almost random (d); (a) original image, (b) added noise, $v = 0.04$, (c) added noise, $v = 0.1$, (d) added noise, $v = 0.3$

training set reduction [%]	Classes Rate [%]
0	93.75
20	93.75
40	79.16

Table 5.4: Sensitivity to Training Set Redution (see figure 5.2 on page 49)

5.5 k-NN stability on car images

5.5.1 Sensitivity to learning set reduction

We reduced the training set to 80% and then to 60% and we observed what will happen to the classification rate.

5.5.2 Sensitivity to image blur

We obtained almost the same result as for truck images (figure 5.8 on page 53). The classification rate is almost not affected by image blur.

5.5.3 Sensitivity to image added noise

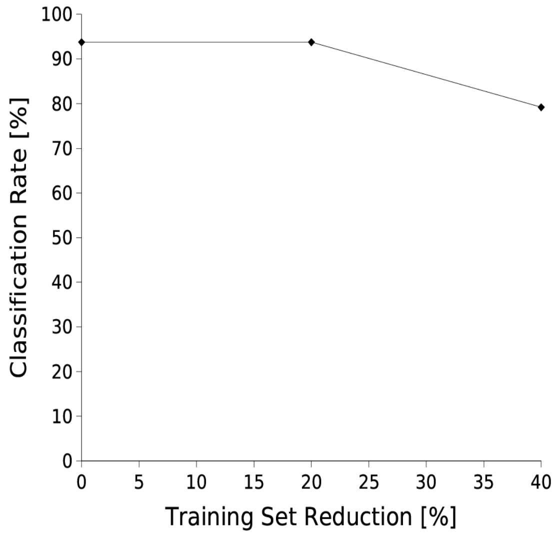

Figure 5.7: The training set was reduced to 80% and 60% from its original size.

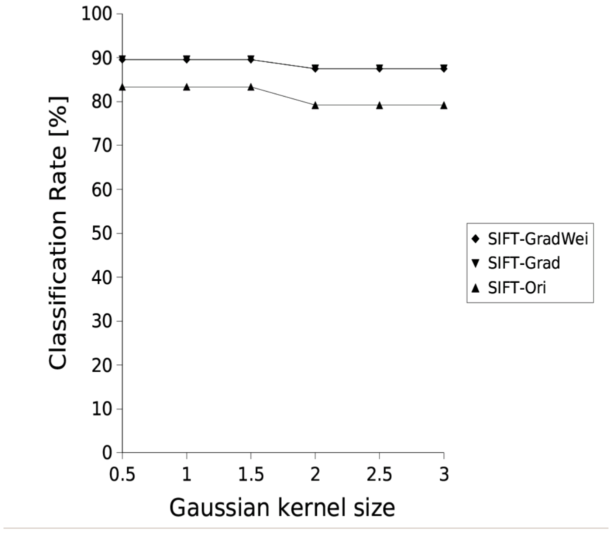

Figure 5.8: k-NN classifier sensitivity to image blur

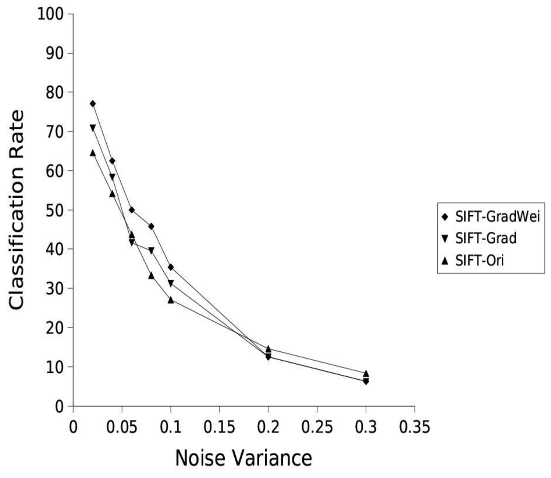

Figure 5.9: k-NN classifier sensitivity to image added noise

We obtained almost the same result as for truck images (figure 5.9 on page 54). The classification rate is heavily affected by image added noise. From variance $v > 0.05$ the classification rate decreases rapidly.

Chapter 6

Conclusion

We tried to classify vehicles into car makes from the frontal view using SIFT descriptor and its variants. We used two databases: one contained car images and the second one contained truck images. We detected the region of interest from an image sample (section 3.1.1, p.16). From this region of interest features were extracted using SIFT descriptor. We tried 6 SIFT representations with various topologies. The SIFT representations were (section 3.2.1, p.22):

- SIFT-Ori
- SIFT-Grad
- SIFT-GradWei

Every SIFT representation had either overlapping tiles or non-overlapping tiles (section 3.2.1, p.23). We tried to classify in the feature space using k-NN and FLD.

The final classifier consists of the SIFT-GradWei (section 3.2.1) and nearest neighbour classifier (section 3.2.3). We succeed with approx. 94% classification rate for car images and approx. 92% classification rate for truck images.

We found that the classification rate is increasing with the increasing number of tiles. From certain number of tiles the classification rate stagnates. (figure 4.4 on page 33 for car images, figure 4.11 on page 39 for truck images). Every SIFT topology had a certain number of bins where the classification rate was the best, we found that SIFT with 15 horizontal square patches, 9 vertical square patches and with 15 bins per square patch and overlapping tiles had the best results for car images database (figure 4.3 on page 32). For truck images the SIFT had 15 square patches in horizontal direction, 9 square patches in vertical direction and 25 bins per square patch and overlapping tiles (figure 4.9 on page 38). Another interesting observation made was that Earth Mover's Distance (section 3.2.2, p. 24) performed similarly to Euclidean distance measurment (section 3.2.2, p. 23). The comparison of SIFT variants can be seen in figure 4.5 on page 35 for car images and in figure 4.12 on page 40 for truck images. We can see that the SIFT-GradWei with overlapping tiles always performed best but their results are comparable to each other. The biggest difference is approx. 8% for truck images and 12% for car images. Then we estimated best k for k-NN algorithm. We found that for truck images the classification rate decreases with the increasing k but only a little bit (figure 4.14 on page 42). For car images the classification rate decreases more rapidly since we had smaller amount of images for class and from certain k it does not make sense to classify using that k (figure 4.7 on page 37). With FLD we tried to identify the optimal dimension for the projection of feature space. For both cases it was the highest possible: 9 for cars and 6 for trucks. We used two classifications in the FLD transformed feature space: 1-NN and nearest μ. We found that nearest μ performed better but it was still worth than k-NN.

The sensitivity in 3 areas was evaluated:

- sensitivity to image blur
- sensitivity to image added noise
- sensitivity to learning set reduction

We observed that SIFT is almost invariant to image blur figure 5.3 on page 50. The results that the classification rate is not affected can be seen in figure 5.2 on page 49 for truck images and in figure 5.8 on page 53 for car images. The SIFT is not stable for noise addition with higher values, the orientation distribution becomes random, figure 5.4 on page 51, the classification rate falls down from variance higher 0.1 for truck images (figure 5.5 on page 51) and from variance higher then 0.05 for car images (figure 5.9 on page 54). The reason for this difference is that truck images have higher resolution (1280x512) in comparison to car images (256x92). The sensitivity of final classifier to training set reduction was good, for the training set reduction to 60% of its original size we got approx. 14% for car images and approx. 7% for truck images.

6.1 Discussion and further research

The recognition system was implemented in Perl, C, Matlab and shell-scripts. The first implementation of SIFT was done in Perl and using Image::Magick library for the basic image manipulaiton such as loading, saving and color channels operations. The SIFT computation took approx. 10 seconds for a geometrically normalized image (256x92 pixels). By rewritting the SIFT module into C with support of libpng for basic image operations we gain a 70–150x speedup.

Other improvements to the classification rate can be achieved by using a different classification methods such as Support Vector Machines or AdaBoost. The proposed method would need to be evaluated with more vehicle sample images.

Appendix A

Software design

The software was designed with respect to maximum modularity and pluggability. This was needed because we had to verify various configurations of feature extraction algorithms, error functions and classifiers.

A.1 Architecture overview

The high-level software architecture is depicted on figure A.1 on page 57.

The basic idea is that the program will get an input image (test car). The program has a database of car images which are used for *training*. The car make is known for these vehicles to the program. We expect the program to display a car make for the test image.

During the classification process there is an *feature vector cache* created from every train car. This helps to speed-up the overall classification process.

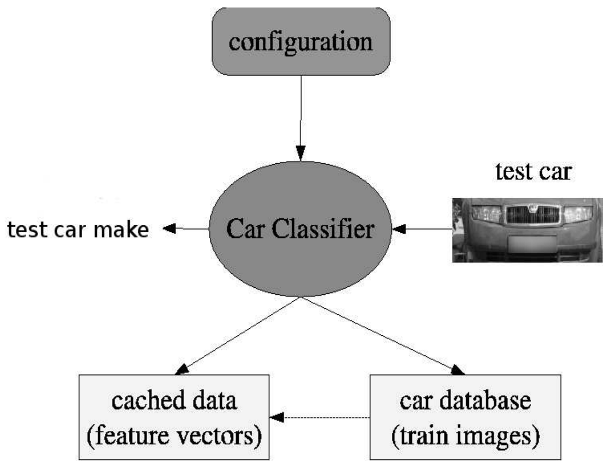

Figure A.1: The Archiecture Overview

A.2 Program flow

On figure A.2 on page 59 we can see in more detail how the car recognition process is implemented. It consists of three basic modules:

- feature extraction module
- error function module
- classification module

We will focus on these modules later.

In the beginning the input test car image is loaded. Features are extracted by *Feature Extraction Module*. If the car image does not have cached feature data the cache is being created. Later this cached is preferred since it is many times faster than to compute the features again. The next step is to compare error in *Error Function Module* between every train car and test car. Once error distances are computed the classification performed by *Classification module* can begin. It can consists from selecting the closest car or by dimension reduction, etc. The classification can also utilize the stpdf above. The last step is to display the closest image from the train database.

A.3 Detailed modules design

A.3.1 Feature extraction module

We can see feature extraction module on figure A.3 on page 59. The abstract module is the interface definition. The following must be implemented by ancestors:

- $\mu_{\mathbf{I}} = \text{extract\_features}(I)$
- $\mu_{\mathbf{n}} = \text{normalize}(\mu_{\mathbf{I}})$

where I is an input image, $\mu_{\mathbf{I}}$ is non-normalized feature vector and $\mu_{\mathbf{n}}$ is its normalized version. The normalize() function is useful since it makes a distribution from feature vector.

A.3.2 Error module

We can see error module architecture on figure A.4 on page 60. The interface makes ancestors to implement the following:

- $d = \text{compute\_distance}(\mu_{\mathbf{1}}, \mu_{\mathbf{2}})$

where $\mu_{\mathbf{1}}, \mu_{\mathbf{2}}$ are feature vectors, d is a real number. This module implements equation (3.1).

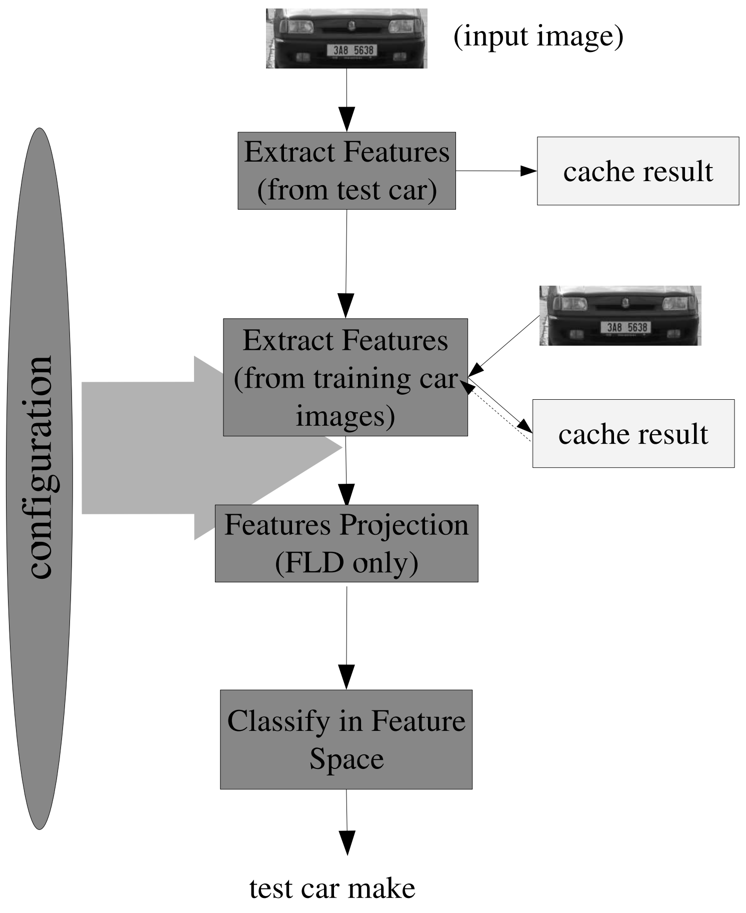
(input image)
Extract Features (from test car)
cache result
Extract Features (from training car images)
cache result
Features Projection (FLD only)
Classify in Feature Space
test car make
configuration

Figure A.2: Program Flow

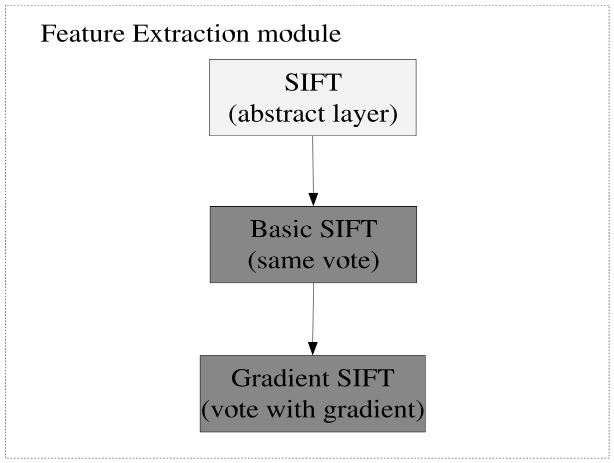

Figure A.3: Feature Extraction Module

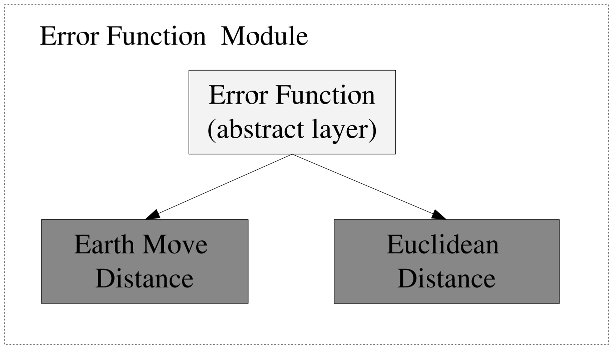

Figure A.4: Error Module

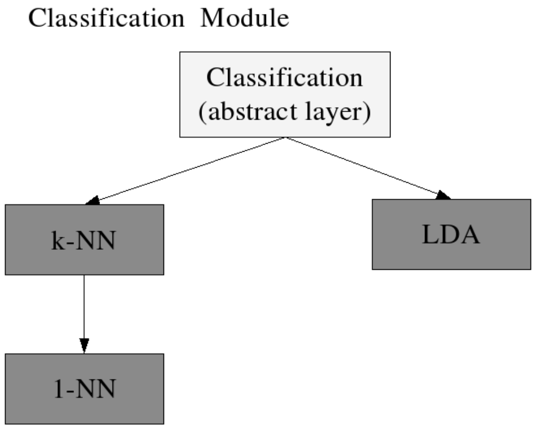

Figure A.5: Classification module

A.3.3 Classification module

Classification module architecture is depicted on figure A.5 on page 60. The interface defines:

- I_k = classify(I_{test}, I_{train_1}, I_{train_2}, $\ldots$)
- I_k = classify($\mu_{\mathbf{test}}$, $\mu_{\mathbf{train_1}}$, $\mu_{\mathbf{train_2}}$, $\ldots$)
- I_k = classify($[e_{ij}]$)
- $\{(I_1, p_1), (I_2, p_2), \ldots\}$ = classify(I_{test}, I_{train_1}, I_{train_2}, $\ldots$)
- $\{(I_1, p_1), (I_2, p_2), \ldots\}$ = classify($\mu_{\mathbf{test}}$, $\mu_{\mathbf{train_1}}$, $\mu_{\mathbf{train_2}}$, $\ldots$)
- $\{(I_1, p_1), (I_2, p_2), \ldots\}$ = classify($[e_{ij}]$)

where I_k is the image from training set, p_k is the classification confidence, I_{test} is the image to classify $\mu_{\mathbf{test}}$ is the test car feature vector and $[e_{ij}]$ is the error matrix. The latter case of classification methods reflects the fact that some of classification methods are able to return more than one solution and tell us the classification probability.

Bibliography

[1] Š. Obdržálek, J. Matas. Object Recognition using Local Affine Frames on Distinguished Regions. In P. L. Rosin and D. Marshall, editors, Proceedings British Machine Conference, volume 1, pages 113–122.

[2] J. Matas, T. Pajdla, O. Chum, M. Urbam. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In British Machine Vision Conference, Cardiff, Wales, pages 384–393

[3] Š. Obdržálek, J. Matas. Image Retrieval Using Local Compact DCT-based Representation. In DAGM'03, 25th Pattern Recognition Symposium, 2003

[4] J. Sivic, A. Zisserman. Video Google. A Text Retrieval Approach to Object Matching in Videos. In Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003

[5] J. Lu, K.N. Plataniotis, A.N. Venetsanopulous. Face Recognition Using LDA Based Algorithms. In IEEE Transactions on Neural Networks, 2002

[6] H. Yum J. Yang. A Direct LDA Algorithm for High-Dimensional Data — with Application to Face Recognition. In Pattern Recognition Letters, 2000

[7] Y. Li, J. Kittler, J. Matas. Effective Implementation of Linear Discriminant Analysis for Face Recognition and Verification

[8] Y. Rubner, C. Tomasi, L.J. Guibas. A Metric for Distributions with Application to Image Databases. In Proceedings of the IEEE International Conference on Computer Vision, 1998

[9] S. Belogie, J. Malik. Matching with Shape Contexts. In IEEE Workshop on Content-Based Access of Image and Video Libraries, 2000

[10] T. Tuytelaars. Local Invariant Features for Registration and Recognition. PhD Dissertaion Thesis, 2000

[11] S. Belogie, J. Malik, J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. In IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 24, 2002

[12] A. Thayananthan, B. Stenger, P.H.S. Torr, R. Cipolla. Shape Context and Chamfer Matching in Cluttered Scenes

[13] A.J. Fitch, A. Kadyrov, W.J. Christmas, J. Kittler. Orientation Correlation.

[14] K. Mikolajczyk, C. Schmid. A performance evaluation of local descriptors

[15] F. Schaffalitzky, A. Zisserman. Viewpoint Invariant Texture Matching and Wide Baseline Stereo

[16] P. Prithcett, A. Zisserman. Wide Baseline Stereo Matching.

[17] T. Kato, Y. Ninonmiya, I. Masaki. Preceding Vehicle Recognition Based on Learning From Sample Images. In IEEE Intelligent Transportation Systems Conference, pages 201–206. IEEE, 2001

[18] Thiang, A.T. Guntoro, R. Lim. Type of Vehicle Recognition Using Gabor Filter Representation and Template Matching Method. In Seminar of Intelligent Technology and Its Applications, 2001

[19] V.S. Petrović, T.F. Cootes. Analysis of Features for Rigid Structure Vehicle Type Recognition. In British Machine Vision Conference, 2004

[20] Z. Zhu, H. Lu, J. Hu, K. Uchimura. Car Detection Based on Multi-Cues Integration.

[21] V.S. Petrović, T.F. Cootes. Vehicle Type Recognition with Matching Refinement

[22] Y. Abramson, Y. Freund. Active learning for visual object recognition

[23] Y. Freund. An introduction to boosting based classification

[24] Y. Freund, R.E. Shapire. A Short Introduction to Boosting. In Journal of Japanese Society for Artificial Intelligence, 14(5), pages 771–780, 1999

[25] D.G. Lowe. Object Recognition from Local Scale Invariant Features

[26] C. Harris, M. Stephens. A combined corner and edge detector. In 4th ALVEY vision conference, pages 147–151,1988

[27] Y. Abarmson, Y. Freund. Active learning for visual object recognition

[28] Ming-Hsuan Yang, David Kriegman, Nareda Ahuja. Face Detection Using Multimodal Density Models. In Computer Vision and Image Understanding 84, pages 264–284, 2001

[29] Juwei Lu, K.N. Plataniotis, A.N. Venetsanopoulos. Face Recognition Using Feature Optimization and μ-support Vector Learning

[30] R. O. Duda, P.E. Hart, D.G. Stork. Pattern Classification. Wiley, Interscience, ISBN: 0-471-05669-3

[31] R. I. Hartley, A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521623049

[32] Ryo Ohta, Shirmila Mohottala, Shintaro Ono,Masataka Kagesawa, Katsushi Ikeuchi. Vehicle Class recognition of street-parking vehicles from side-view range images

[33] Shivani Agarwal, Aatif Awan, Dan Roth. Learning to Detect Object in Images via a Sparse, Part-Based Representation. In IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 26, NO. 11, 2004

[34] Shivani Agarwal, Dan Roth. Learning a Sparse Representation for Object Recognition. In Proceedings of the Seventh European Conference on Computer Vision, 2002

[35] Dongjin Han, Matthew J. Leotta, David B. Cooper, Joseph L. Mundly. Vehicle Class Recognition from Video based on 3D Curve Probes

[36] Shirmila Mohottala, Masataka Kagesawa, Katsushi Ikeuchi. Vehicle class recognition using 3D CG models

[37] Ryuei Nishii. Supervised Image Classification by Contextual AdaBoost Based on Posteriors in Neighborhoods. In IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 43, NO. 11, 2005

[38] Paul Viola, Michael Jones. Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade

[39] Olivier Bousquet, André Elisseeff. Stability and Generalization. In Journal of Machine Learning Research 2, pages 499–526, 2002

[40] W. Freeman, E. Adelson. The design and use of steerable filters

[41] J. Keonderink, A. van Doorn. Representation of local geometry in the visual system

[42] F. Schaffalitzky, A. Zisserman. Multi-view matching for unordered image sets

[43] L. van Gool, T. Moons, D. Ungurenau. Affine l photometric invariants for planar intensity patterns

[44] MP. Dubuisson-Jolly, S. Lakshaman, A. Jain. Vehicle Segmentation and Classification Using Deformable Templates. In IEEE TRANSACTIONS PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 18(3):293–308,1996.

[45] perl.
http://www.perl.com/

[46] stprtool.
Statistical Pattern Recognition Toolbox for Matlab.
http://cmp.felk.cvut.cz/ xfrancv/stprtool/

[47] Image Magick.
http://www.imagemagick.org/

[48] Earth Mover Distance (code).
http://ai.stanford.edu/ rubner/emd/default.htm

[49] Octave.
A high-level interactive language for numerical computations.
http://www.octave.org/

[50] gcc.
GNU Project C and C++ compiler.
http://gcc.gnu.org/

[51] libpng.
Portable Network Graphic (PNG) Library.
http://www.libpng.org/

[52] YAML.
YAML Ain't Markup Language.
http://www.yaml.org