AALBORG UNIVERSITY

UC San Diego

Traffic sign detection using computer vision

Explorations for a driver support system

Andreas Møgelmose
Autumn, 2011 - spring, 2012

Vision, Graphips, and Interactive Systems, AAU
Computer Vision and Robotics Research Lab, UCSD

AALBORG UNIVERSITY

Department of Architecture, Design & Media Technology

Fredrik Bajers Vej 7, 9220 Aalborg, Denmark

Telephone 99 40 86 00

http://es.aau.dk

Title: Traffic sign detection using computer vision

Explorations for a driver support system

Theme: Master thesis: Vision, Graphics, and Interactive Systems.

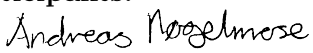
Project period:

P9 and P10, fall 2011 and spring 2012

Project group:

VGIS 1023

Participants:



Andreas Møgelmose

Supervisors:

Thomas B. Moeslund

Mohan M. Trivedi

Issue count: 6

Page count: 121

Enclosed: 4 appendices, 1 CD

Finish date: May 18, 2012

Abstract:

This report is a master thesis in Vision, Graphics, and Interactive Systems. It details the work done during two semesters abroad at UC San Diego. The work has been research oriented, so the report is structured with 5 separate chapters instead of a linear product development flow.

The work has primarily been on US traffic sign detection, but includes a chapter on pedestrian detection as well. A comprehensive survey of traffic sign detection systems has been made and it shows a lack of work with US signs and a lack of public databases for those. Thus, a publicly available dataset with nearly 8000 annotated signs has been created. The dataset is unique, not only because it contains US signs, but also because it include videos. This report also details investigations of using synthetic training data for traffic sign detectors, but concludes that synthetic images are no match for real-world training images. A purely model based detection system based solely on shapes is also presented as a building block for a full detection system. Finally, a two-stage pedestrian detection system has been developed and documented. The system extends a prevous system and produces better detection with fewer false positives.

The work has resulted in the submission of four papers, one to ITS Transactions, one to ICPR and two to ITSC. A the time of writing the journal paper is in the second review stage.

Preface

This report is the documentation of the project "Sign detection using computer vision - detecting US speed limit signs" which was carried out from November 2011 to May 2012.

The report constitutes a master thesis in computer engineering in the area of Vision, Graphics and Interactive Systems from the Department of Electrical Engineering at Aalborg University in Denmark. The bulk of the work for the project was carried out abroad at the Computer Vision and Robotics Research (CVRR) Laboratory and Laboratory for Intelligent and Safe Automobiles (LISA), both at University of California, San Diego (UCSD).

Enclosed with the report are the four papers in the shape they had at the deadline for this report, and a website has been created at *http://moegelmose.com/p10*, where the code and other materials can be downloaded.

Appendix G contains an acronym list.

An example of code formatting can be seen in listing 1. Long lines are wrapped and marked with a ¶ symbol at the end of the line and the rest of the wrapped line is indented.

Listing 1: A example listing of (non-working) code

```
1 void MainWindow::on_generateBtn_clicked()
2 {
3    if(baseImage.channels() == 4) {
4       floodFill(baseImage, mask, Point(baseImage.cols-1,0), Scalar(255)¶
            , 0, Scalar(), Scalar(), 4 | FLOODFILL_MASK_ONLY);
5       mask = mask(Range(1,mask.rows-1), Range(1,mask.cols-1));
6       threshold(mask, mask, 0, 255, THRESH_BINARY_INV);
7    }
8 }
```

This report was written in LaTeX, most non-screenshot figures were created with Inkscape, and plots were made with either Python plus matplotlib, Matlab, or GNUPlot.

Any questions about this work can be sent to andreasm@es.aau.dk.

Andreas Møgelmose, May 2012

Acknowledgments

First of all, I wish to thank professor Mohan Trivedi, head of CVRR and LISA at UC San Diego, for hosting me in his lab, and for his always enthusiastic supervision and great faith in my abilities. I also want to thank my supervisor at AAU, Thomas Moeslund for setting the visit up and encouraging me to study abroad from the very first time I approached him about it. And for patiently and quickly answering all my mails.

I also wish to thank my colleagues in the CVRR Lab at UCSD: Antonio Prioletti for cooperating with me on our work on pedestrian detection. Sayanan Sivaraman for helping me get settled and answering my never ending cascade of stupid questions. Sujitha Martin for discussions about writing, work, and my lack of a sensible work/life balance. Minh Van Ly for showing me his awesome e-bikes. Cuong Tran and Ashish Tawari for always helping out when I had a question.

A thank goes out to all the visitors I have had from home while being over here: My mom, dad, Katrine, Lars, Martin, Joachim, and Johan.

Finally, I would like to thank my very good friend Anders Tornvig for endless chats and discussions about projects, programming, writing, universities, travel, math, LaTeX, and everything else. He is a decent human being, despite studying acoustics and liking Matlab.

Contents

1 Introduction

During my stay at UCSD I have collaborated closely with the research staff in the lab, and as such this project has not been about developing a product as projects at AAU normally are. Instead, I have done research in the area of traffic sign detection and pedestrian detection. My work has resulted in four papers which have been submitted to a journal and two conferences, respectively (all papers are enclosed in appendix A):

- *Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey* submitted to IEEE Intelligent Transportation Systems Transactions, Special Issue on Machine Learning for Traffic Sign Recognition.
- *Traffic Sign Detection and Analysis: Recent Studies and Emerging Trends* submitted to IEEE Intelligent Transportation Systems Conference (ITSC), 2012.
- *Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets* submitted to IAPR 21st International Conference on Pattern Recognition (ICPR), 2012
- *A Two-stage Part-Based Pedestrian Detection System Using Monocular Vision* submitted to IEEE Intelligent Transportation Systems Conference (ITSC), 2012.

As a consequence, this report does not chronicle the development process of a solution to a specific problem. Instead, it consists of 5 relatively self-contained chapters, each about one of the papers, or about significant work I have done which did not directly result in a paper. The chapters which take a submitted paper as their starting point will contain some text taken directly from the papers, but also additional information that did not make it into the paper.

All work presented in this report has been carried out by myself, with input from my two supervisors, Thomas Moeslund and Mohan Trivedi. A sole exception is chapter 6, which has been a cooperation with Antonio Prioletti. All text - including the submitted paper - has been written by me and I have been partly responsible for the overall design and layout of the system. The majority of the implementation and test has been carried out by Antonio Prioletti.

The main theme for my research has been traffic sign detection, and more specifically, detection of US traffic signs. Most of the chapters in the report are concerned with this area. Each of the chapters are summarized in the following sections:

1.1 Survey of the state of the art in traffic sign detection

Chapter 2 is a survey of the state-of-the art in traffic sign detection. It presents the dominant detection strategies and discusses their merits. It also gives an overview of the available public image databases and presents the open issues in the field.

This chapter provides an overview of the state of sign detection. Instead of treating the entire Traffic Sign Recognition (TSR) flow, focus has been solely on the detection of signs. During recent years, a large effort has gone into TSR, mainly from Europe, Japan, and Australia and the developments are described.

The detection process is split into segmentation, feature extraction, and detection. Many segmentation approaches exist, mostly based on evaluating colors in various color spaces. For features there are also a wealth of options. The choice is made in conjunction with the choice of detection method. By far the most popular features are edges and gradients, but other options such as HOG and Haar wavelets have been investigated. The detection stage is dominated by the Hough transform and its derivatives, but for HOG and Haar wavelet features, SVMs, neural networks, and cascaded classifiers have also been used.

Arguably, the biggest current issue with sign detection is the lack of use of public image databases to train and test systems. Currently, every new approach presented uses a new dataset for testing, making comparisons between papers hard. This gives the TSR effort a somewhat scattered look. Recently, a few databases have been made available, but they are still not widely used, and cover only Vienna Convention compliant signs.

This issue leads to the main unanswered question in sign detection: Is a model based shape detector superior to a learned approach, or vice versa? Systems using both approaches exist, but are hard to compare, since they all use different data sets.

Many contributions cite driver assistance systems as their main motivation for creating the system, but so far only little effort has gone into the area of combining TSR systems with other aspects of driver assistance and notably, none of the studies include knowledge about the driver's behavior in order to tailor the performance of the TSR system to the driver.

Other open issues include lack of research in finding non-European style signs and that detected signs are hard to relate to their surroundings.

1.2 LISA Traffic Sign Dataset

One result of the survey was the lack of public databases for US traffic signs. Because of this, it was decided to create one. The creation and structure of this database is the topic of chapter 3.

This chapter describes the assembly of the LISA Traffic Sign Dataset, which has been collected during drives through urban environments in California. It is a dataset of 7855 annotated signs in 47 classes collected over several hours of driving. A set of tools for creating the annotations have been created with great focus on traceability so each annotation can be traced back to its source video.

The dataset fills a gap, since it is the first publicly available dataset with US traffic signs. It also includes full video tracks, which can enable the development and test of detection systems using tracking.

1.3 Synthetic training data

In chapter 4, synthetic training data is covered. Creating a sign database is tedious and time-consuming work, and given that traffic signs are all based on well defined templates, it was natural to research whether synthetically generated training data could be used for sign detection instead of real-world data. Chapter 4 describes the result of this research.

This chapter describes experiments to evaluate the performance of synthetic training data for traffic sign detection compared to real-world data. A program was developed to generate synthetic training data that should emulate pictures of real-world signs based on a drawn template. Then several detectors were trained with both synthetic and real-world data. An AdaBoost cascade with Haar-wavelet like features was used as the detection framework.

Results show that the synthetic data produced here is not of a sufficient quality to rival real-world training data, so unfortunately synthetically generated data is not able to replace painstakingly collected real world data.

1.4 Model based sign detection

Not all sign detection approaches involve training data, however. Chapter 5 describes an alternative way, where the signs are detected using a theoretical model of their shape.

This chapter describes a purely model based detector, which relies on the shape of traffic signs. The method used is Extended Fast Radial Symmetry (EFRS), which votes for shapes based on the gradients in input images. The method has been tested on a

subset of the LISA dataset and shows decent per-sign detection results, with many false positives, however. The tracking system used to reduce the number of false positives is too simple and might even hurt the overall detection performance. The detector is not useful in itself, but will work well as a component of a larger system. This is consistent with the way it is used in the existing literature.

1.5 Pedestrian detection

Chapter 6 is about a pedestrian detection system that was developed in cooperation with Antonio Prioletti of University of Parma, another visiting scholar at the CVRR lab. This system was also created as a part of the lab's ongoing efforts to created intelligent vehicles.

In this chapter, a part-based two-stage pedestrian detector is presented. It builds on previous work by Geismann and Schneider (2008), but extends it by introducing a part-based verification system instead of just a full body verification. The system works in two stages: A detection stage and a verification stage. The detection is based on an AdaBoost cascade on Haar-like features. Its purpose is to find all pedestrian candidate patches in the input image. All Regions Of Interest are sent on to the verification stage, where the Histogram Of Oriented Gradients (HOG) is computed for the entire person, the lower body, and the upper body. Each of the HOGs are then sent through an SVM that computes a confidence value for which class (pedestrian or non-pedestrian) the part belongs to. These values are then passed into a second SVM-classifier, which performs the final verification. The system has been tested on the INRIA dataset and the results show that when compared with the original two-stage detector, it performs better across the full range of false positives per frame.

2

Survey of the state of the art in traffic sign detection

This chapter is largely based on a paper written in conjunction with this project titled *Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey* was submitted to IEEE Intelligent Transportation Systems Transactions, Special Issue on Machine Learning for Traffic Sign Recognition, and is currently passing through its second review stage. Provided it is accepted, it will be published in December 2012. This chapter also includes a section on how the literature search was conducted which is not included the the paper.

A shorter paper on traffic sign detection, but with a greater emphasis on traffic signs classification as well, was submitted to International Conference on Pattern Recognition (ICPR) 2012 under the name *Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets*. Both papers are attached to this report and some of the text in this chapter is taken directly from them.

2.1 Introduction

The area of Traffic Sign Recognition (TSR) systems has been met with growing research interest in the past decade, but the task of recognizing American signs is fairly unexplored so far. TSR is a task with various well defined applications, summarized nicely by De la Escalera et al. (2003):

1. Highway maintenance: Check the presence and condition of signs along major roads.
2. Sign inventory: Similarly to the above task, creating an inventory of signs in city environments.
3. Driver support systems: Assist the driver by informing of current restrictions, limits, and warnings.
4. Intelligent autonomous vehicles: Any autonomous car that is to drive on public

roads must have a means of obtaining the current traffic regulations. This can be done through TSR.

CVRR Lab has a long history of developing active and passive driver support systems, and it is within this realm that this work has been carried out. As described by Trivedi et al. (2007); Trivedi and Cheng (2007); Tran and Trivedi (2011), it is crucial to not only consider the car's surrounding and external environment when designing an assist system, but also to consider the internal environment and take the driver into account. Fusing other types of information with the sign detector, as described by Morris and Trivedi (2010), can make the overall system even better. When the system is considered a distributed system where the driver is an integral part, it allows for the driver to contribute with what he is good at (e.g. seeing speed limit signs, as it will be seen later), while the TSR part can present information from other signs. In addition other surround sensors can also have an influence on what is presented. It is with these ideas in mind, that this literature survey was carried out.

In recent years, speed limit detection systems have been included in top of the line models from various manufacturers, but a more general sign detection solution and an integration into other vehicle systems has not yet materialized. Current state-of-the-art TSR systems neither utilize information about the driver, nor input from the driver, to enhance performance. Extensive studies in Human-Machine Interactivity are necessary to present the TSR information in a careful way, to inform the driver without causing distraction or confusion.

Initially, it was not the plan to carry out an extensive survey, but it turned out that only two surveys in the field exist: Fu and Huang (2010) present a good introduction, but not very comprehensive. Another survey was presented by Fleyeh and Dougherty (2005), but is a few years old, so any improvements in the field from the past 5 years are not presented. A very good comparison of various segmentation methods is offered by Gomez-Moreno et al. (2010), but given that it only covers segmentation, it is not a comprehensive overview of detection methods. Likewise, Houben (2011) provide a good comparison of Hough transform derivatives.

2.2 Using traffic sign detection in driver assistance

Nearly all of the surveyed papers cite driver assistance as the main motivation for creating the system but despite that, little research is concerned with actually including the driver. In order to work with the driver, TSR research needs to take into account the visual system of the driver. This can include factors such as visual saliency of signs, driver focus of attention, and cognitive load. According to Shinar (2007) (see table 2.1 for a summary of the main results), not all signs are equal in their ability to capture the attention of the driver. For example, a driver may fixate his gaze on a sign, but neither notice the sign, nor remember its informational content. While drivers invariably fixate on speed limit signs and recall their information, they are less likely

Table 2.1: Significant results from Shinar (2007) regarding attention to various sign types.

	Fixated		**Not fixated**	
Target	Recalled	Not recalled	Recalled	Not recalled
Speed limit 80 km sign	100	0	0	0
Game Crossing sign	60	0	7	33
Pedestrian crossing ahead	8	54	0	38
Pedestrian crossing sign	0	21	0	79

to notice game crossing and pedestrian signs. This can endanger pedestrians, as it may not leave enough reaction time to stop.

The implications of use of TSR in human-in-the-loop system are clear; instead of focusing on detecting and recognizing all signs of some class perfectly, which would be the objective for an autonomous car, the task is now to detect and highlight signs that the driver has not seen. This gives way to various models of TSR, which take into account the driver's focus of attention, and interactivity issues. Driver attention tracking is covered by Doshi and Trivedi (2010) and Murphy-Chutorian et al. (2007). Fig. 2.1 presents examples on how TSR can be used for driver assistance. Fig. 2.1a shows how a system should act in an autonomous car. It simply recognizes all signs present. In fig. 2.1b there is a driver in the loop, and while the system may see all the signs, it should avoid presenting them in order to avoid driver confusion. Instead, it simply highlights the sign type that is easy to overlook, like the pedestrian crossing warnings in the research. Fig. 2.1c shows how a driver is distracted by a passing car. This causes him to miss two signs. His car has a TSR system for driver assistance, which informs him of the signs as he returns his attention to the road ahead of him. This could, for example, be done using a heads-up display as suggested by Doshi et al. (2009). This chapter does not go into great detail with these issues, apart from the scenarios mentioned here, but as TSR system increase in detection performance, these issues are going to be increasingly relevant.

2.3 On traffic signs

Traffic signs are markers placed along roads to inform drivers about either road conditions and restrictions or which direction to go. They communicate a wealth of information, but are designed to do so efficiently and at a glance. This also means that they are often designed to stand out from their surroundings, making the detection task fairly well defined. Guide and information signs are not particularly interesting from a driver support system point of view, since GPS receivers perform the task of

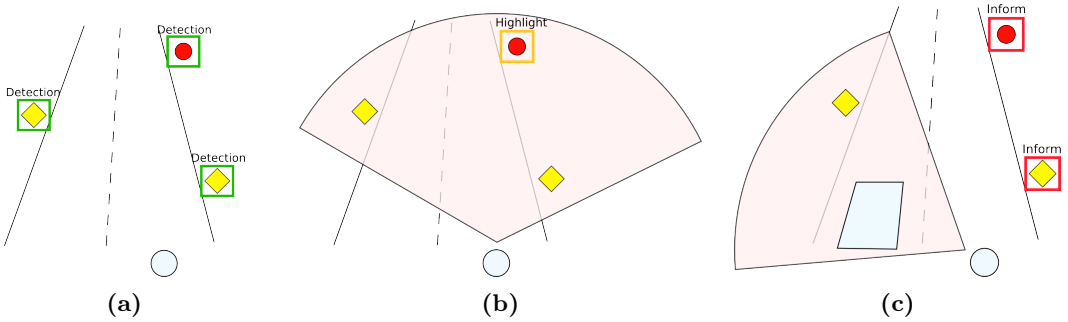


Figure 2.1: Different detection scenarios. The circle is the ego-car and 3 signs are distributed along the road. The area highlighted in red illustrates the driver's area of attention. (a) is the standard scenario used for e.g. autonomous cars. Here, all signs must be detected and processed. (b) and (c) depicts a system which tracks the driver's attention. In (b), the driver is attentive and spots all signs. Therefore the system just highlights the one sign that is known to be difficult for people to notice. In (c), the driver is distracted by a passing car and thus misses two signs. In this case, the system should inform the driver about the two missed signs.

giving directions much better than any sign based system ever could. Therefore this work is only concerned with the other type.

The design of traffic signs are standardized through laws and in Europe many signs are standardized throughout the EU via the Vienna Convention on Road Signs And Signals (United Nations Economic Commission for Europe, 2006). There, shapes are used to categorize different types of signs: Circular signs are prohibitions including speed limits, triangular signs are warnings and rectangular signs are used for recommendations or sub-signs in conjunction with one of the standard shapes. In addition to these, octagonal signs are used to signal a full stop, downwards pointing triangles yield and countries have different other types, e.g. to inform about city limits.

This work was done in the US and is thus concerned with US signs. This is a topic that is not very well covered in existing literature. In the US, traffic signs are regulated by the Manual on Uniform Traffic Control Devices (MUTCD). It defines which signs exist and how they should be used. It is accompanied by the Standard Highway Signs and Markings (SHSM) book, which describes the exact designs and measurements of signs. Curiously, the SHSM is provided in both a metric and an English version, and the measurements are not the same in the two versions, since a "soft conversion" was used to convert from English measurements to metric measurements. The difference is fairly small, though so it is not of any importance for this application. At the time of writing, the most recent MUTCD was from 2009, while the SHSM book has not been updated since 2004, and thus it describes the MUTCD from 2003. An updated version of the SHSM should be on its way. The MUTCD contains a few hundred different signs, divided into 13 categories.

Each state can decide whether it wishes to follow the MUTCD. A state has tree

(a) Stop. Sign R1-1.

(b) Yield. Sign R1-2.

(c) Speed limit. Sign R2-1.

(d) Turn warning with speed recommendation. Sign W1-2a.

Figure 2.2: Examples of signs from the MUTCD. All signs exist in the national MUTCD, and are unchanged in the California MUTCD.

options:

1. Adopt the MUTCD fully as is.
2. Adopt the MUTCD but add a state supplement.
3. Adopt a State MUTCD that is "in substantial conformance with" the national MUTCD.

With regard to this, it is worth noting that the MUTCD covers more than just traffic signs, so it might very well be other parts of it that the states choose to modify. However, they do have the option to add or change signs. 19 states has adopted the national MUTCD without modifications, 23 has adopted the national MUTCD with a state supplement and 10 has opted to create a State MUTCD (the count includes the District of Columbia and Puerto Rico). California has a State MUTCD.

Also of interest to TSR systems is that according to the SHSM, "standardization of these designs does not preclude further improvements by minor changes in the proportion of the symbols, width of borders, or layout of word messages, but all shapes and colors shall be as indicated." Some signs can also have different sizes, depending on which type of road the are used on.

American signs are divided into a number of categories, and in this work only regulatory and warning signs are treated, since, as stated earlier, guide and information signs are less interesting for a driver support system. Another relevant categorization (which does not exist in the MUTCD, but are defined here for the purposes of this project) are zone-signs versus point-signs. Zone-signs are signs that are in effect until they are lifted explicitly by another sign or other traffic regulation, such as turning onto a different road. Speed limit signs are a good example of that. Point-signs are signs that are only relevant at the point where they are displayed. Examples include stop signs and many warning signs.

(a) Speed limit. Sign R2-1.

(b) Minimum speed. Sign R2-4.

(c) End speed limit. Sign R3 (CA).

(d) Red light violation fine. Sign SR58 (CA).

Figure 2.3: Examples of similar signs from the MUTCD. (c) and (d) exist only in the California MUTCD.

With the legalese out of the way, it is time to look at some signs. Figure 4.1 shows three examples of signs from the national MUTCD, which exist unchanged in the California MUTCD. They have different shapes and colors, and seem easy to distinguish. They are examples of the only shapes used for US signs: Octagons (used exclusively for stop signs), downward points triangles (used exclusively for yield signs), diamonds (used for warning signs), and rectangles of various aspect ratios. In addition to these four shapes, Californian route signs has different shapes, such as various shield designs. As these are guide signs, they are not included here. Many signs contain circles, arrows, and other geometric shapes, but crucially, the signs themselves are never circular.

With regards to colors, a few are used. Again, guide and information signs are excluded here. Regulatory signs may have a black, blue, red, or white background, with white being the most common. Warning signs are always yellow. At road works, accidents or other types of temporary traffic control, versions of the normal signs are used, but with an orange background.

A glance through the California MUTCD reveals two things to consider: Many signs are very similar, and many signs rely on text. Examples can be seen in figure 2.3. These are all white signs using text as the only way to communicate their message. Without the accompanying text, it would be near impossible to tell if e.g. figure 2.3a was a maximum or minimum speed limit. Furthermore, figures 2.3a and 2.3b are so similar, that without understanding the text, their meaning could easily be confused. Figure 2.3d is an example of a sign that, while having the same color, practically the same shape, and a lot of text, has absolutely nothing to do with speed limits like the other three. This is in contrast to Europe, where a combination of color, shape, and symbols conveys the meaning of the vast majority of signs without using text.

The number of signs in the MUTCD shows that while TSR is a well defined task - there is not an infinite number of signs - there is still many different signs, signs with

(a) Keep right. Sign R4-7.

(b) Keep right. Sign R4-7a.

(c) Keep right. Sign R4-7b.

Figure 2.4: Three different signs with the same meaning.

subtle differences having different meanings, signs may have additional plaques below for further clarification, and several different signs may even have the same meaning, as illustrated in figure 2.4. This makes the TSR job non-trivial.

New Zealand uses a sign standard with warning signs that are yellow diamonds, as in the US, but regulatory signs that are round with a red border, like the ones from the Vienna Convention countries. Japan uses signs that are generally in compliance with the Vienna Convention, as are Chinese regulatory signs. Chinese warning signs are triangular with a black/yellow color scheme. Central and South American countries do not participate in any international standard, but often use signs somewhat like the American standard.

2.4 Sign detection versus classification

The task of TSR is often split into two different stages: Detection and classification. Detection is concerned with locating signs in input images, while classification is about determining what type of sign the system is looking at. The two tasks can often be treated completely separate, but in some cases, the classifier relies on the detector to supply information, such as the sign shape or sign size. This also means that if necessary, different classifiers could be run, depending on, say, what sign shape was detected.

In this report, the following terminology is used: Sign *detection* denotes the part where the position of a sign in an input image is determined. Sign *classification* is the task of figuring out exactly what type of sign it is. Sign *recognition* is the entire flow from data acquisition to the final output, encapsulating both detection and classification.

2.5 Literature search

The literature search for this survey was carried out in four stages:

1. Find relevant existing surveys.
2. Select a set of about 15 representative papers and read them to get an initial overview of the field.
3. Look at references from the initial selection of papers, as well as other papers written by the same authors, to get a more complete understanding.
4. Search for public databases of traffic sign images.

The first step was to find out whether any surveys existed already. This step was actually carried out before it was decided to do a detailed survey. If any comprehensive survey papers had existed already, the subsequent steps would have been unnecessary. As mentioned earlier, the literature features just two surveys on TSR, none of them very comprehensive.

Step 2 was carried out by looking at references in the surveys that did exist as well as doing searches in several scientific databases. When a selection of papers had been found, they were read and the foundations for the tables in section 2.6 were created.

For step 3 a larger set of papers was selected, bringing the total count up to 41. This was done by following references in the previous pool of papers, looking for further publications by known authors, and simply doing more general searches.

After having read the papers, it was clear that almost no papers used the same datasets for testing, so it was necessary to determine whether any public datasets existed at all. That was step four. Some datasets do exist, but they are very new and not well advertised, so they were mainly found by using references in a few of the papers.

2.6 Results

2.6.1 Public sign databases

A few publicly available traffic sign datasets exist:

- German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2011, 2012)
- KUL Belgium Traffic Signs Dataset (KUL Dataset) (Timofte et al., 2011)
- Swedish Traffic Signs Dataset (STS Dataset) (Larsson and Felsberg, 2011)
- RUG Traffic Sign Image Database (RUG Dataset) (Grigorescu and Petkov, 2003)
- Stereopolis Database (Belaroussi et al., 2010)

Information on these databases can be found in table 2.2. Most of the databases have emerged within the last two years (except for the very small RUG Dataset), and are not

Table 2.2: Information on the publicly available sign databases.

	GTSRB	**STS Dataset**	**KUL Dataset**	**RUG Dataset**	**Stereopolis**	**LISA Dataset**
Number of classes:	43	7	100+	3	10	47
Number of annotations:	50000+	3488	13444	0	251	7855
Number of images:	50000+	20000	9006	48	847	6610
Annotated images:	All images	4000 images	All images	0	All images	All images
Sign sizes:	15x15 to 250x250 px	3x5 to 263x248 px	100x100 to 1628x1236 px	N/A	25x25 to 204x159 px	6x6 to 167x168 px
Image sizes:	15x15 to 250x250 px	1280x960 px	1628x1236 px	360x270 px	1920x1080 px	640x480 to 1024x522 px
Includes videos:	No	No	Yes, 4 tracks	No	No	Yes, for all annotations
Country of origin:	Germany	Sweden	Belgium	The Netherlands	France	United States
Extra info:	Images come in tracks with 30 different images of the same physical sign.	Signs marked visible/blurred/occluded and whether they belong to the current road or a side road.	Includes traffic sign annotations, camera calibrations and poses.	Does not include any annotations, only raw pictures.		Images from various camera types.

yet widely used. One of the most widespread databases is the GTSRB, which has been presented by Stallkamp et al. (2011), created for the competition "The German Traffic Sign Recognition Benchmark". The competition was held at the International Joint Conference on Neural Networks (IJCNN) 2011. It is a large data set containing German signs, thus very suitable for training and testing systems aimed at signs adhering to the Vienna Convention. A sample image from the GTSRB database can be found in fig. 2.5a. The GTSRB is primarily geared towards classification, rather than detection, since each image contains exactly one sign without much background. For detection, images of complete scenes is necessary. Also, many detection systems rely on a tracking scheme to make detection more robust and without video of the tracks (in GTSRB parlance a "track" is a set of images of the same physical sign), this will not work properly. Since the data set is created for the classification task, this is not so much a problem of that database, as it is a testament to its target. In conjunction with the competition, five interesting papers were released (Rajesh et al., 2011; Ciresan et al., 2011; Zaklouta et al., 2011; Sermanet and LeCun, 2011; Boi and Gagliardini, 2011). They all focus on classification rather than detection.

Two other datasets should be highlighted: The STS Dataset and the KUL Dataset. They are both very large, though not as large as the GTSRB, and they contain full images. This means that they can both be used for detection purposes. The STS

(a) **(b)**

Figure 2.5: Example sign images from (a) the GTSRB and (b) the STS Dataset with the sign bounding boxes superimposed.

Dataset does not have all images annotated, but it does include all frames from the videos used to obtain the data. This means that tracking systems can be used on this dataset, but it can only be verified with ground truth every 5 frames. An example from the STS Dataset can be seen in fig. 2.5b. The KUL Dataset also includes 4 recorded sequences which can be used for tracking experiments. KUL also includes a set of sign-free images which can be used as negative training images and it has pose-information for the cameras for each image.

The details of the dataset assembled in this project are also listed in table 2.2. The dataset is described further in chapter 3.

2.6.2 Detection methods

The approaches in the detection stage have traditionally been divided into two kinds:

- Color based methods.
- Shape based methods.

Color based methods take advantage of the fact that traffic signs are designed to be easily distinguished from their surroundings, often colored in highly visible contrasting colors. These colors are extracted from the input image and used as a base for the detection. Just like signs have specific colors, they also have very well defined shapes that can be searched for. Shape based methods ignore the color in favor of the characteristic shape of signs.

Each method has its pros and cons. Color of signs, while well defined in theory, varies a lot with available lighting, as well as with age and condition of the sign. On the other hand, searching for specific colors in an image is fairly straight forward. Sign shapes are invariant to lighting and age, but parts of the sign can be occluded, making the detection harder, or the sign may be located at a background of a similar color, ruining the edge detection that most shape detectors rely on.

The division of systems in this way can be problematic. Almost all color based approaches take shape into account after having looked at colors. Others use shape detection as their main method, but integrate some color aspects as well. Instead, the detection can be split into two steps as proposed by Gomez-Moreno et al. (2010): Segmentation and detection. Here, the detection step is split further into a feature extraction step and the actual detection, which acts on the features that are extracted. Many shape-only based methods have no segmentation step. The flow is outlined in fig. 2.6.

2.6.3 Surveyed papers

An overview of all surveyed papers and their methods is listed in table III in the survey paper included in appendix A. To conserve space, the tables are not reproduced here. It contains each of the systems and lists which segmentation method, feature type, and detection method that is used. The author group numbers are used to mark the papers that are part of an ongoing effort from the same group of authors. They do not constitute a ranking in any way. In tables IV and V in the survey paper, some of their more detailed properties are listed. The systems are split into two tables. Table IV displays those which do not use any tracking. Table V contain those which do use tracking, something that is crucial when using TSR in a driver assistance context, as mentioned earlier. Apart from this division, the two tables are structured in the same way: *Sign type in paper* describes which sign types the authors of the paper have attempted to find, while emphsign type possible are the types of signs the method could be extended to include, usually a very broad group. *Real-time* is about how fast the system runs, if that information is available. Any system with a frame rate faster than 5 fps is considered to have real-time potential. *Rotation invariance* tells whether the used technique is robust to rotation of signs. *Model vs. training* describes if the detection system relies on a theoretical model of signs (such as a pre-defined shape), if it uses a learned type of classifier, or if it uses a combination of the two. *Test image type* is the image resolution the system is designed to work with. Low-res images are usually video frames, while high-res are still images.

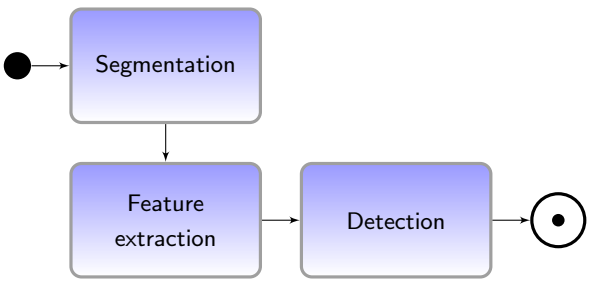


Figure 2.6: The general flow followed by typical sign detection algorithms.

The detection performance of the surveyed papers are presented in table VI, also in the survey paper included in appendix A. Very few papers use common databases to test their performance and the papers detect various types and numbers of signs. Thus, the numbers should not be directly compared, but nevertheless they give an idea of performance. Not all papers give all the measures reported in the table (detection rate, false positives per frame, etc.), so some fields in the table could not be filled. In other cases these exact measures were not given, but could be calculated from other given numbers. Where figures are available, the best detection rate the system obtained is reported along with the corresponding measure of false positives. The detection rate is per frame, meaning that 100% detection is only achieved if a sign is found in every frame it is present. It is not sufficient to just detect the sign in a few frames. This is the way results are presented in most papers, so this is the measure chosen here, even if a real- world system would work fine if each sign is just detected once. Papers which only report the per-sign detection rate as opposed to the per-frame detection rate are marked with a triangle in the right-most column of the table.

Different papers report the false positives in different ways, so a few different measures - which are not directly comparable - are presented in the table:

FPPF False postives per frame: $FPPF = \frac{FP}{f}$ where FP is the number of false positives and f is the number of frames analyzed.

FPR False positive rate: $FPR = \frac{FP}{N}$ where N is the number of negatives in the test set. This measure is rarely used in detection, since the number of negatives does not always make much sense (how many negatives exist in a full frame?).

PPV Positive predictive value: $PPV = \frac{TP}{TP+FP}$ where TP is the number of true positives.

FPTP False/true positive ratio: $FPTP = \frac{FP}{TP}$

WPA Wrong pixels per area: $WPA = \frac{WP}{AP}$ where WP is the number of wrongly classified pixels and AP is the total number of pixels classified.

When papers present results for different sign types, the mean detection performance is also presented in the table. In many cases that will give a better view of the true performance of the approach.

Five papers stick out, claiming a 100% detection rate. Gil Jiménez et al. (2008) test only on synthetic data. It is possible that the synthetic data does not fully encapsulate real world variations, so the performance of that approach is not guaranteed to be as good in real-world scenarios. At first glance Ruta et al. (2010) achieve a 100% detection rate, but that is only the case for one of their sign types. The mean performance is a more accurate (and still promising) gauge of the actual performance. The same is the case for Larsson and Felsberg (2011). Loy and Barnes (2004) detect all signs in the test set, but at the cost of a large number of false positives per frame. Hoferlin and

Zimmermann (2009) only present the per-sign detection rate, so that figure cannot be compared to the other systems.

Generally, systems achieve detection rates well into the 90% range, some at very low false detection rates. From the table no "best system" can be chosen, since the test sets are very different, both in size and content. A system that can detect several different sign types at a low detection rate may in some applications be considered better than a system that can only detect one specific sign type, but do that very well. Good options are presented by Timofte et al. (2009); Baro et al. (2009); Gu et al. (2011); Overett and Petersson (2011). They have all been tested on large datasets and report detection rates above 90% with a decent low number of false positives.

Now that the basics about sign detection are in place, the following sections go in depth with how recent papers perform each step.

2.6.4 Segmentation

The purpose of the segmentation step is to achieve a rough idea about where signs might be, and thus narrow down the search space for the next steps. Not all authors make use of this step. Since the segmentation is traditionally done based on colors, authors who believe this should not be part of a sign detection often do not have any segmentation step, but go directly to the detection.

Of the papers that do use segmentation, all except Gu et al. (2011); Keller et al. (2008) use colors to some extent. Normally, segmentation is done with colors and subsequently a shape detection is run in a later stage. Gu et al. (2011) reverses the usual order, so they use radial symmetry voting (see section 2.6.6) for segmentation and a color based approach for the detection. Keller et al. (2008) also run radial symmetry voting as preprocessing, but follow it up with a cascaded classifier using Haar wavelets (see again section 2.6.6).

Generally, color based segmentation relies on a thresholding of the input image in some color space. Since many believe that the RGB color space is very fragile with regards to changes in lighting, these methods are spearheaded by the HSI-space (or its close sibling, the HSV-space). HSI/HSV is used by Kuo and Lin (2007); Nguwi and Kouzani (2008); Ren et al. (2009); Xu (2009); Chiang et al. (2010); Qingsong et al. (2010). The HSI-space models the human vision better than RGB and allows some variation in the lighting, most notably in the intensity of light. Some papers, like the ones by Vázquez-Reina et al. (2005); Maldonado-Bascon et al. (2007); Gil Jiménez et al. (2008); Lafuente-Arroyo et al. (2010), augment the HSI thresholding with a way to find white signs. Hue and saturation are not reliable for detecting white, since it can be at any hue, so they use an achromatic decomposition of the image proposed by Liu et al. (2002).

(a) Before thresholding **(b)** After thresholding

Figure 2.7: An example of thresholding, looking for red hues.

Some authors are not satisfied with the performance of HSI, since it does not model the change in color temperature in different weather, but only helps in changing light intensity. Gao et al. (2006, 2008) instead threshold in the LCH color space, which is obtained using the CIECAM97-model. This allows them to take variations in color temperature into account. The RGB space is used by Timofte et al. (2009); Prisacariu et al. (2010), but they use an adaptive threshold in an attempt to combat instabilities caused by lighting variations.

Of special interest in this color space discussion is the excellent paper by Gomez-Moreno et al. (2010), which has shown that HSI-based segmentation offers no significant benefit over normalized RGB, but that methods which use color segmentation generally perform much better than shape-only methods. They do, however have trouble with white signs. For a long time, it has simply been assumed that the RGB color space was a bad choice for segmentation, but through rigorous testing, they show that there is nothing to gain from switching to the HSI color space instead of a normalized RGB space. As the authors write: "Why use a nonlinear and complex transformation if a simple normalization is good enough?".

A color based model not relying on thresholding was put forward by Deguchi et al. (2011), which use a cascaded classifier trained with AdaBoost, similar to the one proposed by Viola and Jones (2001), but on Local Rank Pattern features instead of Haar wavelets. Also, Ruta et al. (2010) use a color-based search method that, while closely related to, is not directly thresholding-based. Here, the image is discretized into colors that may exist on signs. The discretization process is less destructive than thresholding in that it does not directly discard pixels, instead it maps them into the closest sign-relevant color. In a more recent contribution (Ruta et al., 2011), they replace the color discretization method with a Quad-tree interest region finding algorithm, which finds interesting areas using an iterative search method for colored signs. In the same realm lies Houben (2011), who use a learned probabilistic color preprocessing.

Kastner et al. (2010) propose a unique approach: Using a biologically inspired attention system. It produces a heat map denoting areas where signs are likely to be found. An example can be seen in figure 2.8. A somewhat similar system was put forth by Xie et al. (2009), who uses a saliency measure to find possible areas of interests.

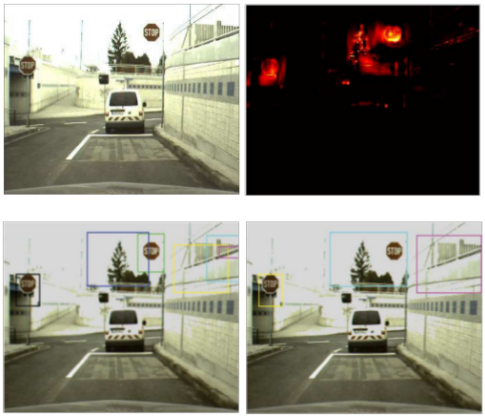


Figure 2.8: The biologically inspired detection stage from Kastner et al. (2010). Image source: Kastner et al. (2010)

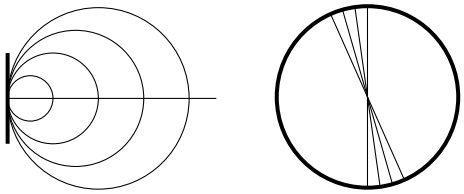
2.6.5 Features and modeling

While various features are available from the vision literature, the choice of feature set is often closely coupled with the detection method, though some feature sets can be used with a selection of different detection methods. The most popular feature is edges - sometimes edges obtained directly from the raw picture, sometimes edges from pre-segmented images. Edges are practically always found using a Canny edge detection or some method very similar, and they are used as the only feature by Ruta et al. (2010, 2011); Loy and Barnes (2004); Barnes and Loy (2006); Barnes et al. (2008); Nunn et al. (2008); Meuter et al. (2011); Garcia-Garrido et al. (2011); Gonzalez et al. (2011); Timofte et al. (2009); Liu et al. (2002); Kuo and Lin (2007); Moutarde et al. (2007); Belaroussi and Tarel (2009); Ren et al. (2009); Chiang et al. (2010); Qingsong et al. (2010); Deguchi et al. (2011); Houben (2011). Prisacariu et al. (2010) combine the edges with Haar-like features and Ruta et al. (2007); Hoferlin and Zimmermann (2009) look only at certain color filtered edges.

Even though edges comprise the most popular feature choice, there are other options. Histogram of Oriented Gradients (HOG) is one. It was first used to detect people in images, but has been used by Alefs et al. (2007); Pettersson et al. (2008); Overett and Petersson (2011); Gao et al. (2006); Xie et al. (2009) to detect signs. HOG is based on creating histograms of gradient orientations on patches of the image and comparing them to known histograms for the sought after objects. HOG is also used by Creusen et al. (2010), but they augment the HOG feature vectors with color information to make them even more robust.

A number of papers by Bahlmann et al. (2005); Keller et al. (2008); Prisacariu et al. (2010); Baro et al. (2009) use Haar wavelet-like features, Bahlmann et al. (2005) only on certain colors, and Baro et al. (2009) in the form of so-called dissociated dipoles with wider structure options than traditional Haar wavelets.

More esoteric choices are Distance to Bounding box (DtB), FFT of shape signatures,



(a) Possible circles for a gradient

(b) Intersecting vote lines

Figure 2.9: The basic principle behind the radial symmetry detector. Image inspired by Barnes et al. (2008).

tangent functions, simple image patches, and combinations of various simple features. DtB, as used by Maldonado-Bascon et al. (2007); Lafuente-Arroyo et al. (2010), are a measure of distances from the contour of a sign-candidate to its bounding box. Similarly, the FFT of shape signatures used by Gil Jiménez et al. (2008) is based on the distance from the shape center to its contour at different angles. Tangent functions, used by Xu (2009), calculate the angles of the tangents at various points around the contour. Simple image patches (though in the YCbCr color space) are championed by Nguwi and Kouzani (2008) and a combination of simple features, such as corner positions and color is used by Kastner et al. (2010).

2.6.6 Detection

The detection stage is where the signs are actually found. This is in many ways the most critical step, and often also the most complicated. The selection of detection method is a bit more constrained than the previous two stages, since the method must work with the features from the previous stage. The decision is therefore often made the other way around: A desired detection method is chosen, and the feature extraction stage is designed to deliver what is necessary to perform the detection. As known from the previous section, the most popular feature is edges, and this reflects on the most popular choice in detection method. Using Hough transforms to process the edges is one option, as done by Moutarde et al. (2007); Garcia-Garrido et al. (2011); Gonzalez et al. (2011); Ren et al. (2009). Moutarde et al. (2007) use a proprietary and undisclosed algorithm for detection of rectangles in addition to the Hough transform used for circles. That said, Hough transforms are computationally expensive and not suited for systems with real-time requirements. Because of that, the most popular methods are derivatives of the radial symmetry detector first proposed by Loy and Zelinsky (2003) and first put to use for sign detection by Barnes and Zelinsky (2004). The algorithm votes for the most likely sign centers in an image based on symmetric edges and is itself inspired by the Hough transform. The basic principle can be seen in fig. 2.9. In a circle, all edge gradients intersect at the center. The algorithm finds gradients with a magnitude above a certain threshold. In the direction pointed out by the gradient, it casts a vote

Figure 2.10: Votes from a radial symmetry system superimposed to the original image. The brightest spot coincides with the center of the sign. This image is from a system developed in conjunction with this paper and is a radial symmetry voting algorithm extended to work for rectangles

in a separate vote image. It looks for circles of a specific radius and thus votes only in the distance from the edge that is equivalent to the radius. The places with most votes are most likely to be the center of circles.

This algorithm was later extended to regular polygons by Loy and Barnes (2004) and a faster implementation for sign detection use was proposed by Barnes and Loy (2006). It is also used in some form by Barnes et al. (2008); Keller et al. (2008); Meuter et al. (2011); Hoferlin and Zimmermann (2009); Gu et al. (2011); Nunn et al. (2008). An example of votes from a system which is extended to work for rectangular signs can be seen on fig. 2.10. An alternate edge-based voting system is proposed by Belaroussi and Tarel (2009).

The HOG features can be used with an SVM, as done by Xie et al. (2009); Creusen et al. (2010), or be compared by calculating a similarity coefficient as Gao et al. (2006) does. Another option with regard to HOG is to use a cascaded classifier trained with some type of boosting. This is done by Pettersson et al. (2008); Overett and Petersson (2011). Cascaded classifiers are traditionally used with Haar wavelets, and sign detection is no exception, as presented by Bahlmann et al. (2005); Keller et al. (2008); Prisacariu et al. (2010); Baro et al. (2009).

Finally, also neural networks and genetic algorithms are represented by Nguwi and Kouzani (2008) and Liu et al. (2002), respectively.

The detection stage reflects the philosophical difference that was also seen in the feature extraction stage: Either reliance on a simple, theoretical model of sign shapes is preferred - at this stage it is nearly always shapes that are searched for - or reliance on training data and then a more abstract detection method. Since it is extremely hard to compare systems tested across different data sets, it is not clear which methods perform the best, so that is clearly an area with a need for further studies. Both ways can be fast enough for real-time performance, and most of them could also work with signs

of any shape. There are outliers using different methods, but no compelling argument that they should perform significantly better.

2.7 Discussion

In the previous sections, different methods and philosophies for each stage are presented. This section discusses the current state of the art and outlines ideas for future directions of research.

At the moment, the problem in TSR is the lack of use of standardized sign image databases. This makes comparisons between contributions very hard. In order to obtain meaningful advances in the field, the development of such databases is crucial. Until now, research teams have only implemented a method they believe has potential, or perhaps tested a few solutions. Without a way to compare performance with other systems, it is not clear which approaches work the best, so every new team starts back at square one, implementing what *they* think might work best. Two efforts to remedy this situation deserve to be mentioned: The sign databases presented earlier and the segmentation evaluation done by Gomez-Moreno et al. (2010). As mentioned earlier (section 2.6.1), a few public sign databases have recently emerged, but have not yet been widely used. Gomez-Moreno et al. (2010) compare various segmentation methods on the same data set containing a total of 552 signs in 313 images. They also propose a way to evaluate the performance of segmentation methods. That paper provides a very good starting point for determining which segmentation method to use.

These two efforts notwithstanding, public databases covering signs from non-Vienna Convention regions are necessary. Databases which include video tracks of signs would also be very beneficial to the development of TSR systems, since many detectors employ a tracking system for signs. This is, to some extent, included in the KUL Dataset.

The absence of usage of public database may not explain in entirety why very few comparative studies of methods exist. Another reason is that TSR systems are long, complex chains of various methods, where it is not always possible to swap individual modules. When it is not feasible to swap, say, the detection method for something else, it is naturally hard to determine whether other solutions may be better. This is solved, if more papers divide their work more clearly into stages, ideally as fine grained as the ones used in this survey, plus a similar set of stages for classification. This is done with success by Gomez-Moreno et al. (2010), as they test different segmentation methods while keeping the feature extraction, detection, and classification stages fixed.

Another problem is the need for work on TSR in regions not adhering to the Vienna Convention. The bulk of the existing work comes out of Europe, Australia, and Japan. Japan and Australia are not parts of the Vienna Convention, but they use similar signs, for example to convey speed limits. Of the surveyed papers here, only two are concerned with US traffic signs (Keller et al., 2008; Moutarde et al., 2007), and even

Figure 2.11: Example of sign relevancy challenges in a crop from our own collected data set. The signs have been manually highlighted, and while both signs would likely be detected, only the one to the right is relevant to the driver. The sign to the left belongs to another road, the one the black and white cars come from.

they only look at speed limit signs.

When looking at sign detection from a driver-in-the-loop perspective, it is also unfortunate that the bulk of research now focuses on speed limit signs. A wealth of papers cite driver assistance as their main application, but carries on focusing on speed limit signs. Detection of speed limits is highly relevant for an autonomous vehicle, but as it turns out, humans are already very good at seeing speed limit signs themselves (Shinar, 2007). As such, recognition of signs other than speed limit is actually more interesting.

The final problem to be highlighted in this section is the relation of signs to the surroundings. TSR has seen significant work, as is evident from this paper, but little work has been done on ensuring that the detected signs are relevant for the ego-car (with the notable exception of Garcia-Garrido et al. (2011)). In many situations, it can occur that a detected sign is not connected to the road the car is on. An example from our own collected data can be seen in fig. 2.11. In this case, two stop signs can be seen, but only the rightmost one pertains to the current road. Similar situations occur often on freeways, where some signs may only be relevant for exit lanes. Related to this problem is that when the driver changes to a different road, most often the restrictions from earlier detected signs no longer apply. This should be detected and relayed to the system. It is very likely that research in other areas, such as lane detection can be of benefit here. Another idea with regard to the surroundings would be to link knowledge of weather and current lighting conditions to enhance the robustness of the detector, similar to what is done for detection of people by Doshi and Trivedi (2007). It is also possible that vehicle dynamics can be taken into account and used in the tracking of detected signs.

3

LISA Traffic Sign Dataset

When doing computer vision projects, training data and test data is essential. If algorithms that train themselves based on images are used, the need of training data is obvious. However, also simpler systems need training data that the algorithms can run on, to asses and adjust their performance. A test set is necessary for evaluating system performance. It is important that the system is not tested on the same data as it is trained on, since that cannot tell if the system generalizes to other data.

This chapter is about the traffic sign dataset that has been created in this project. The survey in chapter 2 showed that no dataset with American signs existed, an given that the ultimate goal of this project was to investigate detection of US traffic signs, creating such a database was necessary.

3.1 Introduction

When training and testing a system like this, it is generally preferable to use data from existing public databases as opposed to collecting data for the particular project. Not only is it saving time and effort, it is also convenient for comparing results to previous works in the area.

In more established fields, such as detection of people, a number of public databases exist. Notable examples are the MIT CBCL Pedestrian Database and the INRIA Person Dataset (Dalal and Triggs, 2005). In the realm of TSR, a few databases has recently emerged, as described in chapter 2, but none of them include US signs.

The recent emergence of these datasets is a very welcome addition to the field of TSR, as the performances of methods in previous papers have been very hard to compare without actually implementing them. Considerations on how the dataset created for this project has been assembled are presented in the following sections.

3.2 Methods

This section describes the methods used to obtain the LISA Traffic Sign Dataset. It describes both considerations about the actual data content, and the technical means with which is was collected. During this section there will be no distinction between training and test data. In the end, splitting the dataset into separate training and test pools is discussed.

3.2.1 Dataset content and structure

The design and content of the real-world dataset has been inspired by the other available datasets presented earlier. The more similarly packaged the different datasets are, the easier they are to use, so some effort has gone into this.

All significant existing datasets contain a number of images annotated with the type and position of signs. That is the bare minimum required for any traffic sign dataset. The German Traffic Sign Recognition Benchmark (GTSRB) dataset has significantly more classes and pictures than the STS dataset. The KUL set is also large and contains the most classes of all. However, the STS and KUL datasets one significant advantage: They include full frames, making them useful for both detection and classification systems. This is critical, especially since this project is on detection. Furthermore, the STS dataset includes additional annotation data: Information on whether a sign is visible, blurred or occluded, and whether it belongs to the current road or a side road. As described in chapter 2, the task of determining whether a recognized sign actually belongs to the current road is not yet well explored, and including this information in datasets lays the groundwork for this effort. All datasets save their annotations in comma separated text files (csv-files) in slightly different formats.

The dataset from this report - named the LISA dataset from the name of the Laboratory for Intelligent and Safe Automobiles at UCSD - tries to take the best from each dataset while also adding even more data. None of the existing databases include video to any large extent (the KUL dataset has 4 video tracks). They all have a large collection of annotated single images, but this means that they cannot be used to test detectors relying on temporal information. Many systems already use various tracking schemes to minimize the number of false positives, and it is quite likely that in the future, detections using temporal data even more will emerge. Therefore, the LISA dataset includes video as well as stand alone frames. The organization of the dataset can be seen in fig. 3.1 and is described further below.

In order to actually create the dataset, two tools were developed: Video Annotator and Frame Annotator, each responsible for one part of the annotation process (see below). Care was taken to create the annotation tools in a way so the can be used for general annotation purposes, not only for annotating traffic signs. This way, the programs may also be used for future unrelated projects which require video annotation.

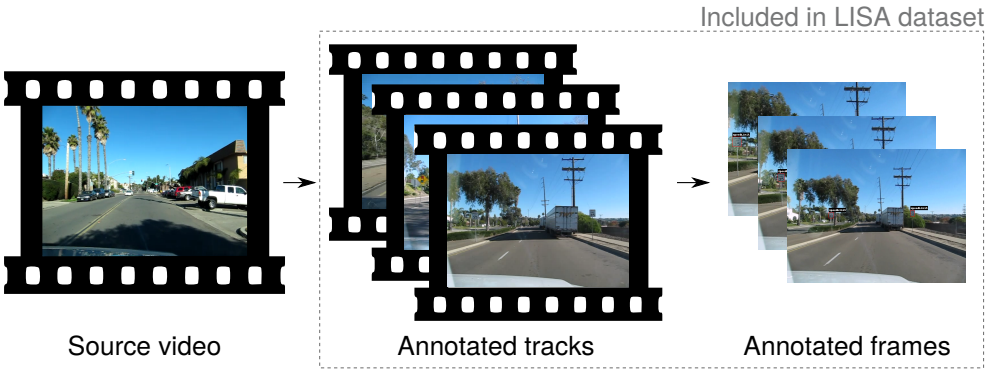


Figure 3.1: An overview of the dataset structure. Each source video is split into a number of tracks. The tracks are annotated with the sign type they contain, but may include other signs. From each track, up to 30 frames are extracted and all signs in these framed are tagged with position, type, and some additional meta data.

Tracks

The source videos are obtained from a driving vehicle. Several different cameras and vehicles have been used to ensure a dataset that is not favored towards a specific setup. Some of the source data is in color, while some of it is grayscale video. Each source video is split into smaller videos which contain signs. These smaller videos are called *tracks*, just like in the GTSRB dataset. Each track has an annotation of the sign that is present in it. In urban driving, it is very common to have several other signs present in one track. This means that the only guarantee about a track is that it contains the sign described by the annotation, but it may still contain other signs as well.

Track annotations do not contain any information on the pixel-wise position of the annotated sign, simply that the sign is present. If pixel positions are desired, they can be inferred from the frame annotations discussed below. Along with the sign type, the annotation of each track includes information on which source file the track comes from and exactly which frames it consists of. This information ensures the traceability of the track to its source.

The technical details and the exact format of the tracks and their annotations are covered below in section 3.2.2.

Frames

A number of frames are extracted from each track and annotated in detailed fashion. Each extracted frame is annotated with the exact position of all signs present in the image. This means that even if a track is annotated as a speed limit sign-track, all other signs in the frames will be annotated too. Inspired by the STS dataset, each sign annotation contain this information:

Tag: The type of sign as a string without spaces, e.g. speedLimit or pedestrianCrossing.

Position: The rectangular bounding box of the sign, given as the upper left and lower right corner.

Occluded: True if the sign is partially occluded.

On side road: True if the sign does not belong to the road currently being driven on, but instead to a side road.

Similar to the track annotations, additional traceability information is included so every annotated frame can be traced back to both its origin track and origin source video. Contrary to the STS dataset, the frame annotations in the LISA database does not contain information on whether a particular sign is blurred. This is omitted since it seems to be a subjective measure and of limited use.

For the LISA dataset, it was decided not to annotate all frames in tracks. In each track a maximum of 30 equidistant frames were annotated, an the annotated frames were required to be at least 5 frames apart. This ensured that long tracks are not overrepresented in the dataset and that there is some variation from sign to sign, since two adjacent frames are likely to produce very similar signs. It also lessens the work load of the annotator.

The technical details for the frame annotations are covered in section 3.2.3.

3.2.2 Video annotation tool

The program Video Annotator was implemented in C++ using OpenCV for image and video I/O and drawing, and Qt for the user interface. An OpenGL based custom Qt widget was used for connecting Qt and OpenCV. A screenshot can be seen in fig. 3.2. A short guide to the usage of Video Annotator can be found in appendix D, so the actual usage and interface of the program will not be discussed here. It should be mentioned, though, that it was created to have a minimum of visual clutter and with keyboard shortcuts to make the usage as efficient as possible.

The output from Video Annotator is the most interesting thing to discuss here. The main output format is a plain text file with annotations. This format was chosen for several reasons:

- A text-based format is standard and used in all other available databases.
- It is human readable, making manual lookups possible for debugging of tools.
- Unix/Linux contains a very comprehensive suite of text-file handling tools, which helps when handling thousands of annotations in many files.
- Developing tools to handle text-based formats is easy and does not depend on the ability to decode and encode obscure binary formats.

When a video is annotated, a semi-colon separated text file with the annotations is created with the name of the video file as a base. If the video is named `vid_cmp2.avi`, the

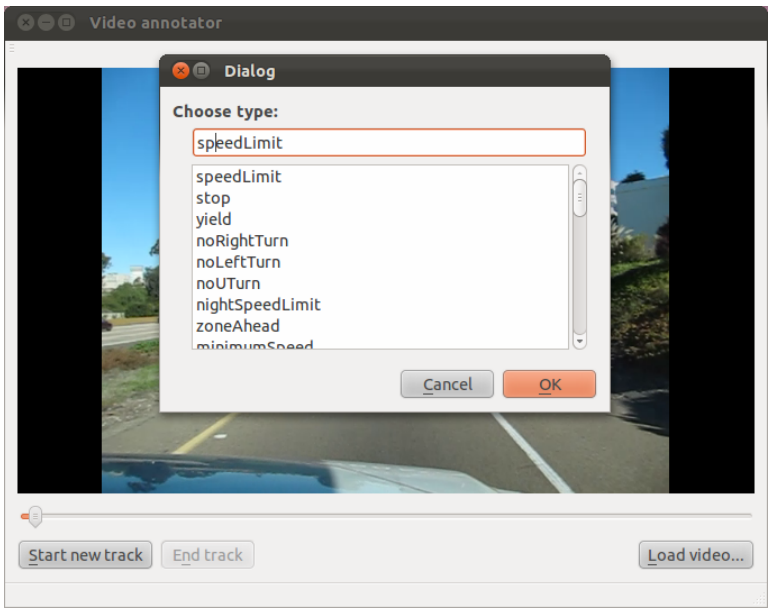


Figure 3.2: Screenshot of Video Annotator

Listing 3.1: The header and two sample annotations in a video annotation file

```
1 Filename;Track type;Origin file;First frame;Length
2 stop_1323802788.avi;vid0/vid_cmp2.avi;10243;145
3 yield_1323802820.avi;yield;vid0/vid_cmp2.avi;10963;30
```

annotation file is name `vid_cmp2.avi_annotations.csv`. Each line contains an annotation of a track in the format listed in listing 3.1.

The first column, `Filename` contains the name of the other output Video Annotator produces: An XviD compressed video file containing the track. The track video is named after the sign in the track, and a creation timestamp is created. The timestamp acts as a unique key for that particular track, ensuring no naming conflicts between tracks with the same sign appear, even in tracks across different source videos. Another option would be to use a video-content based hash of some sort. That might be necessary if the annotation program should work in an environment where multiple videos are annotated in parallel (so to tracks could theoretically be created in the same second), but a timestamp was deemed secure enough for these purposes and chosen due to its simplicity. `Track type` is the sign type the video contains. As mentioned earlier, there is no guarantees that no other signs are present in the track. To ensure traceability, the remaining three columns contain a reference to the original video file and frame numbers so the video of the track can be recreated as long as the original file is retained.

The output file from Video Annotator can be loaded into Frame Annotator, or used stand-alone, for example to extract all tracks which contain a specific sign. This could be relevant a as test pool for a detector.

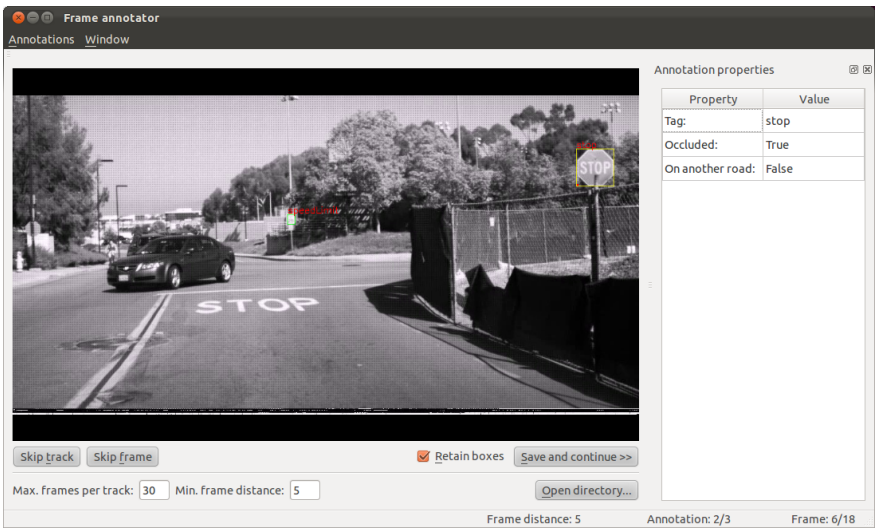


Figure 3.3: Screenshot of Frame Annotator

3.2.3 Frame annotation tool

Frame Annotator was created using the same tools as video annotator: C++ with OpenCV for image and video I/O and drawing, and Qt for the user interface. An OpenGL based custom Qt widget was used for connecting Qt and OpenCV. A short user guide for Frame Annotator is included in appendix E. A screenshot is in fig. 3.3.

Frame Annotator is a more complex program, both implementation- and interface-wise. It is built to work on top of Video Annotator, but as described in the user guide, it can be used with stand alone videos if necessary. The philosophy of Video Annotator is carried over: The output consists of a text-file with annotations and it also saves all annotated frames as PNG image files. The basic idea is simple: The program allows the user to draw rectangular annotations around objects and attach a tag to them - in this case the sign type. The user can annotate multiple signs in the same frame and also save various meta data for each annotation. The program can be set up to annotate all frames in a track, or a maximum number of frames with a minimum internal distance as discussed earlier.

The main output file is called `frameAnnotations.csv` and a short example can be seen in listing 3.2. Each line is an annotation, so if a frame contains several signs, it will have several lines in the annotation file. The two first columns contain the filename for the image of the frame and the tag (sign type) associated with the image. Then four columns describe the location of the annotation. The next column is dynamic: It contains a comma-separated list of binary meta data for the annotation. Because Frame Annotator is flexible and can be used with any number of meta-data fields, but tools should be able to expect at fixed number of columns, all meta data are put in the same column. Finally, four columns provide traceability, both back to the origin track, but also back to the origin video. This means that any annotation made in Frame Annotator can be carried all the way back to the original video and used to test the

Listing 3.2: The header and 5 sample annotations in a frame annotation file. Extra spaces have been added for readability.

```
1 Filename;Annotation tag;Upper left corner X;Upper left corner Y;Lower ¶
    right corner X;Lower right corner Y;Occluded,On another road;Origin¶
     file;Origin frame number;Origin track;Origin track frame number
2
3 yield_1323802820.avi_image0.png;yield;651;38;684;66;0,0;vid0/vid_cmp2.¶
    avi;10965;yield_1323802820.avi;2
4
5 yield_1323802820.avi_image1.png;yield;660;42;693;70;0,0;vid0/vid_cmp2.¶
    avi;10970;yield_1323802820.avi;7
6
7 yield_1323802820.avi_image2.png;yield;668;36;701;65;0,0;vid0/vid_cmp2.¶
    avi;10975;yield_1323802820.avi;12
8
9 keepRight_1323802829.avi_image1.png;keepRight;229;45;250;68;0,0;vid0/¶
    vid_cmp2.avi;11134;keepRight_1323802829.avi;46
10
11 noLeftTurn_1323803031.avi_image0.png;noLeftTurn;62;29;86;51;0,0;vid0/¶
    vid_cmp2.avi;16018;noLeftTurn_1323803031.avi;2
```

performance of a detector which takes advantage of tracking or needs video for other reasons.

3.2.4 Annotation handling tools

Apart from the to tools to do annotations, a set of tools to handle the annotation files was created. The set consists of 4 Python scripts accessed through a command line interface:

- `mergeAnnotationFiles.py`
- `splitAnnotationFiles.py`
- `extractAnnotations.py`
- `evaluateDetections.py`

The tools have been tested with Python 2.7.3, but care was taken to make the code Python 3 compatible, so provided all dependencies exist for Python 3, they should be easily portable. Instead of a user guide, each tool has a comprehensive help-page which explains the command line parameters. It can be accessed using the `-h` parameter. Example: `python extractAnnotations.py -h`.

`mergeAnnotationFiles.py` can combine multiple annotation files into one. When using Frame Annotator, one `frameAnnotations.csv`-file is created for each source video and put in a subfolder. `mergeAnnotationFiles.py` crawls a directory structure at combines all the annotation files into one, while changing the file paths to be relative to the new, large annotation file. It works with both video annotation files and frame annotation files.

It is used to create an easy-to-use index of all annotations in the same format as the individual annotation files.

`splitAnnotationFiles.py` is used to randomly split annotation files into two. It is most commonly used on the combined annotation file to create random sets of training and test data. It allows for the user to specify the split percentage (for example 80% to one file and 20% to the other) and is it also possible to create a split only for a specific sign type. Creating an 80/20 split of all stop signs can for example be done with the command `python splitAnnotationFiles.py -f stopSign 80 allAnnotations.csv`.

`extractAnnotations.py` is the most powerful of the annotation handling tools. It also operates on annotation files, most commonly the combined one. It output is always a set of images in a folder called `annotations`. It has several use cases, which can all be used with a type-filter:

Copy : This mode simply copies all the full images to a different directory. It can be used to easily extract all images containing a stop sign, for example.

Mark : Marks the annotations on the full frames and saves them to the output folder. Good for manually evaluating annotations.

Black-out : Puts a black box over all annotations. This can be used to black out signs that should not be present in test-images.

Crop : Crops the images at the annotation. A margin can be set. Used for extracting training images from the dataset.

As mentioned, all functions can be used with a tag-filter, but they can also be used with a category-filter. Speed limit signs, for example, span several tags: speedLimit15, speedLimit25, etc., which could each be extracted with the filter switch: `-f speedLimit15¶` . If all speed limit signs, regardless of speed, should be extracted, the category switch can be used instead: `-c speedLimit`. Categories are defined in the file `categories.txt` which must be present in the current working directory.

`evaluateDetections.py` is used to evaluate the performance of a detector. It compares the annotation file (ground truth) with the output of any detector. The detections are given in a csv file. It displays various detection statistics and can be set up to display or save false detections for further manual evaluation. Currently, the script tests if all four corners of the detection are within +/- 10 pixels of the ground truth in both dimensions. In the future, the Pascal detection measure should be used instead. It has been implemented, but too late to be used for the bulk of this project (it was implemented in conjunction with the pedestrian detection project, see chapter 6). Appendix F contains commands which demonstrate the use of these tools along with come convenient Linux commands for handling the dataset.

(a) **(b)**

(c) **(d)**

Figure 3.4: Randomly chosen examples of annotated signs from the LISA dataset.

3.3 Results

Using the tools described above, a database with traffic sign images and videos was collected. It is available for download at *http://cvrr.ucsd.edu/LISA/*, but was too large to be included on physical media with this report. The dataset consists of 7855 annotations on 6610 images. It contains 47 classes, listed in table 3.1 along with the number of instances for each type. Speed limit signs has been broken into types after their denominations. Those which are named Urdbl had an unreadable speed limit, but could still be determined to be speed limit signs. Fig. 3.4 shows some annotated sample images from the dataset. The dataset was assembled from footage from drives around California, mostly in San Diego, with several different vehicles and cameras. The resolution of the full captured frames vary from 640x480 to 1024x522 pixels and the annotations vary from 6x6 to 167x168 pixels. Some images are in color and some are grayscale. A comparison of key stats between this dataset and others is shown on table 2.2 in chapter 2.

3.4 Discussion

From the survey, it was evident that there was a need for a set of US traffic signs and a need for datasets which include full video tracks to allow for development of tracking-based detection systems. As a part of this project such a dataset has been assembled. There has been great emphasis on enabling complete traceability for all annotations

Table 3.1: The content of the LISA Traffic Sign Dataset broken down by sign type.

294	addedLane	34	slow
37	curveLeft	11	speedLimit15
50	curveRight	349	speedLimit25
35	dip	140	speedLimit30
23	doNotEnter	538	speedLimit35
9	doNotPass	73	speedLimit40
2	intersection	141	speedLimit45
331	keepRight	48	speedLimit50
210	laneEnds	2	speedLimit55
266	merge	74	speedLimit65
47	noLeftTurn	132	speedLimitUrdbl
26	noRightTurn	1821	stop
1085	pedestrianCrossing	168	stopAhead
11	rampSpeedAdvisory20	5	thruMergeLeft
5	rampSpeedAdvisory35	7	thruMergeRight
3	rampSpeedAdvisory40	19	thruTrafficMergeLeft
29	rampSpeedAdvisory45	60	truckSpeedLimit55
16	rampSpeedAdvisory50	32	turnLeft
3	rampSpeedAdvisoryUrdbl	92	turnRight
77	rightLaneMustTurn	236	yield
53	roundabout	57	yieldAhead
133	school	21	zoneAhead25
105	schoolSpeedLimit25	20	zoneAhead45
925	signalAhead		

In total: 7855 sign annotations

throughout the annotation tool chain. Two questions must be asked when evaluating such a database: Does it contain enough signs? Does it cover the sign types necessary?

The size of the dataset is in league with the two similar European datasets, the STS dataset and the KUL dataset. However, it spans many sign types, and not all sign types are well represented. Depending on the learning algorithm used for a TSR system, thousands of images might be needed. pedestrian crossing, stop, and signal ahead are the only signs with numbers anywhere near that. If the larger sign categories are evaluated, however, speed limit signs and warning signs are both well represented. A larger dataset would not hurt, but in many situations, especially if training a general purpose warning or speed limit sign detector, the size of the dataset is sufficient.

The dataset does not cover the full MUTCD, which is nearly impossible, but it does cover a good selection of signs. Given that the dataset has been collected from hours of real driving through urban environments, it should represent the real-world class distribution well.

One potential problem with the dataset is that a major part of it is in grayscale. This makes it useless for developing color-dependent detectors. While this sounds like a major drawback, it also reflects reality: The grayscale images originate from the newest vehicle in the laboratory, an Audi A8 outfitted with sensors directly from Audi. Due to cost considerations when mounting cameras in mass produced cars, it is common to use grayscale cameras only for other purposes, such as lane detection. Thus, the dataset reflects reality, and if it forces researchers to focus on shape-based detectors, that is only good, because those are the detectors that can be implemented in today's cars.

4

Synthetic training data

This chapter covers the work presented in the paper *Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets* which was submitted to the ICPR 2012 conference. At the time of writing, no decision has been made as to whether it will be accepted for the conference. Like the previous chapters based on papers, some of the text has been reused and further details has been added in some areas. There was also a poster created for this work, which was presented at the Jacobs Research Expo at UCSD, an annual conference where PhD and graduate students can present their current work. A downsized version of the poster is enclosed in appendix B.

4.1 Introduction

Many sign detection systems (see chapter 2) rely on large amounts of training data to work. Over the past two years, a few traffic sign datasets has appeared, but there are not yet datasets which cover anything outside of Vienna Convention signs or - with the contribution from this project - US signs. Since signs differ from region to region and in many cases from country to country, an interesting proposition is to use synthetically generated training data instead of real images, saving a lot of time and effort in gathering data. Synthetic training data has not yet been widely used in the field of TSR, but given that traffic signs have an appearance well defined by law, it is possible to randomly distort templates to simulate real-world variations.

Synthetic data has not been investigated thoroughly for traffic sign detection yet. Ishida et al. (2006); Hoessler et al. (2007) discuss generation of synthetic data specifically for classification. Overett et al. (2011) investigate some aspects of detecting non-US signs with synthetic data. The detection task is somewhat harder the classification due to the lack of knowledge about whether a sign is present, where it is, and what size it has.

The purpose of this chapter is to train similar detectors with both synthetic and real-world training data and compare their performance. This work required a dataset

to be assembled, but if is showed promising results from the synthetic training data, maybe in the future it would be unnecessary to create new databases for training. Test datasets would still be necessary, but they do not need to be as large.

4.2 Methods

The work in this chapter can be divided into three parts: The generation of synthetic data, creation of a machine learning based detector, and evaluating the detection performance. Each is described in further detail below.

4.2.1 Generating synthetic training data

The idea is to generate synthetic training images from a template which is simply a drawing of the sign that should form the base for the training. Examples can be seen in fig. 4.6 later in this chapter.

The goal is to emulate how signs of the given type might look on pictures from the real world. In order to do this, several transformations are made randomly to the template:

Hue variations emulates faded signs and color casts due to lighting of the natural scene.

Lighting variations emulates shadows and variations in exposure.

Rotations around the x-, y-, and z-axis with the origin in the center of the template. To emulate signs captured from different perspectives.

Backgrounds taken from a real image are added to the template. This emulates the various backgrounds a sign might have in real life.

Occlusions are added in the form of tree branches growing in front of some signs.

Gaussian blur is added to emulate an unfocused camera. It should be noted that Gaussian blur does not really emulate the bokeh produced by an unfocused lens, but emulating bokeh properly is hard, and it would likely not give any notable detection benefit.

Gaussian noise to emulate sensor noise.

The transformations are applied in the order specified in the list above. To yield a wide variety of training images, each transformation is applied with a random parameter within some realistic boundaries. In some cases it is possible that only a subset of the transformations are applied, just as that is possible in real life. Adding/subtracting from the hue value must be implemented so it "wraps around" if the hue value goes out of bounds, since it is an angle measurement.

(a) Base image for backgrounds

(b) Base image for occlusions

Figure 4.1: The two base images from where backgrounds and occlusions are cut.

First step is to convert the template to the HSV color space (Agoston, 2005). The hue and lighting variations are done by adding or subtracting a value to the hue and value parameters, respectively. After this, the image is converted back to the RGB space for the remaining operations. Rotations are carried out by applying a perspective transformation matrix, $\mathbf{T}$ to each pixel in the source image:

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \mathbf{T} \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix}$$

where

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Theta_x) & -\sin(\Theta_x) \\ 0 & \sin(\Theta_x) & \cos(\Theta_x) \end{bmatrix} \begin{bmatrix} \cos(\Theta_y) & 0 & \sin(\Theta_y) \\ 0 & 1 & 0 \\ -\sin(\Theta_y) & 0 & \cos(\Theta_y) \end{bmatrix} \begin{bmatrix} \cos(\Theta_y) & -\sin(\Theta_y) & 0 \\ \sin(\Theta_y) & \cos(\Theta_y) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The angles $\Theta_{x,y,z}$ denote rotations around each axis.

In reality, the transformation is performed as a backwards mapping where T^{-1} is applied to each destination pixel to get the corresponding source pixel. This is done so the destination will not have any blank spots in pixels that does not correspond directly to a pixel in the source.

Backgrounds are cut randomly from an image with a road scene containing no signs (displayed in fig. 4.1a). When the initial template is loaded, a background mask is created based on the transparent layer in the png-file, if such a layer is present. Otherwise, it is created by flooding from the corners. This mask is rotated with the same transformation matrix as the sign so it stays current, and in the background step, all pixels inside the mask are substituted for those from the background patch. Occlusions are added in a similar way, except they are cut from an image with the background masked out.

Gaussian blur is added by convoluting a Gaussian kernel with the image with a random (within the boundaries) radius. Gaussian noise is applied by adding/subtracting

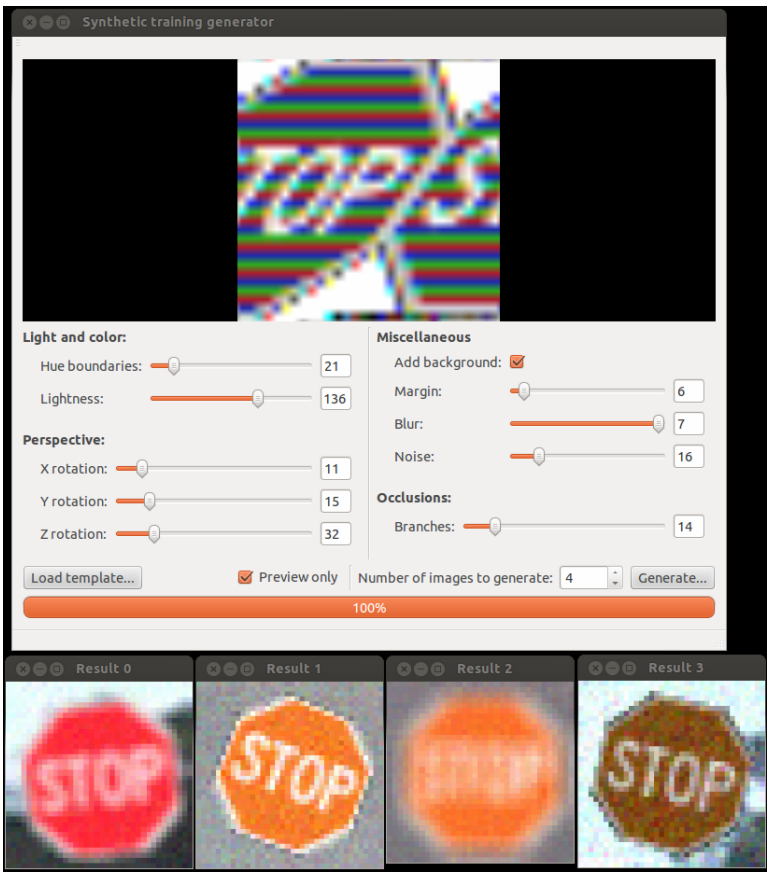


Figure 4.2: Screenshot of the synthetic training generator, showing 4 sample images generated from a template. The distorted template image in the interface is an error and does not affect the output.

a Gaussian random variable to/from each color channel in the image.

In order to perform these transformations, the tool Synthetic Training Generator was created in C++ using OpenCV and Qt. It provides a GUI (see fig. 4.2) in which the boundaries for each parameter can be adjusted. Sample results of any given set of settings can be previewed before the actual training images are generated. On a standard laptop, generating 10000 training images takes a few seconds. Samples of generated images can be seen in fig. 4.7 later in this chapter.

The program and algorithm for generating synthetic training data works for both color and grayscale images, but since the detector chosen for the evaluation (see next section) works only on grayscale, all generated images was converted to that.

4.2.2 AdaBoost cascade on Haar-like features

To evaluate the training data, a machine learning based detection algorithm was necessary. The survey shows several different types, with AdaBoost cascades on Haar-like features (from now on Haar-cascades for short) and HOG+SVM detectors as the most common ones. Several papers use Support Vector Machines (SVMs) on other sets of

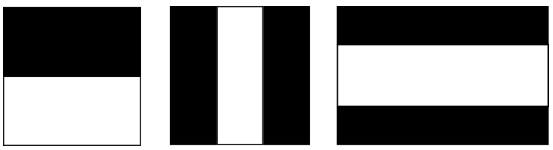


Figure 4.3: Examples of Haar-like features. The value of a single feature is calculated as the difference in total intensity between the white area and the black area.

features as well. In the end, the Haar-cascade was chosen because of its inherent scale invariance and because the OpenCV implementation was easy to use. It is important to remember that this work was not about creating a system with a great detection performance, but rather to compare the detection performance between the same system trained with different training sets.

Haar-cascades were first proposed by Viola and Jones (2001) in a person detection context and has since been applied in a wealth of situations. Due to their cascaded nature and use of integral images, they are very fast and also have a decent detection rate. This section describes how Haar-cascades work. In short, they slide a detection window over an image and calculate a set of learned features. The detector is run as a cascade, where each stage is harder than the previous one, and only windows which pass all stages are considered a true detection. Each stage is a so-called strong classifier, which is a linear combination of several weak classifiers, learned using the AdaBoost algorithm. The division in stages means that each stage does not have to be excellent. They must have a very high detection rate each, but a rather high false positive rate it also allowed. The final detection performance is then the product of the detection rates and false positive rates of all stages.

Haar-like features and integral images

The full name of Haar-like features is Haar-wavelet like features, due to their visual resemblance to the mathematical concept Haar wavelets. Examples can be seen in fig. 4.3. They are calculated on grayscale images. In essence they are very simple to calculate: The cumulative intensity of the pixels in the black area(s) is subtracted from the cumulative intensity of pixels in the white area. The features can have different aspect ratios from the ones shown, and can also be freely scaled.

The Haar-features have two major advantages: They can be calculated very quickly using integral images, and they can be easily scaled to detect objects at other scales than in the training images.

Integral images are images where each pixel contain the sum of all previous pixels, that is, all pixels above and to the left of the current. So a pixel at position $(0, 0)$ in an integral image only holds the value of pixel $(0, 0)$ in the original image, but the integral pixel at $(1, 1)$ contains the cumulative value of pixels $(0, 0)$, $(1, 0)$, $(0, 1)$, and $(1, 1)$ in

the original. Mathematically, that can be expressed as:

$$ii(x,y) = \sum_{x'=0}^{x' \leq x} \sum_{y'=0}^{y' \leq y} i(x',y') \tag{4.1}$$

where $ii(x,y)$ are pixels in the integral image and $i(x,y)$ are pixels in the original image. Remember that Haar-like features are built on the accumulated intensities of certain areas, and the usefulness of integral images becomes apparent. The sum of an area defined by $(0,0)$ and (x,y) is simply $ii(x,y)$. But the sum of an arbitrary area is can also be computed with only four queries to the integral image. Consider fig. 4.4. The goal is to find the cumulative value of the marked area, d in the original image. a, b, c, and d are areas in the original image, while i, j, k, and k are points in the corresponding integral image. The value at point i equals the area of a and similarly $j = a + b$, $k = a + c$, and $l = a + b + c + d$. d can be expressed in terms of the four integral points as $d = l - k - j + i$ because:

$$\begin{aligned} l - k - j + i &= (a + b + c + d) - (a + c) - (a + b) + a \\ &= a + b + c + d - a - c - a - b + a \\ &= d \end{aligned} \tag{4.2}$$

When one rectangle can be computed with 4 queries, the difference between two can be done with 8. But since the simplest Haar-like features share two corners between the rectangles, they can be computed with only 6 references. The middle-stripe features can be calculated in 8 references. So a large number of Haar-like features can be computed extremely fast by creating the integral image once and referring to that for each feature calculation.

Not only can they be computed quickly, they can also be easily resized when searching for objects larger than the initial feature size. So instead of having to run the expensive operation of resizing the target image in a resolution pyramid, the features are resized and evaluated once more on the same integral image, a much quicker operation. This insight means that even for a scale invariant detector, only one integral image must be calculated, and no resizing is necessary.

Learning stages with AdaBoost

The Haar-like features are very simple and only become powerful when combined. A single search-window of 24x24 pixels contains 45,396 different Haar-like features if all original features suggested by Viola and Jones (2001) are calculated in all sizes, so it is not feasible to calculate them all. Instead, the AdaBoost algorithm is used to select the best features (called weak learners) and combine them into a more powerful strong learner, which is a weighted sum of weak learners. This is the training phase of the algorithm. For training, it is fed a set of positive images, known to contain signs and a negative set known *not* to contain any signs.

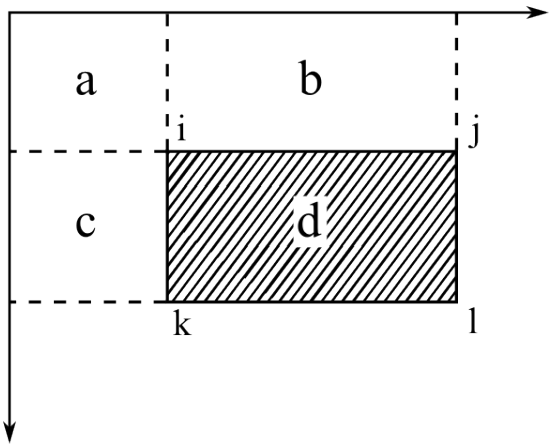


Figure 4.4: Using an integral image to calculate the sum under an arbitrary rectangle.

AdaBoost work by focusing on the training samples that are hard to classify. First step is to take a single feature and create a weak classifier, $h_j(x)$:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \Theta_j \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where p_j is a parity, which can switch the direction of the inequality, $f_j(x)$ is the output of the feature f_j given the window x, and Θ_j is a threshold. In words: It defines a detection as true if the feature is above a certain threshold. The threshold is set by running the feature on all training images and selecting the threshold that best separates the training set.

This procedure is run for all possible features on all training images. The final strong classifier is a combination of the features with the best classification performance, weighted after performance. Each training image also has a weight attached and during the training, the training images which are hard to classify has their weight increased. Weak classifiers which are good at classifying hard training images are given a higher weight and are more likely to be included in the final strong classifier. Not all features are included in the strong classifier, only enough to achieve the desired stage detection performance, and as long as a weak classifier which have a detection rate better than chance (50%) can be found, it can be added to the strong classifier and improve its performance.

The algorithm described here it the one used by Viola and Jones (2001) and is a slightly modified AdaBoost algorithm. More information on AdaBoost and boosting in general can be found in Duda et al. (2001).

The full detection structure

The full detection structure is a cascade of stages trained with AdaBoost. A diagram of that can be seen on fig. 4.5. Each stage is a strong classifier, and later stages are only trained with the training images that have passed all previous stages, so each stage is

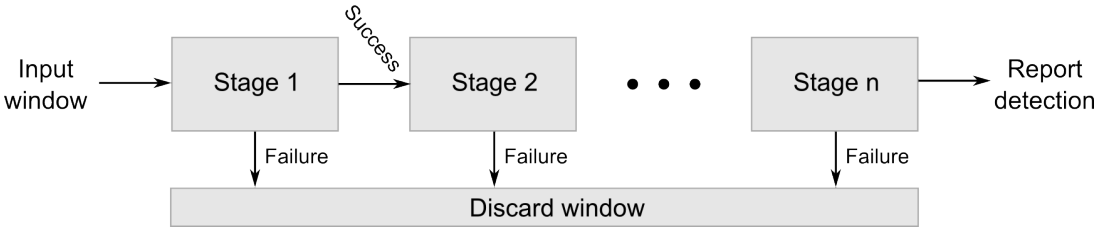


Figure 4.5: Flow of a detection window through a cascaded classifier.

progressively harder. In detection, only windows which pass all stages are considered true. If a window is rejected already at stage 1, only the features involved in that classifier has been calculated. Thus, it is very rarely necessary to compute all features in all stages, a fact that makes the algorithm faster.

As mentioned earlier, each strong classifier only trains until the stage reaches its performance requirements to ensure that they are not consuming any more computation time than necessary. The stage performance requirements can be calculated from the desired total cascade performance requirements if the number of stages is known. For a total detection rate of 95%, each stage in a 20 stage detector must have a detection rate of $\sqrt[20]{0.95} = 0.997$. The false positive rates are compounded in a similar way, so if a maximum false positive rate of $6 \cdot 10^{-}6$ is required, each stage must have a false positive rate of no more than $\sqrt[20]{6 \cdot 10^{-6}} = 0.548$. So with enough stages, even very high false positive rates per stage can be allowed.

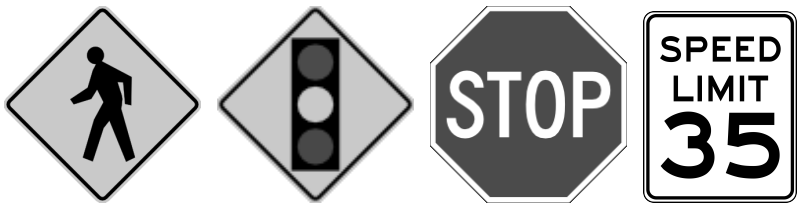
Training and using Haar-cascades in practice

The Haar-cascades has not been implemented from the ground up in this project. Instead the built-in implementation in OpenCV has been used, along with the training program provided with OpenCV. Appendix C contains a guide on how to perform the training using the tools that come with OpenCV.

4.2.3 Evaluating performance

4.3 Results

Haar-cascades were trained for four different sign types: Stop sign, pedestrian crossing, speed limit 35, and signal ahead (fig. 4.6). These were chosen because the cover various shapes and color, and because they are well represented in the LISA dataset. Initially, the detectors were trained with a subset of the LISA dataset, and with a corresponding number of synthetic training images. Generally, Haar-cascades perform better if the training set is larger, and since generating larger synthetic data sets require no more effort, these were also added to the test. Finally, for the classes where more real-world training images could be collected, the real-world detectors were also trained



(a) Pedestrian crossing **(b)** Signal ahead **(c)** Stop **(d)** Speed limit

Figure 4.6: The four signs that were used to train the detector.

with larger data sets. The training images were 20x20 pixels, except for the speed limit signs, which were 18x24 pixels. Subsets of the training images used can be seen in fig. 4.7. The cascades were generally trained with an aim of 20 stages, but in some cases the training terminated early because the required performance on the training set was achieved early.

The detection results can be seen in table 4.1. The tests were carried out on real-world images which were separate from the training set. They were randomly selected from the LISA dataset with the `splitAnnotationFiles.py` tool. The detection performance was evaluated with the `evaluateDetections.py` tool. See section 3.2.4 for more information on both.

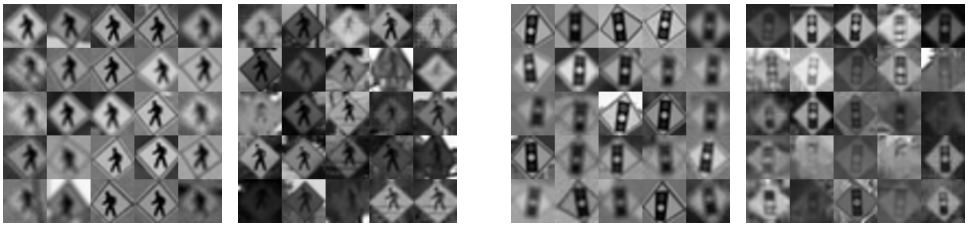
To evaluate how well the synthetic images cover the variance present in the real-world data, two measures were plotted: The average intensity of pixels in each training image and the blur measure of different training sets (fig. 4.8). Each cross represents a single training image. For fig. 4.8a the mean of all pixels the the training images were calculated and plotted. For fig. 4.8b the blur measure, as described by Marziliano et al. (2002), was calculated for each image. The blur measure is the mean of the width of vertical edges in an image:

$$B = \frac{1}{n} \sum_{i=0}^{n} e_i \tag{4.4}$$

where B is the blur-value, n is the number of vertical edges in an image and e_i is the edge width of a specific edge pixel, given as the distance between the pixels with the local maximum and minimum intensity around the edge pixel.

4.4 Discussion

The synthetic data performs consistently worse than the real data. In all cases the real-world detectors have a detection rate above 70%, whereas the synthetic detectors never go above 30%, except in the case of the synthetic stop sign detector which was intentionally only employing 10 stages. That one reaches 58.3% but at a cost of a prohibitively high false positive rate. Providing more training data in the synthetic

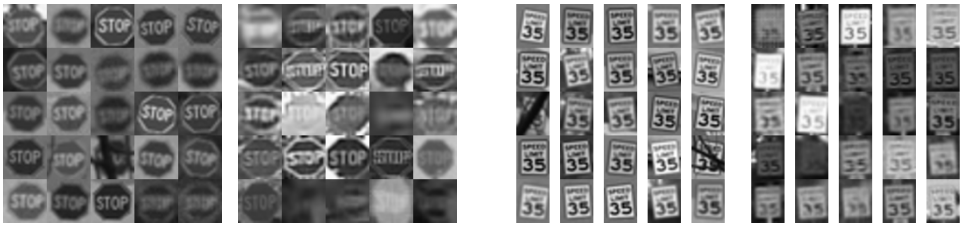


(a) Synthetic training images. Template in fig. 4.6a

(b) Real-world training images.

(c) Synthetic training images. Template in fig. 4.6b

(d) Real-world training images.



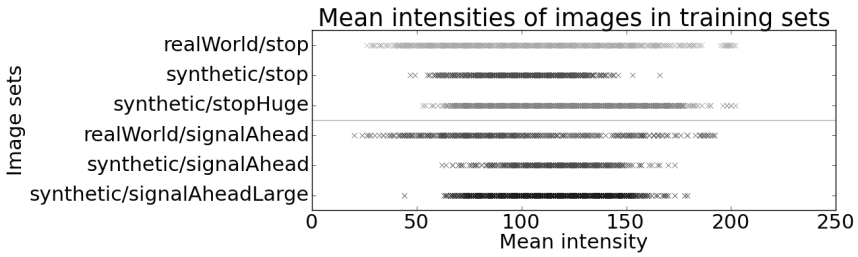
(e) Synthetic training images. Template in fig. 4.6c

(f) Real-world training images.

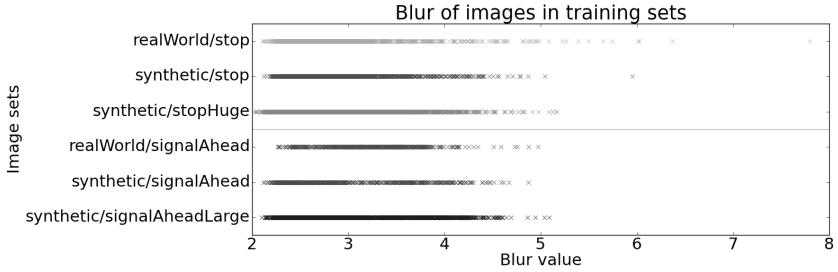
(g) Synthetic training images. Template in fig. 4.6d

(h) Real-world training images.

Figure 4.7: Samples from the training image sets.



(a)



(b)

Figure 4.8: Distribution of two parameters in the training sets.

Table 4.1: Results of the comparative evaluations of detectors

Training type	Training images (positive/negative)	Stages	Signs to find	TP	FP	FN
Stop						
Real-world	1218/2500	20	103	76 (73.8%)	11	27
Real-world	1686/3000	20	103	75 (72.8%)	8	28
Synthetic	1218/2500	17	103	18 (17.5%)	2	85
Synthetic	5000/10000	19	103	26 (25.2%)	5	77
Synthetic	1218/2500	10	103	60 (58.3%)	1500	43
Pedestrian crossing						
Real-world	364/800	20	40	29 (72.5%)	10	11
Real-world	1044/2000	20	40	30 (75%)	2	10
Synthetic	364/800	14	40	11 (27.5%)	28	29
Speed limit 35						
Real-world	253/500	20	21	15 (71.4%)	1	6
Synthetic	253/500	7	21	5 (23.8%)	32	16
Synthetic	2000/4000	7	21	6 (28.6%)	6	15
Signal ahead						
Real-world	597/1500	20	56	42 (75%)	10	14
Real-world	859/2000	20	56	38 (67.9%)	4	18
Synthetic	597/1500	13	56	14 (25%)	117	49
Synthetic	2000/4000	13	56	16 (28.6%)	53	48

case does help, but even a large increase (more than a doubling) of the training data does not make the synthetic data perform comparably to the real-word data, as seen in the case of the stop sign detector.

The reason for this discrepancy must be that the synthetic training data does not encapsulate the variance found in the real world. This is evaluated in fig. 4.8. While the intensity variance is not always fully covered, the sets *realWorld/stop* and *synthetic/stopHuge* are very similar, and even in that case the difference in detection rate is enormous. The blur is generally captures well by the synthetic data. This analysis only covers two of the ten parameters used to generate the synthetic data - the others are hard to measure and plot - so it is possible that the lack of variance stems from one of the remaining 8 parameters, but also that it is caused by a lack of parameters. Unfortunately there is no straight forward way of determining which parameters are necessary to properly emulate the real world in this context.

The experiments performed here shows that it is hard - if not impossible - to create synthetic data of a quality that is useful to train a detector. A visual inspection seems to show that the real world is emulated well, but detector test shows otherwise.

5

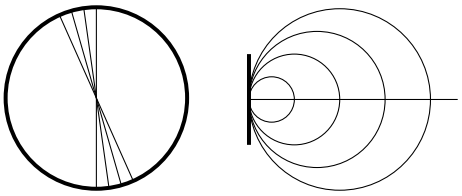
Model based sign detection

This chapter is not based on a paper, but instead presents the work done upon arrival at UCSD. The system is intended to be used as a building block in the lab's continuous work with intelligent cars and in future full sign detection systems. It is not meant to act as a stand-alone sign detection system. It investigates a purely model based detection approach, as opposed to the machine learning based approaches presented earlier in this report.

5.1 Introduction

So far this report has mainly been concerned with machine learning based detection approaches using training sets. But as mentioned in chapter 2, model based approaches are also common. This chapter covers the most common model based approach, the Fast Radial Symmetry (FRS) detector which was first presented by Loy and Zelinsky (2003). FRS is the most common shape-based detector in traffic sign detection and is used stand-alone or in combination with color thresholding. For this project it has been implemented as a stand-alone detector and is essentially a shape detector. In the present version there is no mechanism to remove detections of shapes which fit the model of a sign shape, but which are not signs. It was chosen for several reasons:

- Earlier experiments had shown that color based detection methods were quite unreliable with changing lighting conditions, so it was desired to test a shape based detector.
- There was no guarantee that input data would be color video, which also counted towards a purely shape-based approach.
- US speed limit signs are monochrome with a white background, making them hard to detect with color based approaches (Gomez-Moreno et al. (2010) have shown that achromatic decomposition can be used, albeit with mixed results).
- FRS is the most common sign detection algorithm of all.



(a) Intersecting vote lines

(b) Possible circles for a gradient

Figure 5.1: The basic principle behind the radial symmetry detector. Image inspired by Barnes et al. (2008).

- It is often part of larger systems, so it could likely be used as a building block if a larger system was to be developed.

Furthermore, an extended version, EFRS, was developed and implemented which is able to find rectangular shapes. This was necessary in order to be able to find the rectangular US speed limit signs. Regular FRS is only able to find circles and regular polygons.

This chapter presents the FRS and EFRS algorithms, the implementation that was created, and a test of its detection abilities.

5.2 Methods

This section covers how the EFRS algorithm works and describes the implementation made for this project.

5.2.1 Radial Symmetry Voting

FRS is similar to the Hough transform, but it works directly on the edges of an image instead of in the Hough space. This section will begin by explaining the simplest incarnation of FRS which is able to find circles. Then it is expanded to regular polygons and finally to rectangles.

FRS is a voting based detector. It evaluates pixels one at a times and casts a vote for the center of the shape the current pixel would be a part of. Positions which get many votes are likely to be a center of such a shape, whereas positions with few votes will have gotten them from pixels which are not part of a shape, so they can safely be ignored.

The principle for the simplest case of FRS can be seen on fig. 5.1. The gradients of the edge of a circle are all intersecting in one place: The circle center. This can be seen in fig. 5.1a. A single gradient can be associated with any number of circles - or none at all - as shown in fig. 5.1b. The horizontal line depicts the gradient of the edge shown

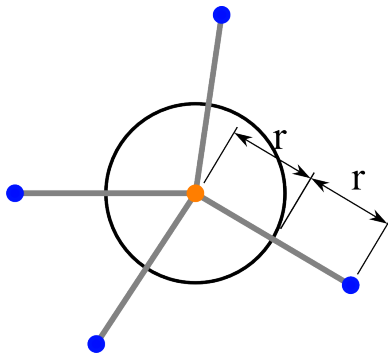


Figure 5.2: Four example gradients when searching for a circle. These gradients result in 8 votes, 4 in the same place, which is the center of the circle.

by the vertical line. The edge could be part of any of the circles show. So to find a circle of a radius r, a vote is cast in the distance r from all edge pixels in the direction of the gradient. A vote is cast to both sides of the edge, because it cannot be known a priori if the circle is dark on a light background or light on a dark background, so the direction of the gradient with respect to the circle is unknown. The voting can be expressed mathematically like this:

$$\mathbf{p}_v = \mathbf{p} \pm \text{round}(r\mathbf{g}(\mathbf{p})) \tag{5.1}$$

where $\mathbf{p}_v$ is the point to cast a vote on, $\mathbf{p}$ is the edge point, r is the radius of the circle, and $\mathbf{g}(\mathbf{p})$ is the unit gradient of edge point $\mathbf{p}$.

An example of the voting is shown in fig. 5.2. For clarity only 4 example votes are shown, but in reality there would be votes from each pixel along the edge of the circle. The gradients (gray lines) exist along the circle edge perpendicular to the edge. Each of the blue points outside the circle receive one vote as they are on the gradient in the distance r from the source pixel. But the center of the circle receives one vote from each gradient, for a total of four votes in this example. To find the center, it is enough to just find the point with the most votes.

In an actual application, the gradients must first be found, and that can be done using the Canny edge detector (Canny, 1986) or by convoluting the image with a Sobel kernel the x and y direction and combining them. That is essentially Canny's algorithm, but without non-maximum suppression. Once the gradients are found, all gradients with a magnitude below some threshold are discarded. This reduces the computational load of the algorithm and is allowed because shapes are expected to differ significantly from their background. Then it is simply assumed the each non-zero gradient is part of a circle and a vote is cast accordingly, as described above. The votes are kept track of on a separate vote-image of the same dimensions as the input image. Because votes are cast for a specific circle size, the algorithm must be run in several passes if it should look for circles of different sizes. Then either all points with a sufficient number of votes, or a fixed number of points with the most votes are selected a circle centers.

When the method is extended to regular polygons, votes are still cast at the distance

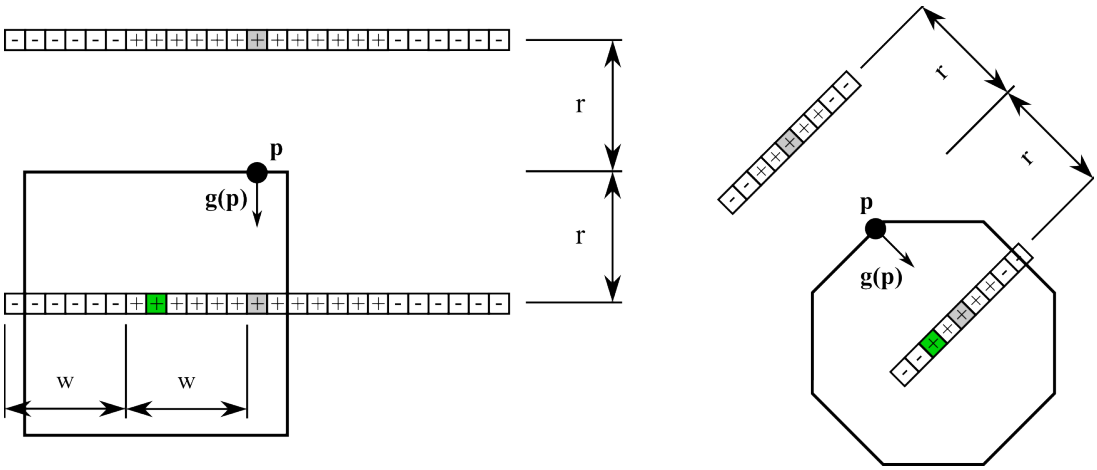


Figure 5.3: Votes cast when searching for regular polygons. + indicate a positive vote and − indicate a negative vote. $\mathbf{p}$ is the edge point, r is apothem of the shape, and $\mathbf{g}(\mathbf{p})$ is the unit gradient of edge point $\mathbf{p}$

.

r from the edge point. Now r is the apothem, the perpendicular distance from the edge to the center. The center of the shape is no longer necessarily at the gradient, so a line of votes is cast to make sure the center is hit by a vote. To compensate for the extra votes, negative votes are introduced at the ends of the vote line. An example is in fig. 5.3. As p is moved, the voting line is moved along with it. Even in the extreme cases at the ends of an edge, as shown here, a vote will hit the center of the shape, highlighted in green. The negative votes make sure that votes outside the center are neutralised as the vote line is moved along. w defines how long the vote lines are, and is calculated as:

$$w = \text{round}(r \tan\left(\frac{\pi}{n}\right)) \tag{5.2}$$

where n is the number of sides of the targeted polygon. The voting line is defined as:

$$L(\mathbf{p}, \mathbf{m}) = \mathbf{p}_v(\mathbf{p}) + \text{round}(r\bar{\mathbf{g}}(\mathbf{p})) \tag{5.3}$$

where $\bar{\mathbf{g}}(\mathbf{p})$ is a unit vector orthogonal to $\mathbf{g}(\mathbf{p})$ and $\mathbf{m}$ is a set of distances from the center of the voting line, $\mathbf{p}_v(\mathbf{p})$. Positive votes are given when:

$$L(\mathbf{p}, \mathbf{m}) \,|\, m \in [-w, w] \tag{5.4}$$

and negative votes when:

$$L(\mathbf{p}, \mathbf{m}) \,|\, m \in [-2w, w-1] \cup [w+1, 2w] \tag{5.5}$$

For sign detection, the assumption is made that the signs are in an upright position, and as such, the possible edge angles can be narrowed down. A square has two possible angles: 0° and 90°, so any gradients outside those angles (plus/minus some tolerance) are not processed. This speeds up the algorithm. Fig. 5.3 shows examples for two shapes, a square and an octagon, but the method can be used with any regular polygon.

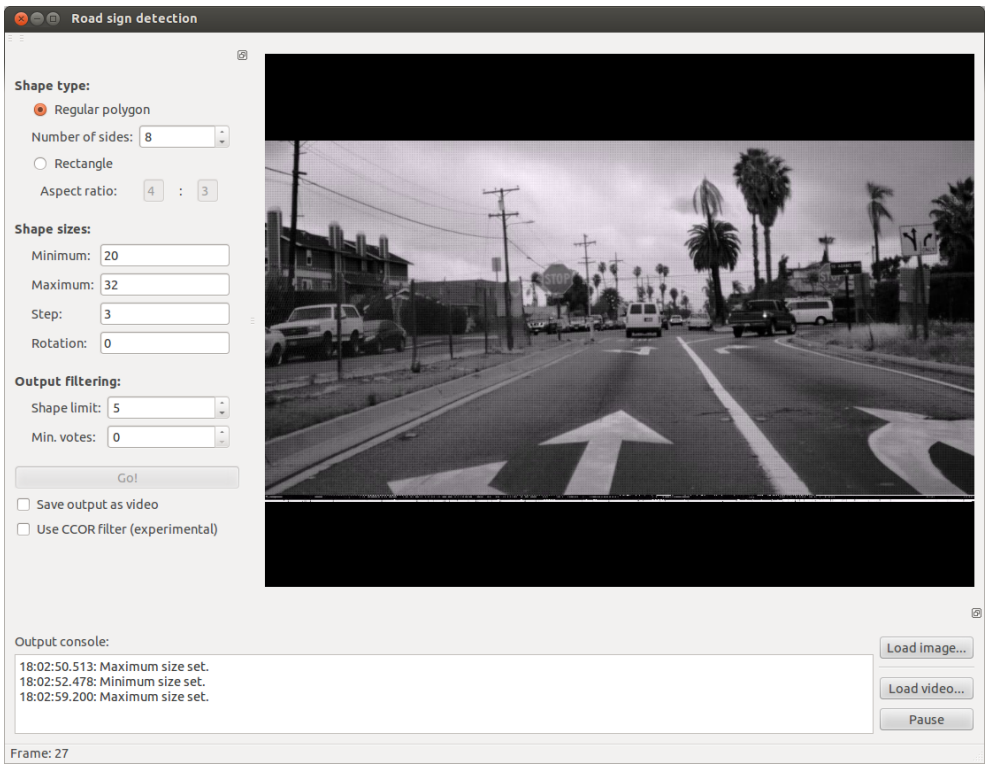


Figure 5.4: Screenshot of the model based detector

Th final extension is that to rectangles. It is the simple act of having different values of r and thus also w depending on the gradient angle. This ensures that the center of the shape is hit by a positive vote also on the sides that are longer.

5.2.2 Implementation

As almost all programs in this project, the implementation was made in C++ with OpenCV, Qt, and OpenGL. A screenshot is in fig. 5.4. It is possible to set a variety of parameters with regards to the shape type and size and it can be set up to find a certain number of shapes in each image (say, the 5 best rectangles), to simply find all shapes with a certain number of sides, or a combination of the two. The program has a very simple tracking algorithm built in, which requires detected shapes to be present in two consecutive frames before it is accepted as a true detection. The tracking simply checks if a similar shape was present at roughly the same coordinates (+/- 5 pixels) in the previous frame.

5.3 Results

The program has been tested on 3 different sign shapes: Octagonal stop signs, diamond shaped signal ahead signs, and rectangular speed limit signs. Because of the tracking, the program cannot be tested on single frames. Instead, 5 tracks for each sign type

Table 5.1: Test results for the model based EFRS detector.

	True positives	False positives	Misses	Frames	Detection rate	FP per frame	Per sign detection rate
Speed limit signs	86	717	319	354	21.23%	2.02	100%
Signal ahead signs	198	315	560	534	26.12%	0.59	60%
Stop signs	67	795	426	537	13.59%	1.41	100%

were processed and the detections counted, both as per-frame and per-sign statistics. Results can be seen in table 5.1.

No attempt was made to optimize the detection parameters, they were simply chosen from what seems reasonable, an a few of the misses are caused by the sign being too large. It is likely that slightly better results could be obtained if the setting were tweaked carefully. Also, a cap was set, so the detector only presented the 5 best shapes in each frame (if more than 5 shapes passed the two-frame tracking requirement). This makes it possible, though unlikely, that the signs were found but as a less strong shape than others in the frame.

5.4 Discussion

The detector has a large number of false positives and cannot work on its own, but it was not intended for that. It is important to remember that this is a general shape detector so some of the false detections may actually be true positives in the sense that they are actually such a shape, just not a sign. A qualitative evaluation of performance, however, shows that is rarely the case. More often the detector is fooled by branches and tree trunks which form shapes against the sky.

The per-frame detection rates are not impressive, but in most cases the sign is at least found at some point in the track. Judging from the per-frame detection rates, the stop sign is the hardest one to find, something that is not much of a surprise, since it is the most complex shape. However, many of the stop sign tracks are of very bad quality, which might explain the poor performance. The best performance is achieved for signal ahead signs, possibly because the combination of diagonal lines is not something found elsewhere in the environment.

A qualitative evaluation of the detection results show that the detector performs much better on sharp, non-shaken pictures, which makes sense as the detector works by looking at sharp gradients formed by strong contrast. As with any detection method, the better the source material is, the better is works.

6

Pedestrian detection

This chapter covers the work presented in *A Two-stage Part-Based Pedestrian Detection System Using Monocular Vision*, which was submitted to IEEE Intelligent Transportation Systems Conference (ITSC), 2012. The work was carried out in close collaboration with Antonio Prioletti of University of Parma, who was responsible for carrying out all implementation and testing.

6.1 Introduction

Pedestrian detection is currently a very large research field. It can be used in surveillance, Advanced Driver Assistance Systems (ADAS), and many other places. The ADAS scenario offers plenty of challenges (as summarized in Geronimo et al. (2010)): High variability in appearance among pedestrians, cluttered backgrounds, highly dynamic scenes with both pedestrian and camera motion, and strict requirements in both speed and reliability. Input from a reliable pedestrian detection system can be used to warn the driver about people in front of the car, prepare or even activate a braking maneuver to prevent a collision, or deploy other safety systems such a airbags.

ADAS is a a challenging domain to work within. Braking systems take a short while to apply, and reaction times must be fast for driving, where fractions of a second can be the deciding factor between a collision and a near-miss. At the same time, the system must be robust, so the braking system is not deployed mistakenly (due to a false positive detection), which could itself lead to accidents, or worse, not employ at all (due to a missed detection). Further reasoning than just detection is necessary in such a framework, with pedestrian intent estimation being a good example, as presented by Gandhi and Trivedi (2008) or as another example, automatic breaking as done by Broggi et al. (2009).

The algorithm introduced here is a part-based 2-stage detection method, an extension what Geismann and Schneider (2008) suggested. It combines the speed of a Haar-based boosted cascade with the low number of false positives from the HOG-SVM detector, bringing it closer to the strict ADAS requirements than any of the two algo-

rithm on their own. Here it is extended to a part-based solution, which lowers the false positive rate even further.

6.2 Methods

As mentioned, pedestrian detection is a field with much attention from the research community. Even when narrowed to applications in connection with cars and ADAS, a large body of work exist. A classic method of pedestrian detection is a boosted cascade on Haar-like features, first presented by Viola and Jones Viola and Jones (2001). It is very fast, but lacks robustness due to the high appearance variability among pedestrians in the real world. Instead, many people turn to the HOG-SVM solution presented by Dalal and Triggs Dalal and Triggs (2005). It is much more robust and generally detect pedestrians in harder situations, while keeping a low number of false positives. Its problem lies in processing speed. As mentioned, the ADAS application requires fast processing, something that is not immediately obtainable with the HOG-SVM detector. The HOG-SVM method was explored for use with infrared images in Suard et al. (2006). For further exploration of pedestrian detectors, we refer the reader to the general survey by Geronimo et al. (2010) or, for vision-only based systems, Gandhi and Trivedi (2006, 2007). The system presented in this chapter uses monocular vision as base for the detection. This means that the hardware requirements for the car are low and realistically possible - many cars are already outfitted with a front facing camera for other purposes, such as lane detection. For a survey of monocular vision based methods, see Enzweiler and Gavrila (2009).

This method combines the speed of the Haar detector with the robustness of a part-based HOG-SVM detector. The base for the method used in this paper was first presented by Geismann and Schneider (2008), but is also covered by Yuan et al. (2011); Yongzhi et al. (2010) in various versions. Apart from using a combination of a Haar-cascade and HOG-SVM, Geismann and Schneider also evaluated using a sparse HOG descriptor to speed up the verification. Part-based pedestrian detection has been presented in various versions before, such as Mao et al. (2007); Wu and Nevatia (2005, 2007).

The properties of the Haar cascade and the HOG-SVM detector makes them prime candidates for combination: The Haar cascade does the initial pass, finding Regions Of Interest (ROIs) that are passed on to the HOG-SVM detector which verifies the initial findings by the Haar cascade. The first stage is called the *detection stage* and the second the *verification stage*. That is the basics of the approach outlined in Geismann and Schneider (2008).

The goal is to lower the number of false positives without too much penalty in the detection rate. In order to do this, the verification stage is altered to not only verify based on a full body classification, but also a lower body and upper body classifier. We combine these results to figure out whether the ROI contains a person.

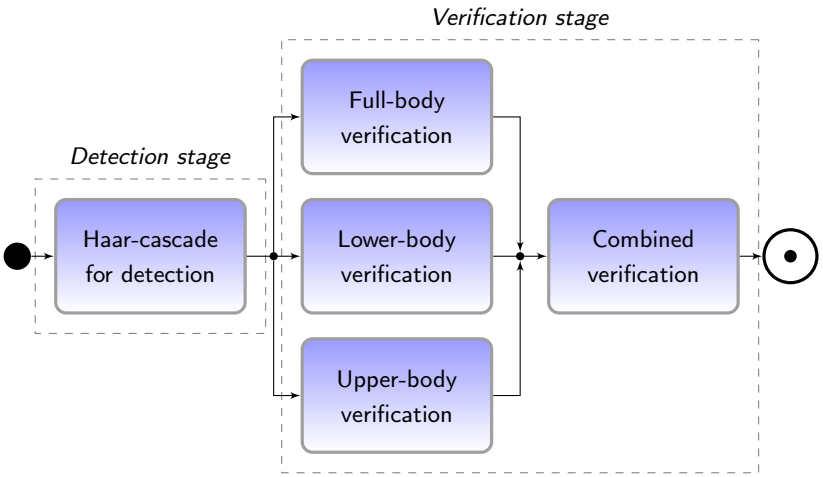


Figure 6.1: The flow of the algorithm described in this paper.

The combination of verification results is done in two ways, which are compared: A simple majority vote, requiring at least two of three classifiers to verify the detection, and a more advanced way, which introduces a third stage to the algorithm, classifying each window based on the estimated function value from an SVM regression performed on each part.

An overview of the flow through the algorithm can be seen in fig. 6.1.

6.2.1 Detection stage

The detection stage is an AdaBoost cascade on Haar-features (Viola and Jones, 2001). It has already been described in detail in chapter 4, so this chapter does not go into details with how the methods works.

Throughout the chapter, the INRIA Pedestrian Dataset Dalal and Triggs (2005) has been used. Thus, the detection cascade was trained with the training set given therein: 2416 positive images and 12180 negative images. The training images were cropped closely around the annotated persons, because Haar-cascades do not benefit from having as much background included as HOG-based classifier. After the crop the training images were resized to 12x28 pixels.

The detection stage is set up so that it finds the maximum possible number of pedestrians, which also means that it will return plenty of false positives. A larger number of false positives will slow down the computation, since the verification stage must process more, but it is a worthy trade-off given that the true positive rate of this stage forms the upper bound of detections for the entire system.

The detection stage returns bounding boxes of all the potential pedestrians in the picture, which are sent on to the verification stage. Part based detection in the detection stage is not used, since the data from Alonso et al. (2007) shows that the Haar cascade generally performs bad in part-based detection schemes. An example of the output of

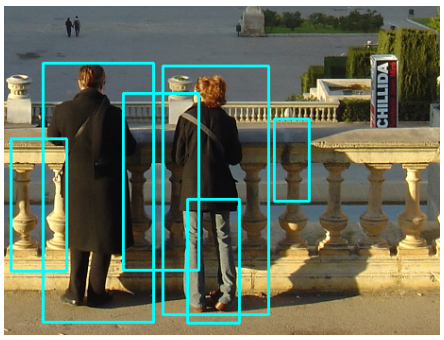


Figure 6.2: Example of the output from the detection stage. It is clear that it contains several false positives, but that is desired, since it ensures that also the true positives are included.

the detection stage can be seen on fig. 6.2.

6.2.2 Histogram of Oriented Gradients

Histograms of Oriented Gradients (HOG) is a descriptor for an image patch. The classic way of using them for person detection was presented by Dalal and Triggs (2005) and involves computing the HOG feature in a sliding window across the target image, sending each descriptor into an SVM, which will determine if the current window contains a detection or not. That is also the method used in this work. A few variants of the HOG descriptor exist, but the most common in the R-HOG (R for rectangular), which is the one described here. The elements of HOG is illustrated in fig. 6.3. At the highest level is the source image over which the sliding window is passed. In the original implementation the detection window i 128x64 [px]. The descriptor is divided into a number of overlapping blocks. The overlap is defined by the block stride parameter, which defines the spacing with which the blocks are calculated. Experiments indicate that a large overlap is beneficial for detection performance. The part-based verification stage used in this work differs from the full-body verification stage of Geismann and Schneider's Geismann and Schneider (2008). We use a part-based detection scheme. The verification stage consists of two sub-stages: The individual part verification and the combined verification. Three SVM regressions based on dense HOG descriptors are calculated and applied to the ROIs given by the detection stage. One is for full body classification, one is for lower body classification, and one is for upper body classifications.

Our algorithm uses classic dense HOG descriptors (as opposed to the sparse descriptors used in Geismann and Schneider (2008)). They are calculated using integral images in an effort to speed up the process, as described in Porikli (2005). Since HOG works best if some amount of background is introduced to the detection window, the ROIs are resized appropriately from the tight boxes that are returned by the detection stage. Then the content of the ROIs is scaled so it matches the size the SVMs were trained with. At this point the HOG is calculated and passed on to the SVMs.

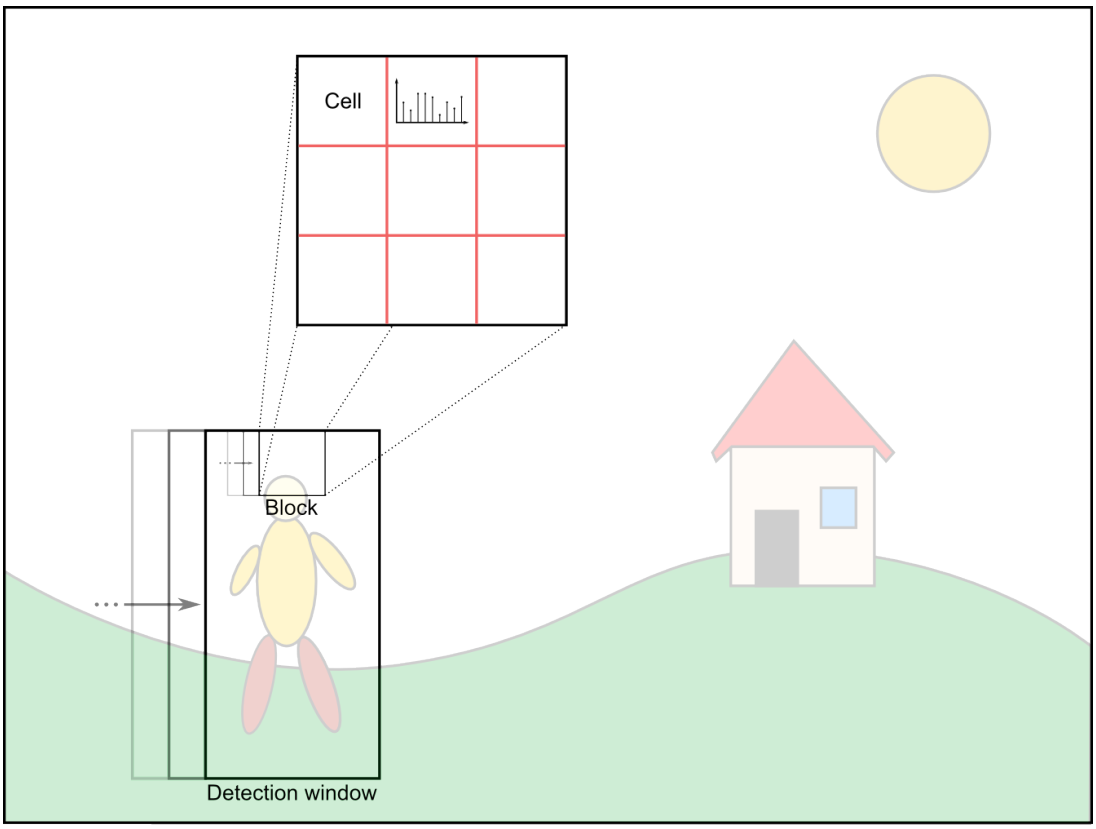


Figure 6.3: Illustration of the elements of a HOG-computation. One HOG descriptor is calculated for each position of the sliding window.

As in the detection stage, each SVM is trained with the INRIA training set. The full body SVM was trained with the full training images, whereas the lower- and upper-body SVMs were trained with the lower and upper half of the training images, respectively. In our system, there is no overlap between the lower and upper body. The parts of training images used for each type are shown in fig. 6.5. So in total, three SVMs were used.

According to Dalal and Triggs (2005), the best performance for human detection is achieved with a grid of 3x3 cells with a size of 6x6 px, however, a cell size of 8x8 px matches better with a window width of 64 px and a block stride of 4 px and does not decrease performance significantly. This makes a block 24x24 px. There is no requirement the the block stride is aligned with cell size, since the cells are calculated relatively to the block's position, but the condition [window size]%[block width in pixels] == 0&&[window size]%[block height in pixels] == 0 must be true to be able to calculate the descriptor for the entire window.

For each cell, a histogram of oriented gradients is calculated. The gradients are calculated using 1D masks [-1, 0, 1] in both the horizontal and vertical direction. Then the gradient angles are divided into 9 bins spanning 0-180° evenly. Because the "sign" of the gradient is unimportant, 0-180° span all possible gradient directions. Each gradient votes in its bin with an amount corresponding to its magnitude, so strong gradient have

more weight, and furthermore, each gradient's vote is bilinearly interpolated between the two closest bins. So given blocks with a size of 3x3 cells, each block contains 9 histograms of 9 bins. These histograms are concatenated, so the descriptor of a single block is a vector of 81 values. Finally the values are normalized. The final HOG descriptor is a concatenation of all the block vectors in the window.

6.2.3 Support Vector Machines

In this pedestrian detection a Support Vector Machine (SVM) is used in the verification stage. SVMs classify samples based on a maximum margin decision boundary. The principle is shown in fig. 6.4. For simplicity, this is the linearly separable 2d case, but most often SVMs are used one datasets with a much greater dimensionality - the HOG features, for example, are thousands of dimensions, depending on settings.

The basic idea is simple. In this example two classes must be separated by a line, a task that can be accomplished with an infinite number of lines. In the figure, two possible lines are shown: The "correct" decision boundary and one example of a non-optimal boundary. The best boundary is chosen as the one which maximizes the margins between the training data and the boundary. Classification of new samples is then done by determining which side of the decision boundary the new sample is on. In the case of a dataset with more than two dimensions, the decision boundary is a $p-1$ dimensional hyperplane for a data dimensionality of p.

The true value of SVMs, however, lies is their ability to be used in data that is not directly linearly separable. Even though the decision boundary is a linear hyper-plane, SVMs transform the input data up to a higher dimensional space where the data is separable. This is done with a so-called SVM kernel and it can be chosen from among several options depending on the actual problem. Normally it will transform the data non-linearly to achieve linear separation.

6.2.4 Verification stage

The part-based verification stage used in this work differs from the full-body verification stage of Geismann and Schneider's Geismann and Schneider (2008). We use a part-based detection scheme. The verification stage consists of two sub-stages: The individual part verification and the combined verification. Three SVM regressions based on dense HOG descriptors are calculated and applied to the ROIs given by the detection stage. One is for full body classification, one is for lower body classification, and one is for upper body classifications.

Our algorithm uses classic dense HOG descriptors (as opposed to the sparse descriptors used in Geismann and Schneider (2008)). They are calculated using integral images in an effort to speed up the process, as described in Porikli (2005). Since HOG works best if some amount of background is introduced to the detection window, the

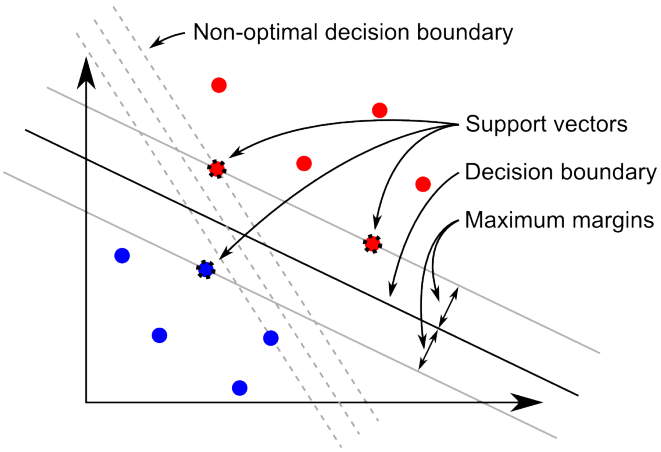


Figure 6.4: A sketch of the output from a Support Vector Machine in the 2 dimensional case. Two classes are linearly separated and the optimal separation plane (or, in this case, line) that is found, is the one with the longest possible margins to the classes. The samples which are closest to the decision plane are called support vectors.

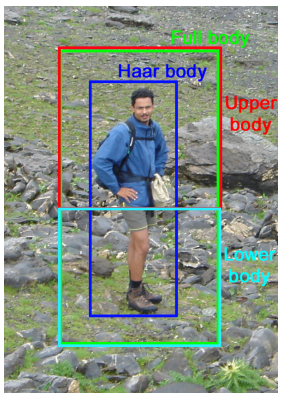


Figure 6.5: The four types of training images used in this system: The three parts for the verification stage, and a closer crop for the detection stage.

ROIs are resized appropriately from the tight boxes that are returned by the detection stage. Then the content of the ROIs is scaled so it matches the size the SVMs were trained with. At this point the HOG is calculated and passed on to the SVMs.

Each SVM is trained with the INRIA training set. The full body SVM was trained with the full training images, whereas the lower- and upper-body SVMs were trained with the lower and upper half of the training images, respectively. In our system, there is no overlap between the lower and upper body. The parts of training images used for each type are shown in fig. 6.5. So in total, three SVMs were used.

To do the combined verification, two different methods were tested: Majority voting and regression output classification.

For majority voting, a regular SVM for classification was trained. It returns which class (pedestrian vs. non-pedestrian) the current detection window belongs to. If at least two out of three classifiers label the window as a pedestrian, it is described as a

Table 6.1: Overview of the detection rates achieved by Geismann and Schneider Geismann and Schneider (2008) with 0.2 false positive per frame

Video	1	2	3	4	5	Mean
Dense descriptor	52%	70%	91%	61%	55%	65.8%
Sparse descriptor	45%	53%	85%	69%	58%	62%

detection. If a detection is labeled 1 and no detection is labeled -1, the formula used for the majority voting is:

$$l_{out} = \begin{cases} 1 & \text{if} \sum_{i=0}^{i<3} l_i >= 1 \\ -1 & \text{if} \sum_{i=0}^{i<3} l_i < 1 \end{cases} \tag{6.1}$$

where l_{out} is the final decision and l_i is the output from one of the three part-based detectors.

For regression output classification, the three part SVMs were instead trained for regression. The training was performed so the resulting function would ideally return 1 in the case of a detection and -1 when nothing was found. When an unknown window is passed through the output function, it will return a value close to 1 if it is a pedestrian, and a value close to -1 otherwise. The output of these three regressions create their own 3 dimensional feature space. Another SVM has been trained to classify in this space. The output from the three regressions is passed into this second SVM and the output from that classifier is the final label.

6.3 Results and discussion

In order to set various parameters so that the best possible performance is achieved, several experiments have been performed. While the training part of the INRIA dataset was used to train both the detection stage and the verification stage, the test part has been used as base for these experiments. It contains 742 images in total, of which 289 contain one or more persons. In total the test set contains 589 persons that should be detected by a perfect system.

The baseline for the comparison is the performance of the system in a configuration similar to the one by Geismann and Schneider: Two stages, but no part-based verification. The principal results from their paper can be seen in table 6.1. Each of the five results in the table are from a test video they obtained from a driving car. Unfortunately we do not have access to the test videos they used, so our results cannot be compared directly with those. Instead we compare the performance of our own implementation of their algorithm to out part-based algorithm.

One of the most important parameters in the system is the number of stages in the Haar-cascade, in this paper designated k. It is interesting to see what impact the changes in k has on the complete system. In fig. 6.6, an ROC curve is shown for the full system with varying depths in the detection stage. The importance of this parameter

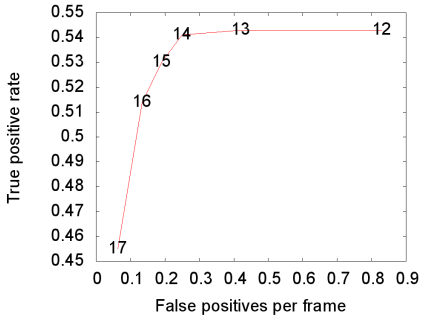


Figure 6.6: Receiver-Operating-Characteristic for the full system with varying k, cascade depths in the detection stage.

Table 6.2: Changes in processing speed for different values of k

Number of images	100
Mean image width	711.88 pixel
Mean image height	818.72 pixel

k	**Mean time per frame**
12	2.5 s
13	1.84 s
14	1.57 s
15	1.46 s
16	1.39 s
17	1.22 s

is evident. As k is lowered, the number of detections rise, but at a large cost in false positives. For the final system, we chose to go with $k = 15$, since it seems to give an acceptable trade-off between true positives and false positives.

The choice of k has an impact on the speed of the system, since more detection windows means slower performance. The full (but non-optimized) system has been run with several numbers of stages and timed, to get a sense for the speed effects it might have. The results are seen in table 6.2

Another important parameter is the padding, p: The amount with which the ROIs returned by the detection stage is enlarged with. The HOG-SVM detector works better if more background is included than what the Haar-cascade uses, so there is no question that the ROIs must be enlarged. Experiments showed that a padding of 3 performed best. Because HOG-SVM is not scale invariant, so when the padding rises, the pedestrian in the ROI becomes a lot smaller, relative to the image, than the pedestrians in the training set. That will alter the output of the detector and some testing was required to make it work properly. The padding value itself is used to calculate the padding in pixels to apply to the ROI. The width in pixels is calculated as:

$$p_{pixels} = \frac{w_{ROI}}{w_t} \cdot p \tag{6.2}$$

where p is the padding value, w_{ROI} is the width of the found ROI, w_t is the width of the training images, and p_{pixels} is the padding measured in pixels. The padding is

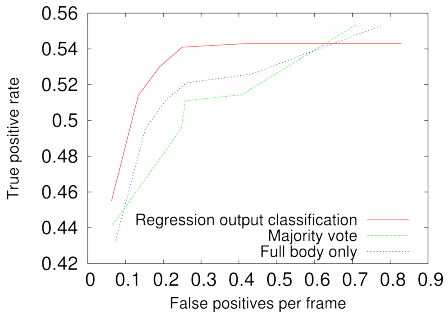
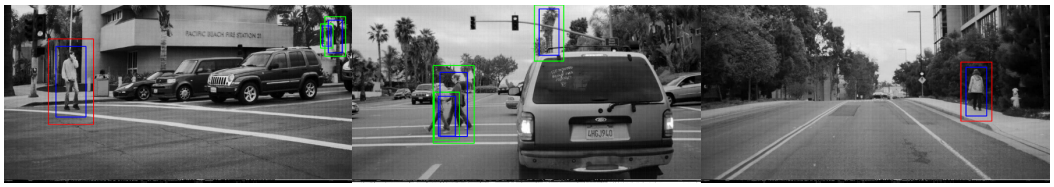


Figure 6.7: Receiver-Operating-Characteristic for the final system. The majority voting approach is not performing better than the full-body approach, but the regression classification approach is consistently better.



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

Figure 6.8: Example outputs of the detector. Input images are images captured using one of LISA's experimental cars. Red boxes indicate a detection, blue are the candidates from the detection stage, and green are the candidates with added padding. In (a) the pedestrian is detected, while a couple of false candidates from the detector are ignored. In (b) the pedestrians are not detected, since the detection stage does not find accurate enough candidate boxes. In (c) the pedestrian is detected and no false windows are found in the detection stage.

applied on all four sides of the image.

After introducing part-based verification to the system, experiments were made to determine whether the simple majority voting or the confidence classification worked the best. These tests were done with the best settings, as determined earlier in this section. The results are shown in fig. 6.7. In absolute numbers, the detection rate is not overwhelming. The important part is the difference between the old two-stage approach with only full-body verification and the new approach. While the voting based approach is not any better then the old full-body verification, the part-based version with regression output classification is better all across the range of false positives per frame.

Examples of detections can be seen in fig. 6.8.

7

Conclusion

This report has detailed the work I have done during my stay in the CVRR lab at UC San Diego. It has been a very interesting stay, especially to work in a research environment as a contrast to the very product oriented approach my previous AAU projects have taken. I believe it is a benefit to have tried both. It has also improved my ability to write scientific papers immensely, something that I had previously only tried at 7th semester.

The main theme in the work has been traffic sign detection, with a brief detour through pedestrian detection, which is related and provided another opportunity to use the methods that was initially investigated with sign detection in mind. The work has resulted in 4 papers which has been submitted for review, one at a journal, and three at conferences.

A large part of the work has been the assembly of a comprehensive and up-to-date survey of traffic sign detection methods in the literature, show a need for research in detection of non-European style signs. It also shows that while a reasonable amount of research has been done, traffic sign detection - and TSR systems in full - is not yet a solved problem.

To facilitate more research in US traffic sign detection specifically, a database of nearly 8000 signs has been assembled. Its structure has been modeled after the most prominent existing databases, but it is extended with full video tracks of signs, something that enables the development of detectors which use tracking.

Synthetic training data based on well-defined traffic sign templates has been investigated for use in machine learning based detectors. Unfortunately the performance has not been as good as hoped, and with the current synthetic generation algorithm, the synthetic data is simply not good enough compared to real-world data.

As a counterpoint to the ML-based approach, a purely model-based detector has also been implemented and tested. The chosen algorithm is the most commonly used in existing systems, and while the test here has used it as a stand-alone detector, most systems combine it with other detection schemes.

Finally, a novel two-stage part-based pedestrian detector has been developed and described. Tests show that it has improved performance from the original work it was based on, and it performs similarly to competing detection methods, but at a faster frame rate.

Bibliography

Agoston, M. K. (2005). *Computer Graphics and Geometric Modeling: Implementation and Algorithms*.

Alefs, B., G. Eschemann, H. Ramoser, and C. Beleznai (2007, june). Road Sign Detection from Edge Orientation Histograms. In *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 993–998.

Alonso, I., D. Llorca, M. Sotelo, L. Bergasa, P. de Toro, J. Nuevo, M. Ocaña, and M. Garrido (2007). Combination of feature extraction methods for SVM pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on 8*(2), 292–307.

Bahlmann, C., Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler (2005). A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pp. 255–260. IEEE.

Barnes, N. and G. Loy (2006). Real-time regular polygonal sign detection. In *Field and Service Robotics*, pp. 55–66. Springer.

Barnes, N. and A. Zelinsky (2004, june). Real-time radial symmetry for speed sign detection. In *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 566–571.

Barnes, N., A. Zelinsky, and L. Fletcher (2008, june). Real-Time Speed Sign Detection Using the Radial Symmetry Detector. *Intelligent Transportation Systems, IEEE Transactions on 9*(2), 322–332.

Baro, X., S. Escalera, J. Vitria, O. Pujol, and P. Radeva (2009, march). Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification. *Intelligent Transportation Systems, IEEE Transactions on 10*(1), 113–126.

Belaroussi, R., P. Foucher, J. Tarel, B. Soheilian, P. Charbonnier, and N. Paparoditis (2010). Road sign detection in images: A case study. In *Proceedings of International Conference on Pattern Recognition (ICPR10), Istanbul, Turkey*.

Belaroussi, R. and J.-P. Tarel (2009, dec.). Angle vertex and bisector geometric model for triangular road sign detection. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pp. 1–7.

Boi, F. and L. Gagliardini (2011). A Support Vector Machines network for traffic sign recognition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 2210–2216. IEEE.

Broggi, A., P. Cerri, S. Ghidoni, P. Grisleri, and H. Jung (2009). A new approach to urban pedestrian detection for automatic braking. *Intelligent Transportation Systems, IEEE Transactions on 10*(4), 594–605.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (6), 679–698.

Chiang, H.-H., Y.-L. Chen, W.-Q. Wang, and T.-T. Lee (2010, dec.). Road speed sign recognition using edge-voting principle and learning vector quantization network. In *Computer Symposium (ICS), 2010 International*, pp. 246–251.

Ciresan, D., U. Meier, J. Masci, and J. Schmidhuber (2011). A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 1918–1921. IEEE.

Creusen, I., R. Wijnhoven, E. Herbschleb, and P. de With (2010, sept.). Color exploitation in hog-based traffic sign detection. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 2669–2672.

Dalal, N. and B. Triggs (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Volume 1, pp. 886–893. Ieee.

De la Escalera, A., J. Armingol, and M. Mata (2003). Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing 21*(3), 247–258.

Deguchi, D., M. Shirasuna, K. Doman, I. Ide, and H. Murase (2011). Intelligent traffic sign detector: Adaptive learning based on online gathering of training samples. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 72–77. IEEE.

Doshi, A., S. Cheng, and M. Trivedi (2009). A novel active heads-up display for driver assistance. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39*(1), 85–93.

Doshi, A. and M. Trivedi (2007). Satellite imagery based adaptive background models and shadow suppression. *Signal, Image and Video Processing 1*(2), 119–132.

Doshi, A. and M. Trivedi (2010). Attention estimation by simultaneous observation of viewer and view. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 21–27. IEEE.

Duda, R., P. Hart, and D. Stork (2001). *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification. Wiley.

Enzweiler, M. and D. Gavrila (2009). Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*(12), 2179–2195.

Fleyeh, H. and M. Dougherty (2005). Road and traffic sign detection and recognition. In *10th EWGT Meeting and 16th Mini-EURO Conference*, pp. 644–653.

Fu, M.-Y. and Y.-S. Huang (2010, july). A survey of traffic sign recognition. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, pp. 119–124.

Gandhi, T. and M. Trivedi (2006). Pedestrian collision avoidance systems: A survey of computer vision based recent studies. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pp. 976–981. IEEE.

Gandhi, T. and M. Trivedi (2007). Pedestrian protection systems: Issues, survey, and challenges. *Intelligent Transportation Systems, IEEE Transactions on 8*(3), 413–430.

Gandhi, T. and M. Trivedi (2008). Image based estimation of pedestrian orientation for improving path prediction. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 506–511. IEEE.

Gao, X., K. Hong, P. Passmore, L. Podladchikova, and D. Shaposhnikov (2008, January). Colour vision model-based approach for segmentation of traffic signs. *J. Image Video Process. 2008*, 6:1–6:7.

Gao, X., L. Podladchikova, D. Shaposhnikov, K. Hong, and N. Shevtsova (2006). Recognition of traffic signs based on their colour and shape features extracted using human vision models. *Journal of Visual Communication and Image Representation 17*(4), 675–685.

Garcia-Garrido, M. A., M. Ocana, D. F. Llorca, M. A. Sotelo, E. Arroyo, and A. Llamazares (2011, oct.). Robust traffic signs detection by means of vision and V2I communications. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1003–1008.

Geismann, P. and G. Schneider (2008). A two-staged approach to vision-based pedestrian recognition using Haar and HOG features. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 554–559. IEEE.

Geronimo, D., A. Lopez, A. Sappa, and T. Graf (2010). Survey of pedestrian detection for advanced driver assistance systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*(7), 1239–1258.

Gil Jiménez, P., S. Bascón, H. Moreno, S. Arroyo, and F. Ferreras (2008). Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies. *Signal Processing 88*(12), 2943–2955.

Gomez-Moreno, H., S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo (2010, dec.). Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition. *Intelligent Transportation Systems, IEEE Transactions on 11*(4), 917–930.

Gonzalez, A., M. Garrido, D. Llorca, M. Gavilan, J. Fernandez, P. Alcantarilla, I. Parra, F. Herranz, L. Bergasa, M. Sotelo, and P. Revenga de Toro (2011, june). Automatic Traffic Signs and Panels Inspection System Using Computer Vision. *Intelligent Transportation Systems, IEEE Transactions on 12*(2), 485–499.

Grigorescu, C. and N. Petkov (2003). Distance sets for shape filters and shape recognition. *Image Processing, IEEE Transactions on 12*(10), 1274–1286.

Gu, Y., T. Yendo, M. Tehrani, T. Fujii, and M. Tanimoto (2011, june). Traffic sign detection in dual-focal active camera system. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 1054–1059.

Hoessler, H., C. Wöhler, F. Lindner, and U. Kreßel (2007). Classifier training based on synthetically generated samples. In *Proceedings of 5th international conference on computer vision systems. Bielefeld, Germany.*

Hoferlin, B. and K. Zimmermann (2009, june). Towards reliable traffic sign recognition. In *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 324–329.

Houben, S. (2011, june). A single target voting scheme for traffic sign detection. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 124–129.

Ishida, H., T. Takahashi, I. Ide, Y. Mekada, and H. Murase (2006). Identification of degraded traffic sign symbols by a generative learning method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, Volume 1, pp. 531–534. IEEE.

Kastner, R., T. Michalke, T. Burbach, J. Fritsch, and C. Goerick (2010, june). Attention-based traffic sign recognition with an array of weak classifiers. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 333–339.

Keller, C., C. Sprunk, C. Bahlmann, J. Giebel, and G. Baratoff (2008, june). Real-time recognition of U.S. speed signs. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 518–523.

Kuo, W.-J. and C.-C. Lin (2007, july). Two-Stage Road Sign Detection and Recognition. In *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 1427–1430.

Lafuente-Arroyo, S., S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras, and R. J. López-Sastre (2010, January). A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems. *Expert Syst. Appl. 37*, 767–773.

Larsson, F. and M. Felsberg (2011). Using fourier descriptors and spatial models for traffic sign recognition. *Image Analysis*, 238–249.

Liao, S., X. Zhu, Z. Lei, L. Zhang, and S. Li (2007). Learning multi-scale block local binary patterns for face recognition. *Advances in Biometrics*, 828–837.

Lienhart, R. and J. Maydt (2002). An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, Volume 1, pp. I–900. Ieee.

Liu, H., D. Liu, and J. Xin (2002). Real-time recognition of road traffic sign in motion image based on genetic algorithm. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, Volume 1, pp. 83–86. IEEE.

Loy, G. and N. Barnes (2004). Fast shape-based road sign detection for a driver assistance system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, Volume 1, pp. 70–75. IEEE.

Loy, G. and A. Zelinsky (2003, aug.). Fast radial symmetry for detecting points of interest. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 25*(8), 959–973.

Maldonado-Bascon, S., S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras (2007). Road-sign detection and recognition based on support vector machines. *Intelligent Transportation Systems, IEEE Transactions on 8*(2), 264–278.

Mao, X., F. Qi, and W. Zhu (2007). Multiple-part based Pedestrian Detection using Interfering Object Detection. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, Volume 2, pp. 165–169. IEEE.

Marziliano, P., F. Dufaux, S. Winkler, and T. Ebrahimi (2002). A no-reference perceptual blur metric. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, Volume 3, pp. III–57–III–60 vol.3.

Meuter, M., C. Nunn, S. M. Gormer, S. Muller-Schneiders, and A. Kummert (2011, December). A Decision Fusion and Reasoning Module for a Traffic Sign Recognition System. *Intelligent Transportation Systems, IEEE Transactions on 12*(4), 1126–1134.

Morris, B. and M. Trivedi (2010). Vehicle Iconic Surround Observer: Visualization platform for intelligent driver support applications. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 168–173. IEEE.

Moutarde, F., A. Bargeton, A. Herbin, and L. Chanussot (2007). Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system. In *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 1122–1126. IEEE.

Murphy-Chutorian, E., A. Doshi, and M. Trivedi (2007). Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 709–714. IEEE.

Nguwi, Y.-Y. and A. Kouzani (2008). Detection and classification of road signs in natural environments. *Neural Computing & Applications 17*, 265–289.

Nunn, C., A. Kummert, and S. Muller-Schneiders (2008, sept.). A two stage detection module for traffic signs. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pp. 248–252.

Overett, G. and L. Petersson (2011, june). Large scale sign detection using HOG feature variants. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 326–331.

Overett, G., L. Tychsen-Smith, L. Petersson, N. Pettersson, and L. Andersson (2011). Creating robust high-throughput traffic sign detectors using centre-surround HOG statistics. *Machine Vision and Applications*, 1–14.

Pettersson, N., L. Petersson, and L. Andersson (2008, june). The histogram feature - a resource-efficient Weak Classifier. In *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 678–683.

Porikli, F. (2005). Integral histogram: A fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Volume 1, pp. 829–836. IEEE.

Prisacariu, V., R. Timofte, K. Zimmermann, I. Reid, and L. Van Gool (2010, aug.). Integrating Object Detection with 3D Tracking Towards a Better Driver Assistance System. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 3344–3347.

Qingsong, X., S. Juan, and L. Tiantian (2010, april). A detection and recognition method for prohibition traffic signs. In *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, pp. 583–586.

Rajesh, R., K. Rajeev, K. Suchithra, V. Lekhesh, V. Gopakumar, and N. Ragesh (2011, 31 2011-aug. 5). Coherence vector of Oriented Gradients for traffic sign recognition using Neural Networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 907–910.

Ren, F., J. Huang, R. Jiang, and R. Klette (2009, nov.). General traffic sign recognition by feature matching. In *Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference*, pp. 409–414.

Ruta, A., Y. Li, and X. Liu (2007). Towards real-time traffic sign recognition by class-specific discriminative features. In *Proc. of the 18th British Machine Vision Conference*, Volume 1, pp. 399–408.

Ruta, A., Y. Li, and X. Liu (2010). Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognition 43*(1), 416–430.

Ruta, A., F. Porikli, S. Watanabe, and Y. Li (2011). In-vehicle camera traffic sign detection and recognition. *Machine Vision and Applications 22*, 359–375.

Sermanet, P. and Y. LeCun (2011). Traffic sign recognition with multi-scale Convolutional Networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 2809–2813. IEEE.

Shinar, D. (2007). *Traffic safety and human behaviour*. Emerald Group Publishing.

Stallkamp, J., M. Schlipsing, J. Salmen, and C. Igel (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 1453–1460. IEEE.

Stallkamp, J., M. Schlipsing, J. Salmen, and C. Igel (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* (0), –.

Suard, F., A. Rakotomamonjy, A. Bensrhair, and A. Broggi (2006). Pedestrian detection using infrared images and histograms of oriented gradients. In *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 206–212. Ieee.

Timofte, R., K. Zimmermann, and L. Van Gool (2009). Multi-view traffic sign detection, recognition, and 3d localisation. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pp. 1–8. Ieee.

Timofte, R., K. Zimmermann, and L. Van Gool (2011, December). Multi-view Traffic Sign Detection, Recognition, and 3D Localisation. *Machine Vision and Applications*, 1–15.

Tran, C. and M. M. Trivedi (2011). Vision for Driver Assistance: Looking at People in a Vehicle. In T. B. Moeslund, L. Sigal, V. Krueger, and A. Hilton (Eds.), *Guide to Visual Analysis of Humans: Looking at People*.

Trivedi, M. and S. Cheng (2007). Holistic sensing and active displays for intelligent driver support systems. *Computer 40*(5), 60–68.

Trivedi, M., T. Gandhi, and J. McCall (2007). Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *Intelligent Transportation Systems, IEEE Transactions on 8*(1), 108–120.

United Nations Economic Commission for Europe (2006). Convention on Road Signs And Signals, of 1968.

Vázquez-Reina, A., S. Lafuente-Arroyo, P. Siegmann, S. Maldonado-Bascón, and F. Acevedo-Rodríguez (2005). Traffic sign shape classification based on correlation techniques. In *Proceedings of the 5th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, pp. 149–154. World Scientific and Engineering Academy and Society (WSEAS).

Viola, P. and M. Jones (2001). Robust real-time object detection. *International Journal of Computer Vision 57*(2), 137–154.

Wu, B. and R. Nevatia (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, Volume 1, pp. 90–97. IEEE.

Wu, B. and R. Nevatia (2007). Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision 75*(2), 247–266.

Xie, Y., L.-F. Liu, C.-H. Li, and Y.-Y. Qu (2009, june). Unifying visual saliency with HOG feature learning for traffic sign detection. In *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 24–29.

Xu, S. (2009, march). Robust traffic sign shape recognition using geometric matching. *Intelligent Transport Systems, IET 3*(1), 10–18.

Yongzhi, W., X. Jianping, L. Xiling, and Z. Jun (2010). Pedestrian Detection Using Coarse-to-Fine Method with Haar-Like and Shapelet Features. In *Multimedia Technology (ICMT), 2010 International Conference on*, pp. 1–4. IEEE.

Yuan, X., X. Shan, and L. Su (2011). A Combined Pedestrian Detection Method Based on Haar-Like Features and HOG Features. In *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pp. 1–4. IEEE.

Zaklouta, F., B. Stanciulescu, and O. Hamdoun (2011, 31 2011-aug. 5). Traffic sign classification using K-d trees and Random Forests. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 2151–2155.

A

Submitted papers

The following pages contain the papers that were submitted to journals and conferences during the project. The papers are attached in the shape they had at the deadline of this report, but as they make it through the review process, they will most likely change before the publications. The papers are:

- *Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey* submitted to IEEE Intelligent Transportation Systems Transactions, Special Issue on Machine Learning for Traffic Sign Recognition.
- *Traffic Sign Detection and Analysis: Recent Studies and Emerging Trends* submitted to IEEE Intelligent Transportation Systems Conference (ITSC), 2012.
- *Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets* submitted to IAPR 21st International Conference on Pattern Recognition (ICPR), 2012
- *A Two-stage Part-Based Pedestrian Detection System Using Monocular Vision* submitted to IEEE Intelligent Transportation Systems Conference (ITSC), 2012.

Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey

Andreas Møgelmose, Mohan M. Trivedi, and Thomas B. Moeslund

***Abstract*—In this paper, we provide a survey of the traffic sign detection literature, detailing detection systems for Traffic Sign Recognition (TSR) for driver assistance. We separately describe the contributions of recent works to the various stages inherent in traffic sign detection: segmentation, feature extraction, and final sign detection. While TSR is a well-established research area, we highlight open research issues in the literature, including a dearth of use of publicly-available image databases, and the over-representation of European traffic signs. Further, we discuss future directions for TSR research, including integration of context and localization. We also introduce a new public database containing US traffic signs.**

***Index terms:* Machine vision, machine learning, object detection, active safety, human-centered computing.**

I. Introduction

In this paper, we provide a survey of traffic sign detection for driver assistance. State-of-the-art research utilizes sophisticated methods in computer vision for traffic sign detection and it has been an active area of research over the past decade. On-road applications of vision have included lane detection, driver distraction detection, and occupant pose inference. As described in [1]–[3], it is crucial to not only consider the car's surrounding and external environment when designing an assist system, but also to consider the intenal environment and take the driver into account. Fusing other types of information with the sign detector, as described in [4], can make the overall system even better.

When the system is considered a distributed system where the driver is an integral part, it allows for the driver to contribute with what he is good at (e.g. seeing speed limit signs, as we shall see later), while the TSR part can present information from other signs. In addition other surround sensors can also have an influence on what is presented.

In recent years, speed limit detection systems have been included in top of the line models from various manufacturers, but a more general sign detection solution and an integration into other vehicle systems has not yet materialized. Current state-of-the-art TSR systems neither utilize information about the driver, nor input from the driver, to enhance performance. Extensive studies in Human-Machine Interactivity are necessary to present the TSR information in a careful way, to inform the driver without causing distraction or confusion. The literature features just two surveys on TSR: [5] is a good introduction, but not very comprehensive. [6] is a few years old, so any improvements in the field from the past 5 years are not presented. A very good comparison of various segmentation methods is offered in [7], but given that it only covers segmentation, it is not a comprehensive overview of detection methods. Likewise, [8] provides a good comparison of Hough transform derivatives. In this paper our emphasis is on framing the TSR problem in the context of human-centered driver assistance systems. We provide a comparative discussion of papers published mostly within the last 5 years and to provide an overview of the recent work in the area of sign detection, a subset of the TSR problem.

We provide a critical review of traffic sign detection, and offer suggestions for future research areas in this challenging problem domain. The next section establishes the driver assistance context and covers TSR systems in general. Section 3 provides a problem description, and a gentle introduction to traffic sign detection. Section 4 deals with segmentation for traffic sign detection. Section 5 details models and feature extraction. Section 6 deals with the detection itself. In the final section, the authors provide analysis and insight on future research directions in the field.

II. Human-Centered TSR for Driver Assistance: Issues and Considerations

Traffic sign recognition research needs to take into account the visual system of the driver. This can include factors such as visual saliency of signs, driver focus of attention, and cognitive load. According to [9] (see table I for a summary of the main results), not all signs are equal in their ability to capture the attention of the driver. For example, a driver may fixate his gaze on a sign, but neither notice the sign, nor remember its informational content. While drivers invariably fixate on speed limit signs and recall their information, they are less likely to notice game crossing and pedestrian signs. This can endanger pedestrians, as it may not leave enough reaction time to stop.

The implications of use of TSR in human-in-the-loop system are clear; instead of focusing on detection and recognizing all signs of some class perfectly, which would be the objetive

Manuscript received ...

A. Møgelmose is a master's student at Vision, Graphics, and Interactive Systems, at Aalborg University, Denmark. E-mail: andreasm@es.aau.dk

M. M. Trivedi is head of the CVRR lab at University of California, San Diego

T. B. Moeslund is head of the VAP Lab at Aalborg University, Denmark

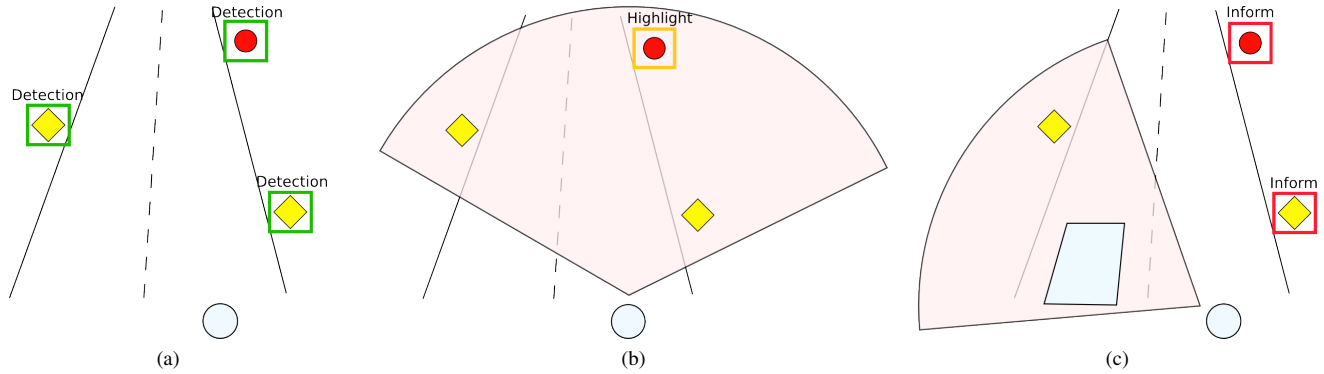


Fig. 1. Different detection scenarios. The circle is the ego-car and 3 signs are distributed along the road. The area highlighted in red illustrates the driver's area of attention. (a) is the standard scenario used for e.g. autonomous cars. Here, all signs must be detected and processed. (b) and (c) depicts a system which tracks the driver's attention. In (b), the driver is attentive and spots all signs. Therefore the system just highlights the one sign that is known to be difficult for people to notice. In (c), the driver is distracted by a passing car and thus misses two signs. In this case, the system should inform the driver about the two missed signs.

for an autonomous car, the task is now to detect and highlight signs that the driver has not seen. This gives way to various models of TSR, which take into account the driver's focus of attention, and interactivity issues. Driver attention tracking is covered in [10] and [11]. Fig. 1 presents examples on how TSR can be used for driver assistance. Fig. 1a shows how a system should act in an autonomous car. It simply recognizes all signs present. In fig. 1b there is a driver in the loop, and while the system may see all the signs, it should avoid presenting them in order to avoid driver confusion. Instead, it simply highlights the sign type that is easy to overlook, like the pedestrian crossing warnings in the research. Fig. 1c shows how a driver is distracted by a passing car. This causes him to miss two signs. His car has a TSR system for driver assistance, which informs him of the signs as he returns his attention to the road ahead of him. This could, for example, be done using a heads-up display as suggested in [12].

Even though this paper is mostly concerned with using TSR for driver assistance, TSR has various well defined applications, summarized nicely by [13]:

1) Highway maintenance: Check the presence and condition of signs along major roads.
2) Sign inventory: Similar to the above task, create an inventory of signs in city environments.
3) Driver assistance systems: Assist the driver by informing of current restrictions, limits, and warnings.
4) Intelligent autonomous vehicles: Any autonomous car that is to drive on public roads must have a means of obtaining the current traffic regulations. This can be done through TSR.

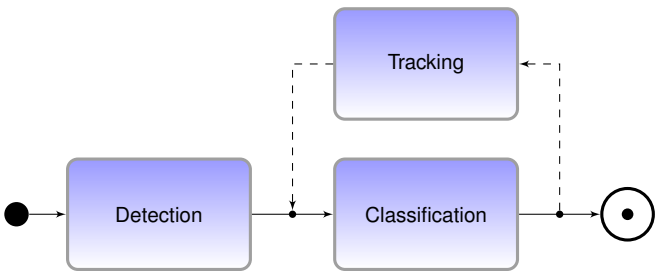


Fig. 2. The basic flow in most TSR systems.

This paper uses the term TSR to refer to the entire chain from detection of signs to their classification, and potentially presentation to the driver. Generally, TSR is split into two stages: Detection and classification (see fig. 2). Detection is concerned with locating signs in input images, while classification is about determining what type of sign the system is looking at. The two tasks can often be treated as completely separate, but in some cases the classifier relies on the detector to supply information, such as the sign shape or sign size. In a full system, the two stages are depending on each other and it does not make sense to have a classifier without a detection stage. Later, we divide the detection stage into three sub-stages, but these should not be confused with the two main stages of a full TSR-system: Detection and classification.

Apart from shape and color, another aspect may be used in TSR: Temporal information. Most TSR systems are designed with a video feed from a vehicle in mind, so signs can be tracked over time. The simplest way of using tracking is to accept sign candidates as signs only if they have shown up on a number of consecutive frames. Sign candidates that only show up once are usually a result of noise. Employing a predictive method, such as a Kalman filter, allows for the system to predict where a sign candidate should show up in the next frame, and if its position is too far away from this prediction, the sign candidate is discarded. A predictive tracking system has the additional benefit of handling occlusions, hence preventing signs that were occluded from being classified as new signs. This is very important in a driver assistance system where signs should only be presented once, and in a consistent way. Imagine a scenario where a sign is detected in a few frames and the occluded for a short while, before being detected again. For an autonomous car it is likely not a problem to be presented with the same information twice: If the first sign prompted the speed to be set at 55 mph, there is no problem in the

TABLE I
SIGNIFICANT RESULTS FROM [9] REGARDING ATTENTION TO VARIOUS SIGN TYPES.

	Fixated		Not fixated	
Target	Recalled	Not recalled	Recalled	Not recalled
Speed limit 80 km sign	100	0	0	0
Game Crossing sign	60	0	7	33
Pedestrian crossing ahead	8	54	0	38
Pedestrian crossing sign	0	21	0	79

system being told once again that the speed limit is 55 mph. In a driver assistance system, the system must not present more information than absolutely necessary at any given moment, so the driver is not overwhelmed with information, for instance, forcing the driver to pay attention to a sign he has already seen should be avoided.

Many TSR systems are tailored to a specific sign type. Due to the vast differences in sign design from region to region (see the following section), and the differences in sign design based on their purpose, many systems narrow their scope down to a specific sign type in a specific country.

There is a wide span in speeds of the systems. For use in driver assistance and autonomous vehicles, real-time performance is necessary. This does not necessarily mean a speed of 30 Hz, but the signs must be read quickly enough to still be relevant to act on. Depending on the exact application, a few Hz is required.

Instead of treating the entire TSR-process in what could easily become a cursory manner, we have opted to look thoroughly on the detection stage. The line between detection and classification is a bit blurry, since some detectors provide more information to the classifier than others. It is normal for the detector to inform the classifier of the general category of signs, since that is often defined by either the overall sign shape or its color, something that the detector itself may use to to localize the sign.

Even though this paper is targeted towards the problem of detecting traffic signs, one must not forget that without a subsequent classification stage, the systems are useless. So even though we encourage a decoupling of the two tasks, this does not mean that the classification is a solved problem. It is a crucial part of a full system.

III. TRAFFIC SIGNS

Traffic signs are markers placed along roads to inform drivers about either road conditions and restrictions or which direction to go. They communicate a wealth of information, but are designed to do so efficiently and at a glance. This also means that they are often designed to stand out from their surroundings, making the detection task fairly well defined.

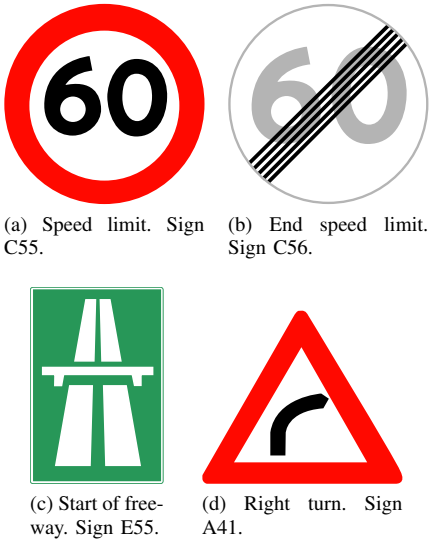
The designs of traffic signs are standardized through laws, but differ across the world. In Europe many signs are standardized via the Vienna Convention on Road Signs And Signals [14]. There, shapes are used to categorize different types of signs: Circular signs are prohibitions including speed limits, triangular signs are warnings and rectangular signs are used for recommendations or sub-signs in conjunction with one of the other shapes. In addition to these, octagonal signs are used to signal a full stop, downwards pointing triangles yield, and countries have other different types, e.g. to inform about city limits. Examples of these signs can be seen in fig. 3.

In the US, traffic signs are regulated by the Manual on Uniform Traffic Control Devices (MUTCD) [15]. It defines which signs exist and how they should be used. It is accompanied by the Standard Highway Signs and Markings (SHSM) book, which describes the exact designs and measurements of signs. At the time of writing, the most recent MUTCD was from 2009, while the SHSM book had not been updated since 2004, and thus it described the MUTCD from 2003. The MUTCD contains a few hundred different signs, divided into 13 categories.

To complicate matters further, each US state can decide whether it wishes to follow the MUTCD. A state has three options:

1) Adopt the MUTCD fully as is.
2) Adopt the MUTCD but add a State Supplement.
3) Adopt a State MUTCD that is "in substantial conformance with" the national MUTCD.

In the US 19 states have adopted the national MUTCD without modifications, 23 have adopted the national MUTCD



(a) Speed limit. Sign C55.
(b) End speed limit. Sign C56.
(c) Start of freeway. Sign E55.
(d) Right turn. Sign A41.

Fig. 3. Examples of European signs. These are Danish, but many countries use similar signs.

(a) Stop. Sign R1-1. (b) Yield. Sign R1-2.

(c) Speed limit. Sign R2-1. (d) Turn warning with speed recommendation. Sign W1-2a.

Fig. 4. Examples of signs from the US national MUTCD. Image source: [15]

with a state supplement and 10 have opted to create a State MUTCD (the count includes the District of Columbia and Puerto Rico). Examples of US signs can be seen in fig. 4.

New Zealand uses a sign standard with warning signs that are yellow diamonds, as in the US, but regulatory signs that are round with a red border, like the ones from the Vienna Convention countries. Japan uses signs that are generally in compliance with the Vienna Convention, as are Chinese regulatory signs. Chinese warning signs are triangular with a black/yellow color scheme. Central and South American countries do not participate in any international standard, but often use signs somewhat like the American standard.

While signs are well defined through laws and designed to be easy to spot, there are still plenty of challenges for TSR systems. They include:

- Signs being similar within or across categories (see fig. 5).
- Signs may have faded or be dirty so they are no longer their specified color.
- The sign post may be bent, so the sign is no longer orthogonal to the road.
- Lighting conditions may make color detection unreliable.
- Low contrast may make shape detection hard.
- In cluttered urban environments, other objects may look very similar to signs.
- Varying weather conditions.

A. Assessing performance of sign detectors

When comparing sign detectors, some comparison metrics must be set up. The straight forward and most important measure is the true positive rate. However, even if all signs are detected, the system is not necessarily perfect. The number of false positives must also be taken into account. If the amount of false positives is too high, the classifier will have to handle a lot more data than it should, degrading the overall system speed. For cases when a system must work in real-time in a car, obviously the detection must be fast. In general, the faster the detection runs, the more time is left over for the classification stage. Adjusting these goals is a trade-off. Often, the target will be to create a system that is just fast enough for a given application, while keeping the receiver operating characteristic acceptable. Another interesting performance characteristic is what sign types a given system works for.

Even with the parameters in mind, and a clear idea of the performance metrics, comparing the performance of different systems is not a straightforward task. Unlike other computer vision areas, until recently no standardized training and test data set existed, so no two systems were tested with the same data. The image quality varies from high resolution still images (as in [16]–[18]) to low resolution frames from in-car video cameras (such as [19]–[21]). That, combined with the facts that signs vary wildly between countries, and many papers limit their scope to specific sign types, makes for a quite uneven playing field.

For a discussion of the performance of the papers presented in this survey, see section IV

B. Public sign databases

A few publicly available traffic sign datasets exist:

- German Traffic Sign Recognition Benchmark (GTSRB) [22], [23]
- KUL Belgium Traffic Signs Dataset (KUL Dataset) [24]
- Swedish Traffic Signs Dataset (STS Dataset) [25]
- RUG Traffic Sign Image Database (RUG Dataset) [26]
- Stereopolis Database [27]

Information on these databases can be found in table II. Most of the databases have emerged within the last two years (except for the very small RUG Dataset), and are not yet widely used. One of the most widespread databases is the GTSRB, which has been presented in [22], created for the competition “The German Traffic Sign Recognition Benchmark”. The competition was held at the International Joint Conference on Neural Networks (IJCNN) 2011. It is a large data set containing German signs, thus very suitable for training and testing systems aimed at signs adhering to the Vienna Convention. A sample image from the GTSRB database can be found in fig. 6a. The GTSRB is primarily

(a) Speed limit. Sign R2-1. (b) Minimum speed. Sign R2-4. (c) End speed limit. Sign R3 (CA).

Fig. 5. Examples of similar signs from the MUTCD. (c) exists only in the California MUTCD. Image source: [15]

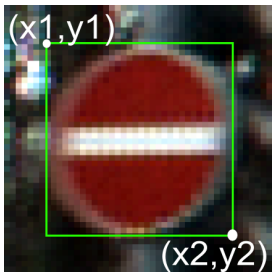
TABLE II
INFORMATION ON THE PUBLICLY AVAILABLE SIGN DATABASES.

	GTSRB	**STS Dataset**	**KUL Dataset**	**RUG Dataset**	**Stereopolis**	**LISA Dataset**
Number of classes:	43	7	100+	3	10	49
Number of annotations:	50000+	3488	13444	0	251	7855
Number of images:	50000+	20000	9006	48	847	6610
Annotated images:	All images	4000 images	All images	0	All images	All images
Sign sizes:	15x15 to 250x250 px	3x5 to 263x248 px	100x100 to 1628x1236 px	N/A	25x25 to 204x159 px	6x6 to 167x168 px
Image sizes:	15x15 to 250x250 px	1280x960 px	1628x1236 px	360x270 px	1920x1080 px	640x480 to 1024x522 px
Includes videos:	No	No	Yes, 4 tracks	No	No	Yes, for all annotations
Country of origin:	Germany	Sweden	Belgium	The Netherlands	France	United States
Extra info:	Images come in tracks with 30 different images of the same physical sign.	Signs marked visible/blurred/occluded and whether they belong to the current road or a side road.	Includes traffic sign annotations, camera calibrations and poses.	Does not include any annotations, only raw pictures.		Images from various camera types.

geared towards classification, rather than detection, since each image contains exactly one sign without much background. For detection, images of complete scenes is necessary. Also, many detection systems rely on a tracking scheme to make detection more robust and without video of the tracks (in GTSRB parlance a "track" is a set of images of the same physical sign), this will not work properly. Since the data set is created for the classification task, this is not so much a problem of that database, as it is a testament to its target. In conjunction with the competition, five interesting papers [28]–[32] were released. They all focus on classification rather than detection.

Two other datasets should be highlighted: The STS Dataset and the KUL Dataset. They are both very large, though not as large as the GTSRB, and they contain full images. This means that they can both be used for detection purposes. The STS Dataset does not have all images annotated, but it does include all frames from the videos used to obtain the data. This means that tracking systems can be used on this dataset, but it can only be verified with ground truth every 5 frames. An example from the STS Dataset can be seen in fig. 6b. The KUL Dataset also includes 4 recorded sequences which can be used for tracking experiments. KUL also includes a set of sign-free images which can be used as negative training images and it has pose-information for the cameras for each image.

From the research it was evident that there was a lack of databases with US traffic signs, so in conjunction with this paper we have assembled one. Its details are also listed in table II. One novel feature of this dataset is that it includes video tracks of all the annotated signs. Many systems already use various tracking schemes to minimize the number of false positives, and it is quite likely that in the future, detectors using temporal data even more will emerge. Therefore, the



(a)

(b)

Fig. 6. Example sign images from (a) the GTSRB and (b) the STS Dataset with the sign bounding boxes superimposed.

LISA dataset includes video as well as stand alone frames. Not all frames have been extracted for annotation, but all annotated frames can be traced back to the source video so the annotations can also be used to verify systems using tracking.

IV. Sign detection

The approaches in this stage have traditionally been divided into two kinds:

- Color based methods.
- Shape based methods.

Color based methods take advantage of the fact that traffic signs are designed to be easily distinguished from their surroundings, often colored in highly visible contrasting colors. These colors are extracted from the input image and used as a base for the detection. Just like signs have specific colors, they also have very well defined shapes that can be searched for. Shape based methods ignore the color in favor of the characteristic shape of signs.

Each method has its pros and cons. Color of signs, while well defined in theory, varies a lot with available lighting, as well as with age and condition of the sign. On the other hand, searching for specific colors in an image is fairly straight forward. Sign shapes are invariant to lighting and age, but parts of the sign can be occluded, making the detection harder, or the sign may be located at a background of a similar color, ruining the edge detection that most shape detectors rely on.

The division of systems in this way can be problematic. Almost all color based approaches take shape into account after having looked at colors. Others use shape detection as their main method, but integrate some color aspects as well. Instead, the detection can be split into two steps as proposed by [7]: Segmentation and detection. In this paper we go one step further and split the detection step into a feature extraction step and the actual detection, which acts on the features that are extracted. Many shape-only based methods have no segmentation step. The flow is outlined in fig. 7.

An overview of all surveyed papers and their methods is listed in table III. It contains each of the systems and lists which segmentation method, feature type, and detection method that is used. The author group numbers are used to mark the papers that are part of an ongoing effort from the same group of authors. They do not constitute a ranking in any way. In tables IV and V, some of their more detailed properties are listed. The systems are split into two tables. Table IV displays those which do not use any tracking. Table V contain those which do use tracking, something we find crucial when using TSR in a driver assistance context, as mentioned earlier. Apart from this division, the two tables are structured in the same way: *Sign type in paper* describes which sign types the authors of the paper have attempted to find, while emphsign type possible are the types of signs the method could be extended to include, usually a very broad group. *Real-time* is about how fast the system runs, if that information is available. Any system with a frame rate faster than 5 fps is considered to have real-time potential. *Rotation invariance* tells whether the used technique is robust to rotation of signs. *Model vs. training* describes if the detection system relies on a theoretical model of signs (such as a pre-defined shape), if it uses a learned type of classifier, or if it uses a combination of the two. *Test image type* is the image resolution the system is designed to work with. Low-res images are usually video frames, while high-res are still images.

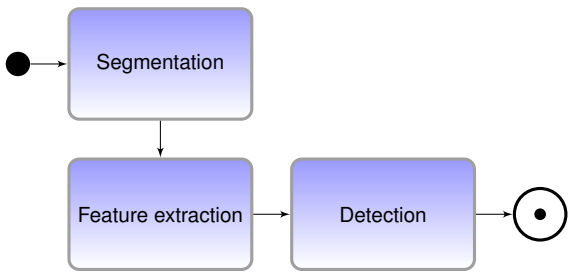


Fig. 7. The general flow followed by typical sign detection algorithms.

The detection performance of the surveyed papers are presented in table VI. As mentioned earlier, very few papers use common databases to test their performance and the papers detect various types and numbers of signs. Thus, the numbers should not be directly compared, but nevertheless they give an idea of performance. Not all papers report all the measures reported in the table (detection rate, false positives per frame, etc.), so some fields in the table could not be filled. In other cases these exact measures were not given, but could be calculated from other given numbers. Where figures are available, the best detection rate the system obtained is reported along with the corresponding measure of false positives. The detection rate is per frame, meaning that 100% detection is only achieved if a sign is found in every frame it is present. It is not sufficient to just detect the sign in a few frames. This is the way results are presented in most papers, so this is the measure chosen here, even if a real-world system would work fine if each sign is just detected once. Papers which only report the per-sign detection rate as opposed to the per-frame detection rate are marked with a triangle in the right-most column of the table.

Different papers report the false positives in different ways, so a few different measures - which are not directly comparable - are presented in the table:

- FPPF False postives per frame: $FPPF = \frac{FP}{f}$ where FP is the number of false positives and f is the number of frames analyzed.
- FPR False positive rate: $FPR = \frac{FP}{N}$ where N is the number of negatives in the test set. This measure is rarely used in detection, since the number of negatives does not always make much sense (how many negatives exist in a full frame?).
- PPV Positive predictive value: $PPV = \frac{TP}{TP+FP}$ where TP is the number of true positives.
- FPTP False/true positive ratio: $FPTP = \frac{FP}{TP}$
- WPA Wrong pixels per area: $WPA = \frac{WP}{AP}$ where WP is the number of wrongly classified pixels and AP is the total number of pixels classified.

When papers present results for different sign types, the mean detection performance is also presented in the table. In many cases that will give a better view of the true performance of the approach.

Five papers stick out, claiming a 100% detection rate. The first [33] is only tested on synthetic data. It is possible that the synthetic data does not fully encapsulate real world variations, so the performance of that approach is not guaranteed to be as good in real-world scenarios. At first glance [34] achieves a 100% detection rate, but that is only the case for one of their sign types. The mean performance is a more accurate (and still promising) gauge of the actual performance. The same is the case for [25]. [35] detects all signs in the test set, but at the cost of a large number of false positives per frame. [36] only presents the per-sign detection rate, so that figure cannot be compared to the other systems.

Generally, systems achieve detection rates well into the 90% range, some at very low false detection rates. From the table no "best system" can be chosen, since the test sets are very different, both in size and content. A system that can detect several different sign types at a low detection rate may in some applications be considered better than a system that can only detect one specific sign type, but do that very well. A few papers that should be highlighted are [18], [37]–[39]. They have all been tested on large datasets and report detection rates above 90% with a decent low number of false positives.

Now that the basics about sign detection are in place, the following sections go in depth with how recent papers perform each step.

V. Segmentation

The purpose of the segmentation step is to achieve a rough idea about where signs might be, and thus narrow down the search space for the next steps. Not all authors make use of this step. Since the segmentation is traditionally done based on colors, authors who believe this should not be part of a sign detection often do not have any segmentation step, but go directly to the detection.

Of the papers that do use segmentation, all except [38], [40] use colors to some extent. Normally, segmentation is done with colors and subsequently a shape detection is run in a later stage. In [38], the usual order is reversed, so they use radial symmetry voting (see section VII) for segmentation and a color based approach for the detection. [40] also run radial symmetry voting as preprocessing, but follow it up with a cascaded classifier using Haar wavelets (see again section VII).

Generally, color based segmentation relies on a thresholding of the input image in some color space. Since many believe that the RGB color space is very fragile with regards to changes in lighting, these methods are spearheaded by the HSI-space (or its close sibling, the HSV-space). HSI/HSV is used by [41]–[46]. The HSI-space models the human vision better than RGB and allows some variation in the lighting, most notably in the intensity of light. Some papers, like the ones in the series starting with [16] and followed by [33], [47], [48], augment the HSI thresholding with a way to find white signs. Hue and saturation are not reliable for detecting white, since it can be at any hue, so they use an achromatic decomposition of the image proposed by [49].

Some authors are not satisfied with the performance of HSI, since it does not model the change in color temperature in different weather, but only helps in changing light intensity. [17], [50] instead threshold in the LCH color space, which is obtained using the CIECAM97-model. This allows them to take variations in color temperature into account. The RGB space is used by [18], [51], but they use an adaptive threshold in an attempt to combat instabilities caused by lighting variations.

Of special interest in this color space discussion is the excellent paper [7], which has shown that HSI-based segmentation offers no significant benefit over normalized RGB, but that methods which use color segmentation generally perform much better than shape-only methods. They do, however have trouble with white signs. For a long time, it has simply been assumed that the RGB color space was a bad choice for segmentation, but through rigorous testing, they show that there is nothing to gain from switching to the HSI color space instead of a normalized RGB space. As the authors write: "Why use a nonlinear and complex transformation if a simple normalization is good enough?".

A color based model not relying on thresholding was put forward in [52], which use a cascaded classifier trained with AdaBoost, similar to the one proposed by [53], but on Local Rank Pattern features instead of Haar wavelets. Also, [34] use a color-based search method that, while closely related to, is not directly thresholding-based. Here, the image is discretized into colors that may exist on signs. The discretization process is less destructive than thresholding in that it does not directly discard pixels, instead it maps them into the closest sign-relevant color. In a more recent contribution [20], they replace the color discretization method with a Quad-tree interest region finding algorithm, which finds interesting areas using an iterative search method for colored signs. In the same realm lies [8], which uses a learned probabilistic color preprocessing.

In [21], a unique approach is proposed: Using a biologically inspired attention system. It produces a heat map denoting areas where signs are likely to be found. An example can be seen in figure 9. A somewhat similar system was put forth by [19], who uses a saliency measure to find possible areas of interests.

VI. Features and modeling

While various features are available from the vision literature, the choice of feature set is often closely coupled with the detection method, though some feature sets can be used with a selection of different detection methods. The most popular feature is edges - sometimes edges obtained directly from the raw picture, sometimes edges from pre-segmented images. Edges are practically always found using a Canny edge detection or some method very similar, and they are used as the only feature in [8], [18], [20], [34], [35], [41], [43], [45], [46], [49], [52], [54]–[61]. [51] combine the edges with Haar-like features and [36], [62] look only at certain color filtered edges.

(a) Before thresholding (b) After thresholding

Fig. 8. An example of thresholding, looking for red hues.

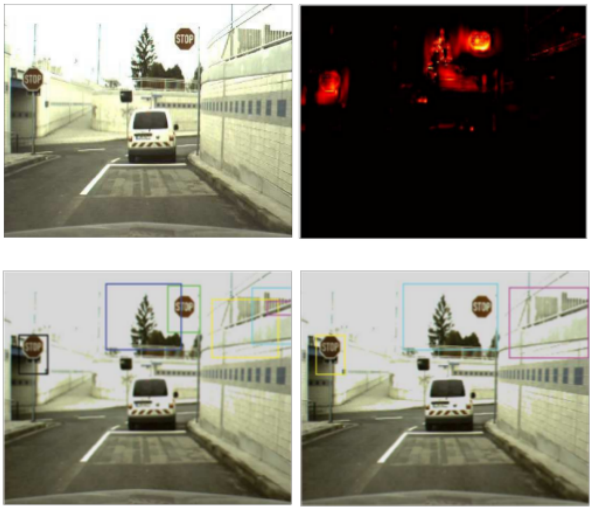


Fig. 9. The biologically inspired detection stage from [21]. Image source: [21]

Even though edges comprise the most popular feature choice, there are other options. Histogram of Oriented Gradients (HOG) is one. It was first used to detect people in images, but has been used by [17], [19], [39], [63], [64] to detect signs. HOG is based on creating histograms of gradient orientations on patches of the image and comparing them to known histograms for the sought after objects. HOG is also used in [65], but they augment the HOG feature vectors with color information to make them even more robust.

A number of papers [37], [40], [51], [66] use Haar wavelet-like features, [66] only on certain colors, and [37] in the form of so-called dissociated dipoles with wider structure options than traditional Haar wavelets.

More esoteric choices are Distance to Bounding box (DtB), FFT of shape signatures, tangent functions, simple image patches, and combinations of various simple features. DtB, as used in [47], [48], are a measure of distances from the contour of a sign-candidate to its bounding box. Similarly, the FFT of shape signatures used in [33] is based on the distance from the shape center to its contour at different angles. Tangent functions, used in [44], calculate the angles of the tangents at various points around the contour. Simple image patches (though in the YCbCr color space) are championed by [42] and a combination of simple features, such as corner positions and color is used in [21].

an area that warrants further research.

VII. Detection

The detection stage is where the signs are actually found. This is in many ways the most critical step, and often also the most complicated. The selection of detection method is a bit more constrained than the previous two stages, since the method must work with the features from the previous stage. The decision is therefore often made the other way around: A desired detection method is chosen, and the feature extraction stage is designed to deliver what is necessary to perform the detection. As we know from the previous section, the most popular feature is edges, and this reflects on the most popular choice in detection method. Using Hough transforms to process the edges is one option, as done by [43], [58]–[60]. In [60], a proprietary and undisclosed algorithm is used for detection of rectangles in addition to the Hough transform used for circles. That said, Hough transforms are computationally expensive and not suited for systems with real-time requirements. Because of that, the most popular methods are derivatives of the radial symmetry detector first proposed in [67] and first put to use for sign detection in [68]. The algorithm votes for the most likely sign centers in an image based on symmetric edges and is itself inspired by the Hough transform. The basic principle can be seen in fig. 10. In a circle, all edge gradients intersect at the center. The algorithm finds gradients with a magnitude above a certain threshold. In the direction pointed out by the gradient, it casts a vote in a separate vote image. It looks for circles of a specific radius and thus votes only in the distance from the edge that is equivalent to the radius. The places with most votes are most likely to be the center of circles.

This algorithm was later extended to regular polygons by [35] and a faster implementation for sign detection use was proposed by [54]. It is also used in some form by [36], [38], [40], [55]–[57]. An example of votes from a system which is extended to work for rectangular signs can be seen on fig. 11. An alternate edge-based voting system is proposed by [61].

The HOG features can be used with an SVM, as in [19], [65], or be compared by calculating a similarity coefficient as in [17]. Another option with regard to HOG is to use a cascaded classifier trained with some type of boosting. This is

done in [39], [64]. Cascaded classifiers are traditionally used with Haar wavelets, and sign detection is no exception, as used in [37], [40], [51], [66].

Finally, also neural networks and genetic algorithms are represented in [42] and [49], respectively.

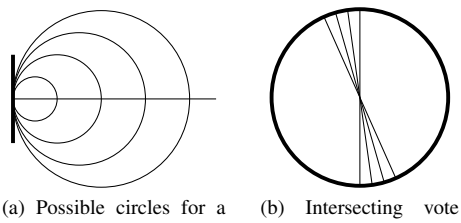
The detection stage reflects the philosophical difference that was also seen in the feature extraction stage: Either reliance on a simple, theoretical model of sign shapes is preferred - at this stage it is nearly always shapes that are searched for - or reliance on training data and then a more abstract detection method. Since it is extremely hard to compare systems tested across different data sets, it is not clear which methods perform the best, so that is clearly an area with a need for further studies. Both ways can be fast enough for real-time performance, and most of them could also work with signs of any shape. There are outliers using different methods, but no compelling argument that they should perform significantly better.

VIII. Discussion and future directions

In the previous sections, different methods and philosophies for each stage are presented. This section discusses the current state of the art and outlines ideas for future directions of research.

At the moment, the problem in TSR is the lack of use of standardized sign image databases. This makes comparisons between contributions very hard. In order to obtain meaningful advances in the field, the development of such databases is crucial. Until now, research teams have only implemented a method they believe has potential, or perhaps tested a few solutions. Without a way to compare performance with other systems, it is not clear which approaches work the best, so every new team starts back at square one, implementing what *they* think might work best. Two efforts to remedy this situation deserve to be mentioned: The sign databases presented earlier and the segmentation evaluation in [7]. As mentioned earlier (section III-B), a few public sign databases have recently emerged, but have not yet been widely used. In [7], the authors compare various segmentation methods on the same data set containing a total of 552 signs in 313 images. They also propose a way to evaluate the performance of segmentation methods. That paper provides a very good starting point for determining which segmentation method to use.

These two efforts notwithstanding, public databases covering signs from non-Vienna Convention regions are necessary. Databases which include video tracks of signs would also be very beneficial to the development of TSR systems, since many detectors employ a tracking system for signs. This is, to some extent, included in the KUL Dataset. In relation to the work on this present survey, we have assembled such a database for US traffic signs, one that includes full video tracks of signs. It is our hope that the GTSRB database will also be extended to include video and full frames and that more US databases will be created.



(a) Possible circles for a gradient (b) Intersecting vote lines

Fig. 10. The basic principle behind the radial symmetry detector. Image inspired by [55].

Fig. 11. Votes from a radial symmetry system superimposed to the original image. The brightest spot coincides with the center of the sign. This image is from a system developed in conjunction with this paper and is a radial symmetry voting algorithm extended to work for rectangles

The absence of usage of public database may not explain in entirety why very few comparative studies of methods exist. Another reason is that TSR systems are long, complex chains of various methods, where it is not always possible to swap individual modules. When it is not feasible to swap, say, the detection method for something else, it is naturally hard to determine whether other solutions may be better. This is solved, if more papers divide their work more clearly into stages, ideally as fine grained as the ones used in this survey, plus a similar set of stages for classification. This is done with success in [7], as they test different segmentation methods while keeping the feature extraction, detection, and classification stages fixed.

Another problem is the need for work on TSR in regions not adhering to the Vienna Convention. The bulk of the existing work comes out of Europe, Australia, and Japan. Japan and Australia are not parts of the Vienna Convention, but they use similar signs, for example to convey speed limits. Of the surveyed papers here, only two are concerned with US traffic signs [40], [60], and even they only look at speed limit signs.

When looking at sign detection from a driver-in-the-loop perspective, it is also unfortunate that the bulk of reseach now focuses on speed limit signs. A wealth of papers cite driver assistance as their main application, but carries on focusing on speed limit signs. Detection of speed limits is

Fig. 12. Example of sign relevancy challenges in a crop from our own collected data set. The signs have been manually highlighted, and while both signs would likely be detected, only the one to the right is relevant to the driver. The sign to the left belongs to another road, the one the black and white cars come from.

highly relevant for an autonomous vehicle, but as it turns out, humans are already very good at seeing speed limit signs themselves [9]. As such, recognition of signs other than speed limit is actually more interesting.

The final problem we wish to highlight in this section is the relation of signs to the surroundings. TSR has seen significant work, as is evident from this paper, but little work has been done on ensuring that the detected signs are relevant for the ego-car (with the notable exception of [58]). In many situations, it can occur that a detected sign is not connected to the road the car is on. An example from our own collected data can be seen in fig. 12. In this case, two stop signs can be seen, but only the rightmost one pertains to the current road. Similar situations occur often on freeways, where some signs may only be relevant for exit lanes. Related to this problem is that when the driver changes to a different road, most often the restrictions from earlier detected signs no longer apply. This should be detected and relayed to the system. It is very likely that research in other areas, such as lane detection can be of benefit here. Another idea with regard to the surroundings would be to link knowledge of weather and current lighting conditions to enhance the robustness of the detector, similar to what is done for detection of people in [69]. It is also possible that vehicle dynamics can be taken into account and used in the tracking of detected signs.

IX. Concluding remarks

This paper provides an overview of the state of sign detection. Instead of treating the entire TSR flow, focus has been solely on the detection of signs. During recent years, a large effort has gone into TSR, mainly from Europe, Japan, and Australia and the developments have been described.

The detection process has been split into segmentation, feature extraction, and detection. Many segmentation approaches exist, mostly based on evaluating colors in various color spaces. For features there are also a wealth of options. The choice is made in conjunction with the choice of detection method. By far the most popular features are edges and gradients, but other options such as HOG and Haar wavelets have been investigated. The detection stage is dominated by the Hough transform and its derivatives, but for HOG and Haar wavelet features, SVMs, neural networks, and cascaded classifiers have also been used.

Arguably, the biggest issue with sign detection as it is currently is the lack of use of public image databases to train and test systems. Currently, every new approach presented uses a new dataset for testing, making comparisons between papers hard. This gives the TSR effort a somewhat scattered look. Recently, a few databases have been made available, but they are still not widely used, and cover only Vienna Convention compliant signs. We have contributed with a new database, the LISA Dataset, which contains US traffic signs.

This issue leads to the main unanswered question in sign detection: Is a model based shape detector superior to a learned approach, or vice versa? Systems using both approaches exist, but are hard to compare, since they all use different data sets.

Many contributions cite driver assistance systems as their main motivation for creating the system, but so far only little effort has gone into the area of combining TSR systems with other aspects of driver assistance and notably, none of the studies include knowledge about the driver's behavoir in order to tailor the performance of the TSR system to the driver.

Other open issues include lack of research in finding non-European style signs and detected signs are hard to relate to their surroundings.

Acknowledgment

The authors would like to thank our colleagues in the LISA-CVRR lab, especially Mr. Sayanan Sivaraman, Mr. Minh Van Ly, Ms. Sujitha Martin, and Mr. Eshed Ohn-Bar for their comments.

References

[1] M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 1, pp. 108–120, 2007.

[2] M. Trivedi and S. Cheng, "Holistic sensing and active displays for intelligent driver support systems," *Computer*, vol. 40, no. 5, pp. 60–68, 2007.

[3] C. Tran and M. M. Trivedi, "Vision for Driver Assistance: Looking at People in a Vehicle," in *Guide to Visual Analysis of Humans: Looking at People*, T. B. Moeslund, L. Sigal, V. Krueger, and A. Hilton, Eds., 2011.

[4] B. Morris and M. Trivedi, "Vehicle Iconic Surround Observer: Visualization platform for intelligent driver support applications," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 168–173.

[5] M.-Y. Fu and Y.-S. Huang, "A survey of traffic sign recognition," in *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, july 2010, pp. 119–124.

[6] H. Fleyeh and M. Dougherty, "Road and traffic sign detection and recognition," in *10th EWGT Meeting and 16th Mini-EURO Conference*, 2005, pp. 644–653.

[7] H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo, "Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 4, pp. 917–930, dec. 2010.

[8] S. Houben, "A single target voting scheme for traffic sign detection," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 124–129.

[9] D. Shinar, *Traffic safety and human behaviour*. Emerald Group Publishing, 2007.

[10] A. Doshi and M. Trivedi, "Attention estimation by simultaneous observation of viewer and view," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 21–27.

[11] E. Murphy-Chutorian, A. Doshi, and M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*. IEEE, 2007, pp. 709–714.

[12] A. Doshi, S. Cheng, and M. Trivedi, "A novel active heads-up display for driver assistance," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 1, pp. 85–93, 2009.

[13] A. De la Escalera, J. Armingol, and M. Mata, "Traffic sign recognition and analysis for intelligent vehicles," *Image and vision computing*, vol. 21, no. 3, pp. 247–258, 2003.

[14] United Nations Economic Commission for Europe, "Convention on Road Signs And Signals, of 1968," 2006.

[15] State of California, Department of Transportation, "California Manual on Uniform Traffic Control Devices for Streets and Highways."

[16] A. Vázquez-Reina, S. Lafuente-Arroyo, P. Siegmann, S. Maldonado-Bascón, and F. Acevedo-Rodríguez, "Traffic sign shape classification based on correlation techniques," in *Proceedings of the 5th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*. World Scientific and Engineering Academy and Society (WSEAS), 2005, pp. 149–154.

[17] X. Gao, L. Podladchikova, D. Shaposhnikov, K. Hong, and N. Shevtsova, "Recognition of traffic signs based on their colour and shape features extracted using human vision models," *Journal of Visual Communication and Image Representation*, vol. 17, no. 4, pp. 675–685, 2006.

[18] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," in *Applications of Computer Vision (WACV), 2009 Workshop on*. Ieee, 2009, pp. 1–8.

[19] Y. Xie, L.-F. Liu, C.-H. Li, and Y.-Y. Qu, "Unifying visual saliency with HOG feature learning for traffic sign detection," in *Intelligent Vehicles Symposium, 2009 IEEE*, june 2009, pp. 24–29.

[20] A. Ruta, F. Porikli, S. Watanabe, and Y. Li, "In-vehicle camera traffic sign detection and recognition," *Machine Vision and Applications*, vol. 22, pp. 359–375, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00138-009-0231-x

[21] R. Kastner, T. Michalke, T. Burbach, J. Fritsch, and C. Goerick, "Attention-based traffic sign recognition with an array of weak classifiers," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, june 2010, pp. 333–339.

[22] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1453–1460. [Online]. Available: http://benchmark.ini.rub.de/?section=gtsrb

[23] ——, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[24] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view Traffic Sign Detection, Recognition, and 3D Localisation," *Machine Vision and Applications*, pp. 1–15, December 2011. [Online]. Available: http://dx.doi.org/10.1007/s00138-011-0391-3

[25] F. Larsson and M. Felsberg, "Using fourier descriptors and spatial models for traffic sign recognition," *Image Analysis*, pp. 238–249, 2011.

[26] C. Grigorescu and N. Petkov, "Distance sets for shape filters and shape recognition," *Image Processing, IEEE Transactions on*, vol. 12, no. 10, pp. 1274–1286, 2003.

[27] R. Belaroussi, P. Foucher, J. Tarel, B. Soheilian, P. Charbonnier, and N. Paparoditis, "Road sign detection in images: A case study," in *Proceedings of International Conference on Pattern Recognition (ICPR10), Istanbul, Turkey*, 2010.

[28] R. Rajesh, K. Rajeev, K. Suchithra, V. Lekhesh, V. Gopakumar, and N. Ragesh, "Coherence vector of Oriented Gradients for traffic sign recognition using Neural Networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 907–910.

[29] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1918–1921.

[30] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using K-d trees and Random Forests," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2151–2155.

[31] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale Convolutional Networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2809–2813.

[32] F. Boi and L. Gagliardini, "A Support Vector Machines network for traffic sign recognition," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2210–2216.

[33] P. Gil Jiménez, S. Bascón, H. Moreno, S. Arroyo, and F. Ferreras, "Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies," *Signal Processing*, vol. 88, no. 12, pp. 2943–2955, 2008.

[34] A. Ruta, Y. Li, and X. Liu, "Real-time traffic sign recognition from video by class-specific discriminative features," *Pattern Recognition*, vol. 43, no. 1, pp. 416–430, 2010.

[35] G. Loy and N. Barnes, "Fast shape-based road sign detection for a driver assistance system," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2004, pp. 70–75.

[36] B. Hoferlin and K. Zimmermann, "Towards reliable traffic sign recognition," in *Intelligent Vehicles Symposium, 2009 IEEE*, june 2009, pp. 324–329.

[37] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, "Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 1, pp. 113–126, march 2009.

[38] Y. Gu, T. Yendo, M. Tehrani, T. Fujii, and M. Tanimoto, "Traffic sign detection in dual-focal active camera system," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 1054–1059.

[39] G. Overett and L. Petersson, "Large scale sign detection using HOG feature variants," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 326–331.

[40] C. Keller, C. Sprunk, C. Bahlmann, J. Giebel, and G. Baratoff, "Real-time recognition of U.S. speed signs," in *Intelligent Vehicles Symposium, 2008 IEEE*, june 2008, pp. 518–523.

[41] W.-J. Kuo and C.-C. Lin, "Two-Stage Road Sign Detection and Recognition," in *Multimedia and Expo, 2007 IEEE International Conference on*, july 2007, pp. 1427–1430.

[42] Y.-Y. Nguwi and A. Kouzani, "Detection and classification of road signs in natural environments," *Neural Computing & Applications*, vol. 17, pp. 265–289, 2008. [Online]. Available: http://dx.doi.org/10.1007/s00521-007-0120-z

[43] F. Ren, J. Huang, R. Jiang, and R. Klette, "General traffic sign recognition by feature matching," in *Image and Vision Computing New Zealand, 2009. IVCNZ '09. 24th International Conference*, nov. 2009, pp. 409–414.

[44] S. Xu, "Robust traffic sign shape recognition using geometric matching," *Intelligent Transport Systems, IET*, vol. 3, no. 1, pp. 10–18, march 2009.

[45] H.-H. Chiang, Y.-L. Chen, W.-Q. Wang, and T.-T. Lee, "Road speed sign recognition using edge-voting principle and learning vector quantization network," in *Computer Symposium (ICS), 2010 International*, dec. 2010, pp. 246–251.

[46] X. Qingsong, S. Juan, and L. Tiantian, "A detection and recognition method for prohibition traffic signs," in *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, april 2010, pp. 583–586.

[47] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 2, pp. 264–278, 2007.

[48] S. Lafuente-Arroyo, S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras, and R. J. López-Sastre, "A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems," *Expert Syst. Appl.*, vol. 37, pp. 767–773, January 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1628324.1628558

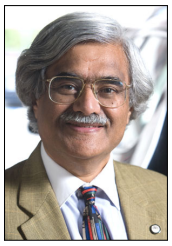
[49] H. Liu, D. Liu, and J. Xin, "Real-time recognition of road traffic sign in motion image based on genetic algorithm," in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. 83–86.

[50] X. Gao, K. Hong, P. Passmore, L. Podladchikova, and D. Shaposhnikov, "Colour vision model-based approach for segmentation of traffic signs," *J. Image Video Process.*, vol. 2008, pp. 6:1–6:7, January 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/386705

[51] V. Prisacariu, R. Timofte, K. Zimmermann, I. Reid, and L. Van Gool, "Integrating Object Detection with 3D Tracking Towards a Better Driver Assistance System," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, aug. 2010, pp. 3344–3347.

[52] D. Deguchi, M. Shirasuna, K. Doman, I. Ide, and H. Murase, "Intelligent traffic sign detector: Adaptive learning based on online gathering of training samples," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 72–77.
[53] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.
[54] N. Barnes and G. Loy, "Real-time regular polygonal sign detection," in *Field and Service Robotics*. Springer, 2006, pp. 55–66.
[55] N. Barnes, A. Zelinsky, and L. Fletcher, "Real-Time Speed Sign Detection Using the Radial Symmetry Detector," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 2, pp. 322–332, june 2008.
[56] C. Nunn, A. Kummert, and S. Muller-Schneiders, "A two stage detection module for traffic signs," in *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, sept. 2008, pp. 248–252.
[57] M. Meuter, C. Nunn, S. M. Gormer, S. Muller-Schneiders, and A. Kummert, "A Decision Fusion and Reasoning Module for a Traffic Sign Recognition System," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1126–1134, Dec. 2011.
[58] M. A. Garcia-Garrido, M. Ocana, D. F. Llorca, M. A. Sotelo, E. Arroyo, and A. Llamazares, "Robust traffic signs detection by means of vision and V2I communications," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, oct. 2011, pp. 1003–1008.
[59] A. Gonzalez, M. Garrido, D. Llorca, M. Gavilan, J. Fernandez, P. Alcantarilla, I. Parra, F. Herranz, L. Bergasa, M. Sotelo, and P. Revenga de Toro, "Automatic Traffic Signs and Panels Inspection System Using Computer Vision," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 485–499, june 2011.
[60] F. Moutarde, A. Bargeton, A. Herbin, and L. Chanussot, "Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system," in *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 2007, pp. 1122–1126.
[61] R. Belaroussi and J.-P. Tarel, "Angle vertex and bisector geometric model for triangular road sign detection," in *Applications of Computer Vision (WACV), 2009 Workshop on*, dec. 2009, pp. 1–7.
[62] A. Ruta, Y. Li, and X. Liu, "Towards real-time traffic sign recognition by class-specific discriminative features," in *Proc. of the 18th British Machine Vision Conference*, vol. 1, 2007, pp. 399–408.
[63] B. Alefs, G. Eschemann, H. Ramoser, and C. Beleznai, "Road Sign Detection from Edge Orientation Histograms," in *Intelligent Vehicles Symposium, 2007 IEEE*, june 2007, pp. 993–998.
[64] N. Pettersson, L. Petersson, and L. Andersson, "The histogram feature - a resource-efficient Weak Classifier," in *Intelligent Vehicles Symposium, 2008 IEEE*, june 2008, pp. 678–683.
[65] I. Creusen, R. Wijnhoven, E. Herbschleb, and P. de With, "Color exploitation in hog-based traffic sign detection," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 2669–2672.
[66] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 255–260.
[67] G. Loy and A. Zelinsky, "Fast radial symmetry for detecting points of interest," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 8, pp. 959–973, aug. 2003.
[68] N. Barnes and A. Zelinsky, "Real-time radial symmetry for speed sign detection," in *Intelligent Vehicles Symposium, 2004 IEEE*, june 2004, pp. 566–571.
[69] A. Doshi and M. Trivedi, "Satellite imagery based adaptive background models and shadow suppression," *Signal, Image and Video Processing*, vol. 1, no. 2, pp. 119–132, 2007.

Andreas Møgelmose received his Bachelor in Computer Engineering on the topic of Information Processing Systems in 2010 from Aalborg University, Denmark. He is pursuing a Master's Degree in Vision, Graphics, and Interactive Systems, also at Aalborg University. Currently, he is a visiting scholar at the University of California, San Diego in the Computer Vision and Robotics Reasearch Laboratory. His main interests are computer vision and machine learning.



Mohan Manubhai Trivedi is a Professor of Electrical and Computer Engineering and the Founding Director of the Computer Vision and Robotics Research Laboratory and Laboratory for Intelligent and Safe Automobiles (LISA) at the University of California, San Diego. He and his team are currently pursuing research in machine and human perception, machine learning, human-centered multimodal interfaces, intelligent transportation, driver assistance and active safety systems. He was the Program Chair of the IEEE Intelligent Vehicles Symposium in 2006 and General Chair of the IEEE IV Symposium in 2010. He is an Editor of the IEEE Transactions on Intelligent Transportation Systems and Image and Vision Computing Journal. He is elected to the Board of the ember of the IEEE ITS Society. He served on the Executive Committee of the California Institute for Telecommunication and Information Technologies as the leader of the Intelligent Transportation Layer at UCSD and as a charter member and Vice Chair of the Executive Committee of the University of California System wide Digital Media Initiative. Trivedi serves as a consultant to industry and government agencies in the U.S. and abroad, including the National Academies, major auto manufactures and research initiatives in Asia and Europe. Trivedi is a Fellow of the IEEE ("for contributions to Intelligent Transportation Systems field"), Fellow of the IAPR ("for contributions to vision systems for situational awareness and human-centered vehicle safety"), and Fellow of the SPIE ("for distinguished contributions to the field of optical engineering").

Thomas B. Moeslund Associate Professor Thomas B. Moeslund is head of the Visual Analysis of People Laboratory at Aalborg University: www.vap.aau.dk. His research includes all aspects of computer vision, with a special focus on automatic analysis of people. He has been involved in ten national and international research projects, both as coordinator, WP leader and researcher. He has published close to 100 peer reviewed papers (citations: 2655. H-index: 16. Source: Harzing 24/1-2012). He serves as associate editor and editorial board member for four international journals, and has co-chaired four international workshops/tutorials and acted as PC member/reviewer for a number of conferences and workshops. Awards include a most cited paper award in 2009, a best IEEE paper award in 2010, and a teacher of the year award in 2010.

TABLE III
OVERVIEW OF DETECTION METHODS IN 41 RECENT PAPERS. PAPERS WITH THE SAME BACKGROUND COLOR ARE PAPERS WRITTEN BY THE SAME GROUP. WHITE BACKGROUND INDICATE STAND-ALONE PAPERS.

Paper	Year	Author group	Segmentation method	Features	Detection method
[16]	2005	1	HSI thresholding with addition for white signs ([49])	Boundary distance transform	Correlation with model distance transforms
[47]	2007	1	HSI thresholding with addition for white signs ([49])	DtB (distance to bounding box)	Linear SVM
[33]	2008	1	HSI thresholding with addition for white signs ([49])	FFT of shape signatures	Euclidian nearest neighbor
[48]	2010	1	HSI thresholding with addition for white signs ([49])	DtB (distance to bounding box)	Linear SVM
[66]	2005	2	None	Haar wavelet features computed on specific color channels	Cascaded classifier
[40]	2008	2	Extended radial symmetry voting	Haar wavelet features	Cascaded classifier
[17]	2006	3	LCH thresholding (obtained with CIECAM97)	HOG	Comparison with template vectors
[50]	2008	3	LCH thresholding (obtained with CIECAM97)	None	None
[62]	2007	4	None	Color filtered edges	Extended radial symmetry voting
[34]	2010	4	HSV discretization	Edges	Extended radial symmetry voting
[20]	2011	4	Quad-tree color selection	Edges	Extended radial symmetry voting
[35]	2004	5	None	Edges	Extended radial symmetry voting
[54]	2006	5	None	Edges	Extended radial symmetry voting
[55]	2008	5	None	Edges	Radial symmetry voting
[56]	2008	6	None	Edges	Votes for symmetric areas to be used as ROI with another shape-detector
[57]	2011	6	None	Edges	Two-tier radial symmetry voting
[58]	2011	7	None	Edges of closed contours with certain aspect ratios	Hough shape detection
[59]	2011	7	None	Edges of closed contours with certain aspect ratios	Hough shape detection
[18]	2009	8	Adaptive RGB threshold	Edges	Fuzzy templates (a Hough derivative)
[51]	2010	8	Adaptive RGB threshold	Edges and Haar-like features	Fuzzy templates, cascaded classifier, and SVM
[64]	2008	9	None	HistFeat (HOG derived)	Cascaded classifier
[39]	2011	9	None	Various HOG-features	5 stage cascaded classifier trained with LogitBoost
[49]	2002	None	HSI thresholding with edge detection and removal of achromatic colors	Edges	Genetic algorithm looking for circles
[63]	2007	None	None	Edge orientation histograms	Comparison with template vectors
[41]	2007	None	HSI thresholding	Edges	Hough shape detection
[60]	2007	None	None	Edges	Hough transform for circular signs, proprietary (not described) for rectangular
[42]	2008	None	HSI thresholding	30x30 px YcbCr patches	Neural network
[37]	2009	None	None	Dissociated dipoles	Cascaded classifier
[61]	2009	None	None	Edges	Vertex and Bisector transform (VBT)
[36]	2009	None	Radial symmetry voting combined with SIFT features	Edge colors	Contracting Curve Density
[43]	2009	None	HSV thresholding	Edges	Hough shape detection
[19]	2009	None	Saliency detection with color and edges	HOG	SVM
[44]	2009	None	Hue thresholding on chromatic colors only	Tangent function of simplified contours	Distance from model tangent function
[45]	2010	None	HSI thresholding	Edges	Circle center voting
[65]	2010	None	None	HOG augmented with color information	SVM
[21]	2010	None	Biologically inspired attention model	Color, corner positions, height, excentricity	Color, corner positions, height, excentricity
[46]	2010	None	HSI thresholding	Edges	Radial symmetry voting
[52]	2011	None	Nested cascade classifier with Local Rank Pattern features (based on 7 RGB based colors)	Edges	RANSAC circle fit
[38]	2011	None	Radial symmetry voting	Colors in modified RGB-space	Distance to learned colors
[8]	2011	None	Probabilistic color preprocessing	Edges	Hough derivative shape detector
[25]	2011	None	None	Fourier descriptors	Correlation based matching

TABLE IV
OVERVIEW OF DETAILED PROPERTIES OF THE 27 PAPERS WHICH DO NOT USE TRACKING

Paper	Year	Author group	Sign type in paper	Sign type possible	Real-time	Rotation inv.	Model vs. training	Test image type	
[16]	2005	1	5 regular polygons, various colors	Colored	N/A	Yes	Both	High-res	
[33]	2008	1	Circular, triangular, square and semiellipses	Colored	N/A	Yes	Both	Low-res	
[48]	2010	1	Circular red	Circular colored	No	No	Both	Low-res	
[17]	2006	3	Circular red, circular blue and triangular red	Colored	No	Yes	Both	High-res	
[50]	2008	3	Circular red and blue	Colored	N/A	Yes	Training	High-res	
[62]	2007	4	Circular, triangular and square, various colors	Regular, colored polygons	Yes	No	Both	N/A	
[35]	2004	5	Regular polygons	Regular polygons	Yes	Yes	Model	Low-res	
[54]	2006	5	Regular polygons	Regular polygons	Yes	No	Model	Low-res	
[55]	2008	5	Circular red	Circular	Yes	Yes	Model	Low-res	
[18]	2009	8	Circular red, circular blue, diamond white	Colored	No	No	Both	High-res	
[64]	2008	9	Circular, triangular and octagonal red	Any sign	Yes	No	Training	Low-res	
[39]	2011	9	Circular red	Any sign	Yes	No	Training	Low-res	
[49]	2002	None	Circular, red	Colored	N/A	Yes	N/A	N/A	
[63]	2007	None	Circular, triangular, diamond, octagonal	Any sign	Yes	No	Training	Low-res	
[41]	2007	None	Circular and triangular, red	Colored	N/A	Yes	N/A	N/A	
[42]	2008	None	Circular red, triangular red, and octagonal red	Colored	No	No	Training	N/A	
[37]	2009	None	Circular and triangular, red	Any sign	No	No	Training	N/A	
[61]	2009	None	Triangular red and blue	Triangular	Yes	Yes	Model	Low-res	
[43]	2009	None	Circular, triangular, square, various colors	Colored	No	Yes	Model	Low-res	
[19]	2009	None	Circular, red and square blue	Colored	N/A	Yes	Training	Low-res	
[44]	2009	None	Circular, triangular, square, various colors	Colored	N/A	No	Both	Low-res	
[45]	2010	None	Circular red	Colored	No	No	Model	Low-res	
[65]	2010	None	Circular red, circular blue and triangular red	Any sign	N/A	No	Training	High-res	
[21]	2010	None	Triangular and octagonal, red	Any sign	Yes	N/A	Model	N/A	
[46]	2010	None	Circular red	Colored	N/A	No	Both	Low-res	
[8]	2011	None	Circular and triangular, red and blue	Regular, colored polygons	No	No	Both	High-res	
[25]	2011	None	7 sign types	Any sign	N/A	Yes	Model	Low-res	No

TABLE V
OVERVIEW OF DETAILED PROPERTIES OF THE 14 PAPERS WHICH USE TRACKING

Paper	Year	Author group	Sign type in paper	Sign type possible	Real-time	Rotation inv.	Model vs. training	Test image type
[47]	2007	1	Circular and triangular, red	Colored	No	No	Both	Low-res
[66]	2005	2	Circular red	Any sign	Yes	No	Training	Low-res
[40]	2008	2	Rectangular white	Any sign	Yes	No	Both	Low-res
[34]	2010	4	40 different signs	Any sign	Yes	No	Model	Low-res
[20]	2011	4	Circular red and blue	Regular, colored polygons	Yes	No	Both	Low-res
[56]	2008	6	Circular, triangular and octagonal	Any sign	Yes	No	Model	N/A
[57]	2011	6	Circular	Circular	Yes	Yes	Model	Low-res
[58]	2011	7	100 different signs, circular and triangular	Any sign	Yes	Yes	Model	Low-res
[59]	2011	7	8 sign categories	Any sign	No	Yes	Model	Low-res
[51]	2010	8	N/A	Colored	Yes	N/A	Both	Low-res
[60]	2007	None	Circular red and rectangular white	Circular and rectangular	N/A	N/A	Model	Low-res
[36]	2009	None	Circular red	Circular	N/A	Yes	Both	Low-res
[52]	2011	None	Circular red	Colored	No	Yes	Training	Low-res
[38]	2011	None	Circular red and blue	Circular colored	Yes	Yes	Both	Low-res

TABLE VI

OVERVIEW OF THE PERFORMANCE OF THE PAPERS INCLUDED IN THIS SURVEY. FOR THOSE PAPERS WHERE THE NUMBERS ARE AVAILABLE, THE BEST AND MEAN DETECTION RATE IS PRESENTED, ALONG WITH THE CORRESPONDING FALSE POSITIVE MEASURE. NOTE THAT THE SYSTEMS HAVE ALL BEEN TESTED IN DIFFERENT WAYS, SO A DIRECT COMPARISON IS NOT FEASIBLE. SEE SECTION IV FOR FURTHER DETAILS.

Paper	Year	Group	Evaluation data format	Pos/neg in evaluation data	Best detection rate	False positives for best detection	Mean detection rate	Mean false positives
[16]	2005	1	No statistical results given					
[47]	2007	1	5176 images from 5 videos containing 104 signs	N/A	N/A	N/A	N/A	N/A
[33]	2008	1	2000 synthetic images	N/A	100%	WPA: 0.74%	89.08%	WPA: 13.17%
[48]	2010	1	No statistical results given					
[66]	2005	2	Images from videos	1700 pos/40000 neg	98.6%	FPR: 0.03%	-	-
[40]	2008	2	16828 images from videos	80 positives	98.75%	FPPF: 0.062	-	-
[17]	2006	3	Images from videos	98 positives	95%	N/A	-	-
[50]	2008	3	128 images	142 positives	94%	PPV: 23%	89.67%	PPV: 26%
[62]	2007	4	No results given for the detection stage only					
[34]	2010	4	Images from videos containing 210 signs	N/A	100%	N/A	92.9%	N/A
[20]	2011	4	No statistical results given (graphs are available in the paper)					
[35]	2004	5	45 images	49 positives	100%	FPPF: 0.67	96.67%	FPPF: 0.56
[54]	2006	5	47 images from 1 video	47 positives	93.62%	FPPF: 2.26	-	-
[55]	2008	5	Images from videos	N/A	93%	FPPF: 0.5	-	-
[56]	2008	6	No statistical results given (graphs are available in the paper)					
[57]	2011	6	Images from 34 videos containing more than 100 signs	N/A	87.12%	FPR: 0.14%	-	-
[58]	2011	7	30000 images from 1 video	340 positives	97.74%	FPPF: 0.0024	96.45%	FPPF: 0.0014
[59]	2011	7	Images from videos containing 500 signs	N/A	99.96%	N/A	99.52%	N/A
[18]	2009	8	7356 images containing 269 signs	2459 positives	95.7%	FPPF: 2.5	-	-
[51]	2010	8	No statistical results given (see [18] instead)					
[64]	2008	9	No statistical results given (graphs are available in the paper)					
[39]	2011	9	Images from videos	21500 pos/40000 neg	98.68%	FPR: 10^{-8}%	-	-
[49]	2002	None	No statistical results given					
[63]	2007	None	Video tracks, 10-200 frames in length	105 positives	81.9%	N/A	-	-
[41]	2007	None	No results given for the detection stage only					
[60]	2007	None	Images from videos containing 281 signs	N/A	88.97%	FPPF: 0	-	-
[42]	2008	None	164 images	164 positives	92.45%	N/A	-	-
[37]	2009	None	4755 images from 4 videos	N/A	97%	FPPF: 0.056	92%	FPPF: 0.048
[61]	2009	None	48 images	40 positives	82.5%	FPPF: 0.042	-	-
[36]	2009	None	Images from 1 30 min. video containing 94 signs	N/A	100%	N/A	-	- △
[43]	2009	None	Images from videos containing 20 signs	N/A	N/A	N/A	-	-
[19]	2009	None	More than 500 images from videos	N/A	99.16%	FPR: 5.56%	98.3%	FPR: 4.72%
[44]	2009	None	1000 images	N/A	95%	FPTP: 0%	91.8%	FPTP: 0.9%
[45]	2010	None	Images from videos	397 pos/697 neg	89.42%	FPR: 0.05%	-	-
[65]	2010	None	3000 images	N/A	85%	N/A	72.47%	N/A
[21]	2010	None	820 images from 2 videos	117 positives	89.8%	PPV: 98.3%	-	-
[46]	2010	None	85 images from video	95 positives	76.64%	FPPF: 0.094	-	-
[52]	2011	None	2967 images from videos	4886 positives	90.1%	PPV: 85.6%	-	-
[38]	2011	None	2134 images from videos	3298 positives	94.03%	FPPF: 3.41	-	-
[8]	2011	None	Comparison of different methods, thus no final result to report.					
[25]	2011	None	STS dataset	641 positives	95.33%	PPV: 100%	77.08%	PPV: 91.85%

Traffic Sign Detection and Analysis: Recent Studies and Emerging Trends

Andreas Møgelmose, Mohan M. Trivedi, and Thomas B. Moeslund

***Abstract*— Traffic sign recognition (TSR) is a research field that has seen much activity in the recent decade. This paper introduces the problem and presents 4 recent papers on traffic sign detection and 4 recent papers on traffic sign classification. It attempts to extract recent trends in the field and touch upon unexplored areas, especially the lack of research into integrating TSR with a driver-in-the-loop system and some of the problems that presents. TSR is an exciting field with great promises for integration in driver assistance systems and that particular area deserves to be explored further.**

I. Introduction

Traffic Sign Recognition (TSR) has seen much work in the past decade. With the emergence of increasingly complex Driver Assistance Systems (DAS), such as adaptive cruise control, including some sort of TSR for driver support has become a logical next step for inclusion in top-of-the-line cars. Some cars already come equipped with TSR for speed limit detection, but there are obviously many other signs that would be interesting to recognize from a DAS perspective.

The recent research in the field has been focused on the narrow vision-problem of detection, classification, and - to some extent - tracking of signs in images. For true integration in DAS, a TSR system should rather been looked upon as a driver-in-the-loop system where the driver is an integral part, as described in [1], [2], [3]. By also monitoring the driver, the system can tailor its output to specific situations. Furthermore, research indicates [4] that people are better at perceiving some signs than others, something that a TSR system could also benefit from taking into account to make sure that only relevant information is presented to the driver. There is not point in presenting a sign that the driver has already noticed.

TSR systems are traditionally split into a detection stage and a classification stage. The detection stage takes care of finding signs, while the classification stage figures out what a particular sign means. This paper describes each stage separately. It is possible to add a third stage that does tracking of the detected signs. The structure can be seen in fig. 1. The purpose of this paper is not to be a complete survey, but to highlight trends in the TSR research by using some recent prominent papers as examples. The next section describes traffic signs along with some of the challenges and problem in detecting and recognizing them. After that is sections on how selected recent papers do the detection, classification, and tracking, respectively. That is followed up by a discussion of future directions in which the recent trends are examined and new or under-developed research areas are described.

A. Møgelmose is a master's student in Vision, Graphics, and Interactive Systems at Aalborg University, Denmark. andreasm@es.aau.dk

M. M. Trivedi is head of the CVRR lab at University of California, San Diego mtrivedi@ucsd.edu

T. B. Moeslund is head of the VAP Lab at Aalborg University, Denmark tbm@create.aau.dk

II. On traffic signs

Traffic signs have the purpose of guiding people through the traffic in a safe manner. They are defined through laws, so the TSR task is quite well-defined. It is still, however, a complicated multi-class detection and classification problem, in some cases with extremely low intra-class variance.

The designs of traffic signs are standardized through laws, but differ across the world. In Europe many signs are standardized via the Vienna Convention on Road Signs And Signals [5]. There, shapes are used to categorize different types of signs: Circular signs are prohibitions including speed limits, triangular signs are warnings and rectangular signs are used for recommendations or sub-signs in conjunction with one of the standard shapes. In addition to these, octagonal signs are used to signal a full stop, downwards pointing triangles yield and countries have different other types, e.g. to inform about city limits. Examples of these signs can be seen in fig. 2.

In the US, traffic signs are regulated by the Manual on Uniform Traffic Control Devices (MUTCD) [6]. It defines which signs exist and how they should be used. It is accompanied by the Standard Highway Signs and Markings (SHSM) book, which describes the exact designs and measurements of signs. At the time of writing, the most recent MUTCD was from 2009, while the SHSM book has not been updated

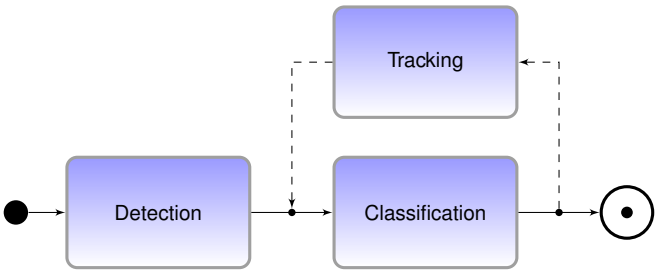


Fig. 1. The basic flow in most TSR systems.

since 2004, and thus it describes the MUTCD from 2003. An updated version of the SHSM should be on its way. The MUTCD contains a few hundred different signs, divided into 13 categories. US signs are white rectangles for regulatory signs, yellow diamonds for warnings, downwards pointing triangles for yield and octagons for full stop. Examples of American signs can be seen in fig. 3.

The Vienna Convention and the US MUTCD are the main standards. Most other countries use standards that are close to one of them, or a combination of the two. While signs seem to be well defined in many cases, the TSR task is made more difficult by a number of challenges. The signs may not be placed properly, so they are not perpendicular to the road, colors may be off due to wear or lighting conditions, they may be occluded by trees, poles, or other cars. Many signs, such as speed limit signs with different limits, are very similar to each other, making the classification task complicated.

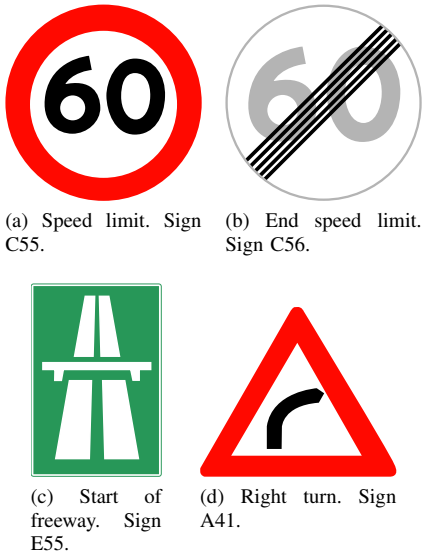
(a) Stop. Sign R1-1.
(b) Yield. Sign R1-2.
(c) Speed limit. Sign R2-1.
(d) Turn warning with speed recommendation. Sign W1-2a.

Fig. 3. Examples of signs from the US national MUTCD. Image source: [6]

III. Detection

As mentioned, the purpose of the detection stage is to find sign and pass them on to a classifier. It is common to treat detection and classification as two separate steps, but the interface between them is not standardized. Some classifiers rely on the detector to provide information on not only the center of the sign, but also its size, shape or overall sign type (e.g. regulatory sign vs. warning sign). Very often the attributes that determine the sign type - commonly shape and color - are also attributes the detector use, so this information is directly available.

Traditionally [12], [13], sign detectors have been classified as being either color-based or shape-based. Color-based detectors would find signs based on their distinctive background- or border-color, whereas shape-based detectors would ignore color-information completely and find sign-shapes instead. This classification of detectors seem a bit outdated, since all color detectors also use shape information for further filtering. Champions of shape-based methods argue that color-detection is unreliable due to changes in lighting and sign wear. However, similar arguments can be put forth against shape-based detectors: Signs can be partly occluded or they may be rotated or otherwise distorted so their shapes look different, something not all shape based detectors can handle.



(a) Speed limit. Sign C55.
(b) End speed limit. Sign C56.
(c) Start of freeway. Sign E55.
(d) Right turn. Sign A41.

Fig. 2. Examples of European signs. These are Danish, but many countries use similar signs.

A better way to look at detectors is by splitting them into three blocks: Segmentation, feature extraction, and detection. Classification is not covered here, as that second part of the system is described in section IV. Almost all detection algorithms can be split into these blocks, making comparison across systems easy. Segmentation is usually color-based, but it may also be shape-based. It is the act of narrowing down the search to areas that are likely to contain signs. When that is done, features can be extracted from their areas. The choice of features is usually made in combination with the choice of the detector, since they work in unison to determine the actual signs.

In this paper, we have chosen to cover 4 recent leading papers [7], [9], [10], [11] that describe different methods of detecting signs. They were selected as recent trend-setting papers from the pool shown in table **??**. These papers, apart from being very recent, cover trends in the area well: Some use theoretical sign models, some use learned models, some are mainly color-based, some rely more on shapes, some have extensive focus on tracking. This means that they cover most directions in the field. An overview of the selected papers can be seen in table I. Each of the following subsections cover their methods used for the three blocks: Segmentation, feature extraction, and detection.

TABLE I
OVERVIEW OF DETECTION METHODS IN 4 RECENT PAPERS.

Paper	Year	Segmentation method	Features	Detection method
[7]	2010	HSI thresholding with addition for white signs ([8])	DtB (distance to bounding box)	Linear SVM
[9]	2011	Quad-tree color selection	Edges	Extended radial symmetry voting
[10]	2011	None	Various HOG-features	5 stage cascaded classifier trained with LogitBoost
[11]	2010	Biologically inspired attention model	Color, corner positions, height, eccentricity	Color, corner positions, height, eccentricity

A. Segmentation

[9] opts to use a color based segmentation. They propose a quad-tree attention operator. First step is a filtering that amplifies red and blue colors, the colors of the signs that the system is intended to work with. Then they compute a gradient magnitude map for each of the colors, and their corresponding integral images. Now, the image is evaluated for whether it contains a total color gradient over a certain threshold. If it does not, there is simply not enough colored edges in the sign to constitute any signs. If it does, the image is now split into four quarters, and the same check is done for each quarter. This process continues until a region goes below the threshold, or the minimum region size is reached. Adjacent regions that reach the minimum size while still containing enough gradients are clustered and constitute a sign candidate.

In [7], they follow the method described in their earlier paper, [14], and segment with a thresholding in the HSI (Hue, Saturation, Intensity) color space. It is argued that the HSI space is more robust to changes in lighting than the regular RGB (Red, Green, Blue) color space. They do, however add a method (originally pioneered by [8]), that finds achromatic colors and use this to find white signs. After the segmentation, image pixels that belong to the same color are grouped together.

[11] use a biologically inspired segmentation algorithm, which attempts to find areas in the image that are "interesting". They compute an attention map based on various features, such as Difference of Gaussians (DoG), and Gabor filter kernels that mimics the brain of a mammal. This is done in the RGBY space, since that models how an eye works. These features are weighted and result in a map where high value areas are likely to contain signs.

In [10] they simply opt to not do any segmentation or preprocessing, but jump directly into feature extraction and detection.

For more on segmentation, see the great overview and comparison in [15].

B. Feature extraction

The features that must be extracted must be chosen in close connection with the detection method. In [9], they test both an edge based detector and a cascade using Haar-like feature [16], but end up using the edge based one. Thus, their features are simply the image gradients.

The detector in [7] relies on Distance to Bounding box (DtB) features. It is a measure of distances from the edges of an object to its rectangular bounding box. A rectangular sign will have zero distance to its bounding box, while an upwards pointing triangle will have zero distance to the bottom of its bounding box, but increasing distances when approaching to the upper corners of the bounding box.

To obtain features in [11], they run a color thresholding and then calculate a number of geometric features, such as corner positions, size and eccentricity.

In [10], two different types of Histogram of Oriented Gradient (HOG) features are used. HOG features are, as the name suggests, histograms detailing the orientation of the gradients in an area. Thus, all horizontal lines are binned together, as are vertical lines, etc.

C. Detection

The detection block is where the features for each sign candidate are evaluated and it is determined whether they describe a sign or not. The detection can either be done by matching a theoretical model with the feature (such as deciding whether the candidate looks like a circle), or by matching the features with a learned model of how signs should look in these particular features.

[9], [11] use a theoretical model. In [9], a center-voting scheme based on circles' edges, first presented in [17], is used to find sign candidates. [11] use a template for where corners should be located.

[7], [10] instead use learned classifiers. [7] use a Support Vector Machine (SVM) classifier on the DtB features and [10] use a similar cascaded classifier, trained with LogitBoost.

IV. CLASSIFICATION

Classification is where the meaning of the detected signs are determined. It is a classical computer vision task. Recently, the competition "The German Traffic Sign Recognition Benchmark" (GTSRB) [22] has put renewed focus on the classification. It is a competition with the objective of classifying a number of German (and thus Vienna Convention compliant) signs in no less than 43 classes. The number of classes alone makes this a challenging task. The competition attracted many competitors and spawned four papers [18], [19], [20], [21] from the best competitors. These papers can be said to represent the state-of-the-art in sign classification. An overview can be seen in table II. They achieve very good classification rates for the GTSRB dataset.

TABLE II
OVERVIEW OF CLASSIFICATION METHODS IN THE 4 PAPERS FROM THE GTSRB CONTEST.

Paper	Year	Features	Classification method	Classification rate
[18]	2011	Hue histograms and HOG	Network of SVM classifiers	96.89%
[19]	2011	48x48 pixel color normalized image patches	Convolutional neural network	98.98%
[20]	2011	32x32 pixel image patches in the YUV color space	Convolutional neural network	98.97%
[21]	2011	HOG-features	K-d trees and random forests	97.2%

Unlike the detection task, where some systems employ a theoretical model instead of a learned one, all competitors used a learned classifier. [18] use a network of SVM classifiers. It runs a preprocessing to normalize and enhance colors and calculate the features used: A set of hue histograms and a set of HOG-features. [19] - the winner of the competition - use a Convolutional Neural Network (CNN) and does not extract specific features, but use full 48x48 pixel color normalized image patches. A CNN is inspired by the primary visual cortex [23] and described further in [24], [25]. [20] also use a convolutional network on full image patches, this time resized to 32x32 pixels and converted to the YUV color space. [21] use K-d trees (similar to [26]) with the Best Bin First algorithm described in [27] and random forests on HOG-features.

V. TRACKING

Tracking is the act of following a sign through several frames. Tracking is not used by any of the papers mentioned in the classification section above, since they were simple passed an image of a sign and could leave any tracking to the detector. Detectors, however, can benefit vastly from incorporating a tracking algorithm. Not only can it be used to discard false positives by discarding signs that only appear in a single frame - usually the result of noise - they can also use it to only present new signs to the classifier, enhancing the speed of the system. Furthermore, a sophisticated tracking system can make sure that signs that are temporarily occluded are not reported as new signs when they show up again.

Of the selected papers, only one employ tracking: [9]. It has a sophisticated tracking system based on the changes in appearance of the sign. When detecting a sign, it is assumed to be undistorted. Then a number of random deformations of that particular sign is generated. These distorted views are used to train the tracker on the fly. The motion is learned by fitting these to the sign in following frames using regression. The system is described further in [28], [29].

VI. DISCUSSION AND FUTURE DIRECTIONS

TSR is an area that has seen a lot of contributions recently, and it is an area that is well researched. The main shortcoming is that for detection, no standardized dataset is used, so comparison among papers is hard. One public dataset exist that is suitable for detection: The Swedish Traffic Signs Dataset [30]. It is not yet widely used. That situation was recently remedied for the classification stage, where the GTSRB dataset is a good contribution which is already used in a few papers. Another publicly available dataset for both detection and classification is the KUL Belgium Traffic Signs Dataset and its companion, the KUL Belgium Traffic Sign Classification Benchmark.

The trends seem to be towards more thoroughly tested and compared systems. This effort is spearheaded by the GTSRB, but something similar is needed for detection. It also seems that the trend goes toward learned systems rather than pre-programmed heuristics. Earlier, the common thing has been to create full systems covering both detection and classification, but with the GTSRB, systems has been more modularized and it has become common to create systems that only do classification, something that will make it easier to mix and match approaches to arrive at a system that is fit for a specific application.

However, when looking at TSR in a bigger perspective, much remains to be done. Good detection and classification systems exist, but little work on how to apply TSR in actual systems exist. As mentioned in the introduction, many TSR systems cite driver assistance as their motivation, but simply recognizing signs does not help the driver. In order for TSR to be really applied to driver-in-the-loop systems, it is crucial to take him into account. One option is to look at driver attention: Why present the driver with signs that he has already seen? That will only contribute to information overload. It may also be necessary to pay special attention to signs that drivers are known to simply glance over, as presented in [4].

For a driver-in-the-loop system tracking becomes even more crucial than it already is. As of now, it is mostly used to increase robustness, or not at all. When a driver is present, it is important not to present the same sign to him twice, again to prevent information overload. This means that when a sign is temporarily occluded, it should be handled by tracking so it is not discovered as a new sign when it shows up again. There is also the issue of how to present recognized signs to the driver. In general, the area of really including the driver in TSR systems are virtually unexplored.

VII. CONCLUDING REMARKS

This paper has presented 4 significant recent papers in the area of sign detection and 4 in the area of classification. TSR systems have seen much activity recently, but progress is hampered by the fact that comparison across papers is hard when no standardized dataset for detection exists. Still, very good systems show up, and especially the classification

seems to fare very well. This is helped by the new image database, the GTSRB.

Still, much research remains to be done in the area of applying TSR to DAS. Proper integration of the two is a very promising and exiting task that is in need of much more attention. While many systems perform well in the area when viewed strictly as an object detection or classification task, not much work has been done in applying such systems to driver assistance.

VIII. Acknowledgment

The authors would like to thank our colleagues in the LISA-CVRR lab.

References

[1] M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 1, pp. 108–120, 2007.

[2] M. Trivedi and S. Cheng, "Holistic sensing and active displays for intelligent driver support systems," *Computer*, vol. 40, no. 5, pp. 60–68, 2007.

[3] C. Tran and M. M. Trivedi, "Vision for Driver Assistance: Looking at People in a Vehicle," in *Guide to Visual Analysis of Humans: Looking at People*, T. B. Moeslund, L. Sigal, V. Krueger, and A. Hilton, Eds., 2011.

[4] D. Shinar, *Traffic safety and human behaviour*. Emerald Group Publishing, 2007.

[5] United Nations Economic Commission for Europe, "Convention on Road Signs And Signals, of 1968," 2006.

[6] State of California, Department of Transportation, "California Manual on Uniform Traffic Control Devices for Streets and Highways."

[7] S. Lafuente-Arroyo, S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras, and R. J. López-Sastre, "A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems," *Expert Syst. Appl.*, vol. 37, pp. 767–773, January 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1628324.1628558

[8] H. Liu, D. Liu, and J. Xin, "Real-time recognition of road traffic sign in motion image based on genetic algorithm," in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. 83–86.

[9] A. Ruta, F. Porikli, S. Watanabe, and Y. Li, "In-vehicle camera traffic sign detection and recognition," *Machine Vision and Applications*, vol. 22, pp. 359–375, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00138-009-0231-x

[10] G. Overett and L. Petersson, "Large scale sign detection using HOG feature variants," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, june 2011, pp. 326–331.

[11] R. Kastner, T. Michalke, T. Burbach, J. Fritsch, and C. Goerick, "Attention-based traffic sign recognition with an array of weak classifiers," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, june 2010, pp. 333–339.

[12] M.-Y. Fu and Y.-S. Huang, "A survey of traffic sign recognition," in *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, july 2010, pp. 119–124.

[13] H. Fleyeh and M. Dougherty, "Road and traffic sign detection and recognition," in *10th EWGT Meeting and 16th Mini-EURO Conference*, 2005, pp. 644–653.

[14] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 2, pp. 264–278, 2007.

[15] H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo, "Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 4, pp. 917–930, dec. 2010.

[16] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[17] G. Loy and N. Barnes, "Fast shape-based road sign detection for a driver assistance system," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2004, pp. 70–75.

[18] F. Boi and L. Gagliardini, "A Support Vector Machines network for traffic sign recognition," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2210–2216.

[19] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1918–1921.

[20] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale Convolutional Networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2809–2813.

[21] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using K-d trees and Random Forests," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2151–2155.

[22] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1453–1460. [Online]. Available: http://benchmark.ini.rub.de/?section=gtsrb

[23] D. Hubel and T. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2146–2153.

[26] W.-J. Kuo and C.-C. Lin, "Two-Stage Road Sign Detection and Recognition," in *Multimedia and Expo, 2007 IEEE International Conference on*, july 2007, pp. 1427–1430.

[27] J. Beis and D. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp. 1000–1006.

[28] E. Bayro-Corrochano and J. Ortegón-Aguilar, "Lie algebra approach for tracking and 3D motion estimation using monocular vision," *Image and Vision Computing*, vol. 25, no. 6, pp. 907–921, 2007.

[29] O. Tuzel, F. Porikli, and P. Meer, "Learning on lie groups for invariant detection and tracking," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[30] F. Larsson and M. Felsberg, "Using fourier descriptors and spatial models for traffic sign recognition," *Image Analysis*, pp. 238–249, 2011.

Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets

Andreas Møgelmose
VGIS, AAU & CVRR, UCSD
andreasm@es.aau.dk

Mohan M. Trivedi
CVRR Lab, UC San Diego
mtrivedi@ucsd.edu

Thomas B. Moeslund
VAP Lab, Aalborg University
tbm@create.aau.dk

Abstract

This study compares the performance of sign detection based on synthetic training data to the performance of detection based of real-world training images. Viola-Jones detectors are created for 4 different traffic signs with both synthetic and real data, and varying numbers of training samples. The detectors are tested and compared. The result is, that while others have successfully used synthetic training data in a classification context, it does not seem to be a good solution for detection. Even when the synthetic data covers a large part of the parameter space, it still performs significantly worse than real-world data.

1 Motivation

With the emergence of more advanced sensors embedded in cars, the field of Traffic Sign Recognition (TSR) has seen increasing interest over the last decade. TSR systems can be used in a number of scenarios, ranging from Driver Assistance Systems (DASs) to fully autonomous cars.

Many sign detection systems (see section 2) rely on large amounts of training data to work. Over the past two years, a few traffic sign datasets has shown up: The GTSRB dataset [11, 12], the Swedish Traffic Signs Dataset [7], and the KUL Belgium Traffic Signs Dataset. A commonality among these datasets is that they contain European Signs conforming to the Vienna Convention. Since signs differ from region to region and in many cases from country to country, an interesting proposition is to use synthetically generated training data, saving a lot of time and effort in gathering the data. Synthetic training data has not yet been widely used in the field of TSR, but is worth researching since no US dataset exist. A recent survey [9] shows that research on the detection and recognition of US traffic signs is lacking in general. This paper investigates if using synthetic data for the detection of traffic signs is feasible.

The role and importance of high quality, representative datasets in the development of TSR systems cannot be overemphasized. Collection of such datasets is expensive (in time as well as effort) task. This brings forward the idea of using synthetic data, since signs have a well-defined appearance. The use of synthetic training in sign detection is not yet widespread, prompting this paper. Our paper is focused closely on the generation of synthetic training data for detection purposes. It is also the first of its kind dealing with US signs. In [5, 4], generation of synthetic data specifically for classification is investigated. In [10], some aspects of detecting non-US signs with synthetic data is discussed. The detection task is somewhat harder the classification due to the lack of knowledge about whether a sign is present, where it is, and what size it has.

The following section briefly covers the general workings of TSR systems, followed by a section on how we generate synthetic training data. Towards the end of the paper, the performance of synthetic training data is compared to the performance of real-world training data when used to train a simple AdaBoost cascade with Haar-like features [13].

2 TSR: General approaches

Overviews of TSR can be found in [9, 2, 3]. TSR can be split into two main stages: Sign detection and sign classification, as seen in fig. 1. Not all detection approaches require training as such, since they are using a theoretical model of the sign, based on e.g. the shape. With that said, many papers present Machine Learning (ML) based approaches. In [1], an AdaBoost Cascade similar to the one used in this paper was used, albeit on specific color channels. In [6], the image is segmented with a HSI threshold and then classifies the resulting blobs using a linear Support Vector Machine (SVM) on Distance to Bounding Box (DtB) features. DtB features

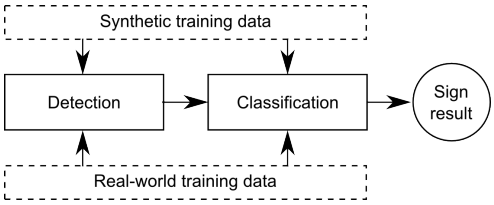
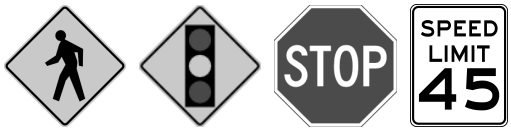


Figure 1: Flow for ML-based TSR-systems. The stages can be trained with synthetic or real-world data, and two stages does not have to be trained with the same type.



(a) Pedestrian crossing (b) Signal ahead (c) Stop (d) Speed limit

Figure 2: Examples of typical US sign templates.

are measurements of the distance between the edge of the blob and its rectangular bounding box.

3 Synthetic training data for detection

The question this paper tries to answer is: Can we substitute real-world training data with synthetic in ML based sign detection systems? The idea is to generate synthetic training images from a drawn template. Template examples can be seen in fig. 2.

The goal is to emulate how signs of the given type might look on pictures from the real world. In order to do this, several transformations are made randomly to the template:

Hue variations emulates faded signs and color casts due to lighting of the natural scene. Done by adding to/subtracting from the hue-parameter in the HSV color space.

Lighting variations emulates shadows and variations in exposure. Done by adding to/subtracting from the value-parameter in the HSV color space.

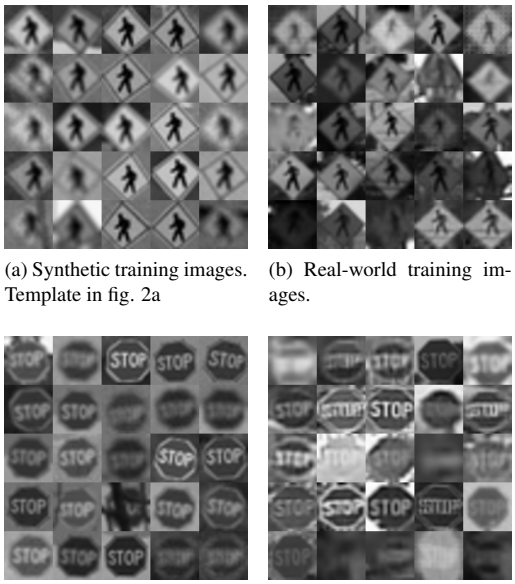
Rotations around the x-, y-, and z-axis with the origin in the center of the template. Emulates signs captured from different perspectives.

Backgrounds taken from a real image are added to the template. This emulates the various backgrounds a sign might have in real life.

Gaussian blur is added to emulate an unfocused camera. It should be noted that Gaussian blur does not really emulate the bokeh produced by an unfocused lens, but emulating bokeh properly is not a straightforward task, and it would likely not give any notable detection benefit.

Gaussian noise to emulate sensor noise.

Occlusions are added in the form of tree branches growing in front of some signs.



(a) Synthetic training images. Template in fig. 2a (b) Real-world training images. (c) Synthetic training images. Template in fig. 2c (d) Real-world training images.

Figure 3: Samples from the training image sets.

Each transformation should be applied with a random parameter within some realistic boundaries. Samples of training images can be seen in fig. 3.

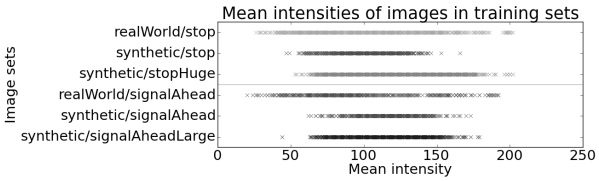
To evaluate whether the synthetic datasets cover the same variance in appearance as the real-world data, we compare the distributions in intensity- and blur-values among training sets. In fig. 4a a plot of the mean of the intensities in the training images is shown. Each point in the plot is a single image. Data for the detectors of two different signs is shown. In a few sets, the intensity span does not match, but the large 5000 image stop sign set is similar to real-world data. Another parameter is shown in fig. 4b: Blur. Blur is calculated as

$$B = \frac{1}{n} \sum_{i=0}^{n} e_i \tag{1}$$

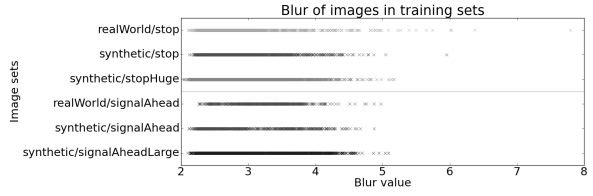
where B is the blur-value, n is the number of vertical edges in an image and e_i is the edge width of a specific edge pixel, given as the distance between the pixels with the local maximum and minimum intensity around the edge pixel. The measure is described further in [8]. This shows that the blur variance is covered well by the synthetic data.

Table 1: Results of the comparative evaluations of detectors

Training type	Training images (positive/negative)	Stages	Signs to find	TP	FP	FN
Stop						
Real-world	1218/2500	20	103	76 (73.8%)	11	27
Real-world	1686/3000	20	103	75 (72.8%)	8	28
Synthetic	1218/2500	17	103	18 (17.5%)	2	85
Synthetic	5000/10000	19	103	26 (25.2%)	5	77
Synthetic	1218/2500	10	103	60 (58.3%)	1500	43
Pedestrian crossing						
Real-world	364/800	20	40	29 (72.5%)	10	11
Real-world	1044/2000	20	40	30 (75%)	2	10
Synthetic	364/800	14	40	11 (27.5%)	28	29
Speed limit 35						
Real-world	253/500	20	21	15 (71.4%)	1	6
Synthetic	253/500	7	21	5 (23.8%)	32	16
Synthetic	2000/4000	7	21	6 (28.6%)	6	15
Signal ahead						
Real-world	597/1500	20	56	42 (75%)	10	14
Real-world	859/2000	20	56	38 (67.9%)	4	18
Synthetic	597/1500	13	56	14 (25%)	117	49
Synthetic	2000/4000	13	56	16 (28.6%)	53	48



(a)



(b)

Figure 4: Distribution of two parameters in the training sets.

4 Comparative evaluation

To compare the synthetic training data to training data obtained from real footage, a simple Viola-Jones based detector [13] was trained for the four sign types illustrated in fig. 2. The choice of detection algorithm is not crucial, as the purpose of this paper is not to find a perfect traffic sign detector, but rather look at the relative differences between detectors trained with synthetic and real-world images. It was trained with an image size of 20x20 pixels in all cases, except for the rectangular speed limit sign, trained with 18x24 pixels.

The detectors created with various numbers of training images was tested on a set of real-world images, collected from cars in conjunction with this lab's research.

With all signs, the real-world data performs significantly better than the synthetic data. Providing more training data in the synthetic case does help, but even a large increase (more than a doubling) of the training data does not make the synthetic data perform comparably to the real-word data. All detectors were trained with a target of 20 stages, but some terminated earlier due to a sufficiently good fit to the training data, and others were lowered to give better detection performance at the cost of more false positives. It is indeed possible for the synthetic detector to find more true signs, but at a huge cost in false positives, and still not as good as the real-world detector.

Even in the cases (like the stop sign detector with 5000/10000 training images) where the synthetic data spans nearly the same space as the real-world detector, the synthetic detector fails to achieve a detection rate anywhere near the real-world data.

5 Concluding remarks

We discussed a research study to assess the feasibility of using carefully synthesized training datasets for developing traffic sign detectors. In this research, output from a synthetic training generator has been used to train a stock AdaBoost cascade and its performance compared with real-world training images. The real-world training data consistently performs significantly better than the synthetic training data, even in cases where the synthetic data seems to span a similar set of appearances. This leads to the conclusion that there is simply no substitute for real-world images in the case of detection.

An ML-approach to setting the synthetic data generation parameters would be a logical place to go from here, if further study of synthetic data for detection is desired. It is also possible that the system could benefit from further transformations to the template image, such as motion blur. Other works have shown promising results in using synthetic training data for classification of signs. An interesting direction of research could be to explore hybrid (real and synthetic) datasets for TSR approaches.

References

[1] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 255–260. IEEE, 2005.

[2] H. Fleyeh and M. Dougherty. Road and traffic sign detection and recognition. In *10th EWGT Meeting and 16th Mini-EURO Conference*, pages 644–653, 2005.

[3] Meng-Yin Fu and Yuan-Shui Huang. A survey of traffic sign recognition. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference on*, pages 119–124, july 2010.

[4] H. Hoessler, C. Wöhler, F. Lindner, and U. Kreßel. Classifier training based on synthetically generated samples. In *Proceedings of 5th international conference on computer vision systems. Bielefeld, Germany*, 2007.

[5] H. Ishida, T. Takahashi, I. Ide, Y. Mekada, and H. Murase. Identification of degraded traffic sign symbols by a generative learning method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 531–534. IEEE, 2006.

[6] S. Lafuente-Arroyo, S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras, and R. J. López-Sastre. A decision support system for the automatic management of keep-clear signs based on support vector machines and geographic information systems. *Expert Syst. Appl.*, 37:767–773, January 2010.

[7] F. Larsson and M. Felsberg. Using fourier descriptors and spatial models for traffic sign recognition. *Image Analysis*, pages 238–249, 2011.

[8] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi. A no-reference perceptual blur metric. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–57–III–60 vol.3, 2002.

[9] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund. Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey. *IEEE Intelligent Transportation Systems Transactions and Magazine*, Special Issue on MLFTSR:–, dec 2012.

[10] G. Overett, L. Tychsen-Smith, L. Petersson, N. Pettersson, and L. Andersson. Creating robust high-throughput traffic sign detectors using centre-surround HOG statistics. *Machine Vision and Applications*, pages 1–14, 2011.

[11] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011.

[12] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[13] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2001.

A Two-stage Part-Based Pedestrian Detection System Using Monocular Vision

Andreas Møgelmose, Antonio Prioletti, Mohan M. Trivedi, Alberto Broggi, and Thomas B. Moeslund

***Abstract*— This paper introduces a part-based two-stage pedestrian detector. The system finds pedestrian candidates with an AdaBoost cascade on Haar-like features. It then verifies each candidate using a part-based HOG-SVM doing first a regression and then a classification based on the estimated function output from the regression. It uses the Histogram of Oriented Gradients (HOG) computed on both the full, upper and lower body of the candidates, and uses these in the final verification. The system has been trained and tested on the INRIA dataset and performs better than similar previous work, which only uses full-body verification.**

I. Introduction

Pedestrian detection is currently a very large research field. It can be used in surveillance, Advanced Driver Assistance Systems (ADAS), and many other places. The ADAS scenario offers plenty of challenges (as summarized in [1]): High variability in appearance among pedestrians, cluttered backgrounds, highly dynamic scenes with both pedestrian and camera motion, and strict requirements in both speed and reliability. Input from a reliable pedestrian detection system can be used to warn the driver about people in front of the car, prepare or even activate a braking maneuver to prevent a collision, or deploy other safety systems such a airbags.

ADAS is a a challenging domain to work within. Braking systems take a short while to apply, and reaction times must be fast for driving, where fractions of a second can be the deciding factor between a collision and a near-miss. At the same time, the system must be robust, so the braking system is not deployed mistakenly (due to a false positive detection), which could itself lead to accidents, or worse, not employ at all (due to a missed detection). Further reasoning than just detection is necessary in such a framework, with pedestrian intent estimation being a good example, as presented in [2] or as another example, automatic breaking as in [3].

This paper introduces a part-based 2-stage detection method, an extension to [4]. It combines the speed of a Haar-based boosted cascade with the low number of false positives from the HOG-SVM detector, bringing it closer to the strict ADAS requirements than any of the two algorithm on their own. We extend it to a part-based solution, which lowers the false positive rate even further.

This paper is structured as follows: In the next section, we describe some of the work related to ours and we provide an overview of our algorithm. In the next sections we describe each stage in the algorithm in detail. Finally, in V, we describe the performance of our algorithm followed by suggestions for future work and a conclusion.

II. General approach and related work

As mentioned, pedestrian detection is a field with much attention from the research community. Even when narrowed to applications in connection with cars and ADAS, a large body of work exist. A classic method of pedestrian detection is a boosted cascade on Haar-like features, first presented by Viola and Jones [5]. It is very fast, but lacks robustness due to the high appearance variability among pedestrians in the real world. Instead, many people turn to the HOG-SVM solution presented by Dalal and Triggs [6]. It is much more robust and generally detect pedestrians in harder situations, while keeping a low number of false positives. Its problem lies in processing speed. As mentioned, the ADAS application requires fast processing, something that is not immediately obtainable with the HOG-SVM detector. The HOG-SVM method was explored for use with infrared images in [7]. For further exploration of pedestrian detectors, we refer the reader to the general survey by Gerónimo et. al. [1] or, for vision-only based systems, Gandhi and Trivedi [8], [9]. The system presented in our paper uses monocular vision as base for the detection. This means that the hardware requirements for the car are low and realistically possible - many cars are already outfitted with a front facing camera for other purposes, such as lane detection. For a survey of monocular vision based methods, see [10].

We combine the speed of the Haar detector with the robustness of a part-based HOG-SVM detector. The base for the method used in this paper was first presented by Geismann and Schneider [4], but is also covered by others in various versions [11], [12]. Apart from using a combination of a Haar-cascade and HOG-SVM, Geismann and Schneider also evaluated using a sparse HOG descriptor to speed up the verification. Part-based pedestrian detection has been presented in various versions before, such as [13], [14], [15].

The properties of the Haar cascade and the HOG-SVM detector makes them prime candidates for combination: The Haar cascade does the initial pass, finding Regions Of Interest (ROIs) that are passed on to the HOG-SVM detector which verifies the initial findings by the Haar cascade. The

A. Møgelmose is a master's student in Vision, Graphics, and Interactive Systems at Aalborg University, Denmark. andreasm@es.aau.dk

A. Prioletti is a master's student at Vislab, University of Parma, Italy. antonio.prioletti@studenti.unipr.it

M. M. Trivedi is head of the CVRR lab at University of California, San Diego mtrivedi@ucsd.edu

A. Broggi is head of Vislab at University of Parma, Italy broggi@vislab.it

T. B. Moeslund is head of the VAP Lab at Aalborg University, Denmark tbm@create.aau.dk

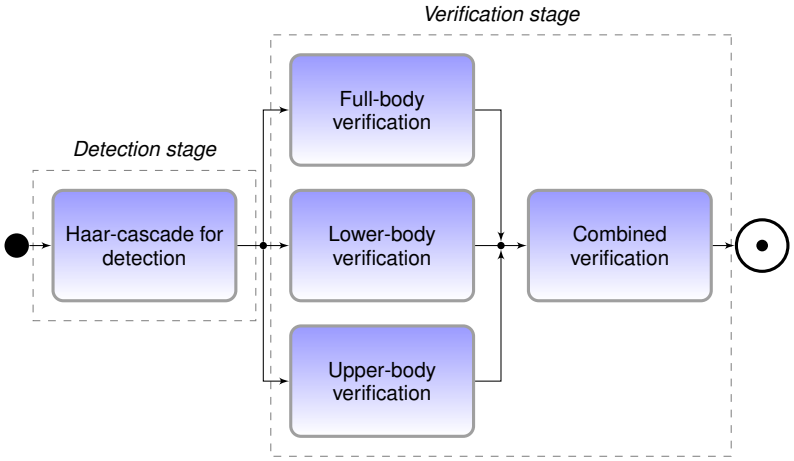


Fig. 1. The flow of the algorithm described in this paper.

first stage is called the *detection stage* and the second the *verification stage*. That is the basics of the approach outlined in [4].

Our goal is to lower the number of false positives without too much penalty in the detection rate. In order to do this, we alter the verification stage to not only verify based on a full body classification, but also a lower body and upper body classifier. We combine these results to figure out whether the ROI contains a person.

The combination of verification results is done in two ways, which are compared: A simple majority vote, requiring at least two of three classifiers to verify the detection, and a more advanced way, which introduces a third stage to the algorithm, classifying each window based on the estimated function value from an SVM regression performed on each part.

An overview of the flow through the algorithm can be seen in fig. 1.

III. Detection stage

The detection stage is an AdaBoost cascade on Haar-features [5]. It works by using AdaBoost to learn a number of weak classifiers, which are combined into strong classifiers. Several layers (called stages) of these strong classifiers are then combined in a cascade to create the final detection. The cascaded structure makes the algorithm very fast, since most candidates are discarded in one of the first stages, thus not having to be calculated in following stages. Only the actual detections have to pass through all stages. The algorithm is described in detail in [5].

Throughout this paper, we work with the INRIA Pedestrian Dataset [6]. Thus, the detection cascade was trained with the training set given therein: 2416 positive images and 12180 negative images. The training images were cropped closely around the annotated persons, because Haar-cascades does not benefit from having as much background included as HOG-based classifier. After the crop the training images were resized to 12x28 pixels.

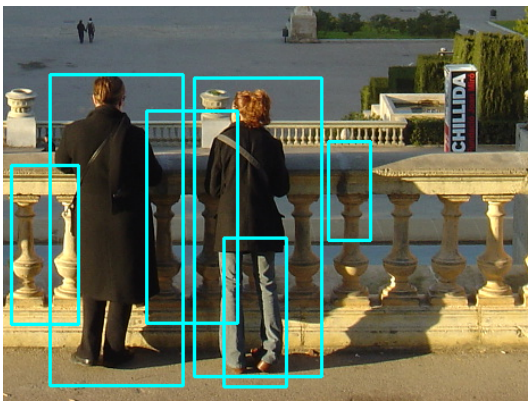


Fig. 2. Example of the output from the detection stage. It is clear that it contains several false positives, but that is desired, since it ensures that also the true positives are included.

The detection stage is set up so that it finds the maximum possible number of pedestrians, which also means that it will return plenty of false positives. A larger number of false positives will slow down the computation, since the verification stage must process more, but it is a worthy trade-off given that the true positive rate of this stage forms the upper bound of detections for the entire system.

The detection stage returns bounding boxes of all the potential pedestrians in the picture, which are sent on to the verification stage. Part based detection in the detection stage is not used, since the data from [16] shows that the Haar cascade generally performs bad in part-based detection schemes. An example of the output of the detection stage can be seen on fig. 2.

IV. Verification stage

The part-based verification stage used in this work differs from the full-body verification stage of Geismann and Schneider's [4]. We use a part-based detection scheme. The verification stage consists of two sub-stages: The individual part verification and the combined verification. Three SVM

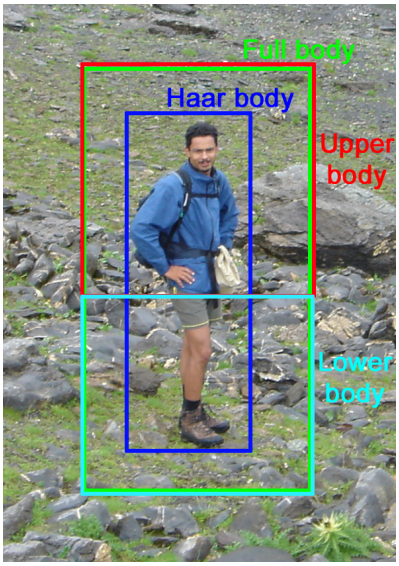


Fig. 3. The four types of training images used in this system: The three parts for the verification stage, and a closer crop for the detection stage.

regressions based on dense HOG descriptors are calculated and applied to the ROIs given by the detection stage. One is for full body classification, one is for lower body classification, and one is for upper body classifications.

Our algorithm uses classic dense HOG descriptors (as opposed to the sparse descriptors used in [4]). They are calculated using integral images in an effort to speed up the process, as described in [17]. Since HOG works best if some amount of background is introduced to the detection window, the ROIs are resized appropriately from the tight boxes that are returned by the detection stage. Then the content of the ROIs is scaled so it matches the size the SVMs were trained with. At this point the HOG is calculated and passed on to the SVMs.

As in the detection stage, each SVM is trained with the INRIA training set. The full body SVM was trained with the full training images, whereas the lower- and upper-body SVMs were trained with the lower and upper half of the training images, respectively. In our system, there is no overlap between the lower and upper body. The parts of training images used for each type are shown in fig. 3. So in total, three SVMs were used.

To do the combined verification, two different methods were tested: Majority voting and regression output classification.

For majority voting, a regular SVM for classification was trained. It returns which class (pedestrian vs. non-pedestrian) the current detection window belongs to. If at least two out of three classifiers label the window as a pedestrian, it is described as a detection. If a detection is labeled 1 and no detection is labeled -1, the formula used for the majority voting is:

$$l_{out} = \begin{cases} 1 & \text{if} \sum_{i=0}^{i<3} l_i >= 1 \\ -1 & \text{if} \sum_{i=0}^{i<3} l_i < 1 \end{cases} \quad (1)$$

TABLE I

OVERVIEW OF THE DETECTION RATES ACHIEVED BY GEISMANN AND SCHNEIDER [4] WITH 0.2 FALSE POSITIVE PER FRAME

Video	1	2	3	4	5	Mean
Dense descriptor	52%	70%	91%	61%	55%	65.8%
Sparse descriptor	45%	53%	85%	69%	58%	62%

where l_{out} is the final decision and l_i is the output from one of the three part-based detectors.

For regression output classification, the three part SVMs were instead trained for regression. The training was performed so the resulting function would ideally return 1 in the case of a detection and -1 when nothing was found. When an unknown window is passed through the output function, it will return a value close to 1 if it is a pedestrian, and a value close to -1 otherwise. The output of these three regressions create their own 3 dimensional feature space. Another SVM has been trained to classify in this space. The output from the three regressions is passed into this second SVM and the output from that classifier is the final label.

V. EXPERIMENTS AND TEST

In order to set various parameters so that the best possible performance is achieved, several experiments have been performed. While the training part of the INRIA dataset was used to train both the detection stage and the verification stage, the test part has been used as base for these experiments. It contains 742 images in total, of which 289 contain one or more persons. In total the test set contains 589 persons that should be detected by a perfect system.

The baseline for the comparison is the performance of the system in a configuration similar to the one by Geismann and Schneider: Two stages, but no part-based verification. The principal results from their paper can be seen in table I. Each of the five results in the table are from a test video they obtained from a driving car. Unfortunately we do not have access to the test videos they used, so our results cannot be compared directly with those. Instead we compare the performance of our own implementation of their algorithm to out part-based algorithm.

One of the most important parameters in the system is the number of stages in the Haar-cascade, in this paper designated k. It is interesting to see what impact the changes in k has on the complete system. In fig. 4, an ROC curve is shown for the full system with varying depths in the detection stage. The importance of this parameter is evident. As k is lowered, the number of detections rise, but at a large cost in false positives. For the final system, we chose to go with $k = 15$, since it seems to give an acceptable trade-off between true positives and false positives.

The choice of k has an impact on the speed of the system, since more detection windows means slower performance. The full (but non-optimized) system has been run with several numbers of stages and timed, to get a sense for the speed effects it might have. The results are seen in table II

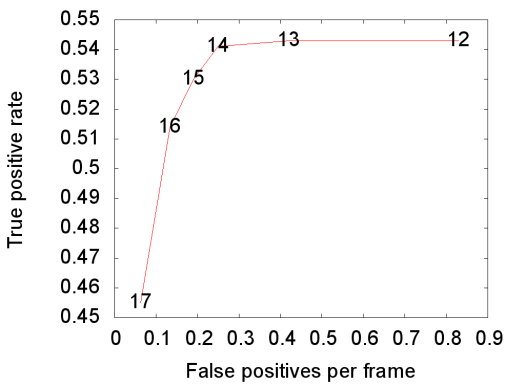


Fig. 4. Receiver-Operating-Characteristic for the full system with varying k, cascade depths in the detection stage.

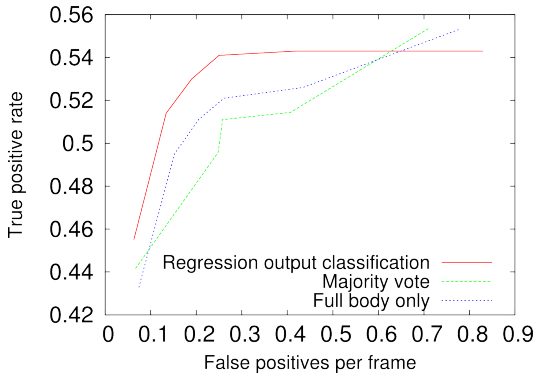


Fig. 5. Receiver-Operating-Characteristic for the final system. The majority voting approach is not performing better than the full-body approach, but the regression classification approach is consistently better.

Another important parameter is the padding, p: The amount with which the ROIs returned by the detection stage is enlarged with. The HOG-SVM detector works better if more background is included than what the Haar-cascade uses, so there is no question that the ROIs must be enlarged. Experiments showed that a padding of 3 performed best. Because HOG-SVM is not scale invariant, so when the padding rises, the pedestrian in the ROI becomes a lot smaller, relative to the image, than the pedestrians in the training set. That will alter the output of the detector and some testing was required to make it work properly. The padding value itself is used to calculate the padding in pixels to apply to the ROI. The width in pixels is calculated as:

$$p_{pixels} = \frac{w_{ROI}}{w_t} \cdot p \tag{2}$$

where p is the padding value, w_{ROI} is the width of the found ROI, w_t is the width of the training images, and p_{pixels} is the padding measured in pixels. The padding is applied on all four sides of the image. width of detected ROI/width of the training image * padding

After introducing part-based verification to the system, experiments were made to determine whether the simple majority voting or the confidence classification worked the best. These tests were done with the best settings, as determined earlier in this section. The results are shown in fig. 5. In absolute numbers, the detection rate is not overwhelming. The important part is the difference between the old two-stage approach with only full-body verification and the new approach. While the voting based approach is not any better then the old full-body verification, the part-based version with regression output classification is better all across the range of false positives per frame.

Examples of detections can be seen in fig. 6.

VI. Future work

The algorithm has a series of parameters that can be adjusted to enhance performance. In this work, a few tests and comparisons has been carried out, in order to give the best performance. However, a more formal investigation of the optimal parameters would be interesting. One possibility is to use a genetic algorithm or particle swarm optimization to set the best parameters.

TABLE II

Changes in processing speed for different values of k

Number of images	100
Mean image width	711.88 pixel
Mean image height	818.72 pixel

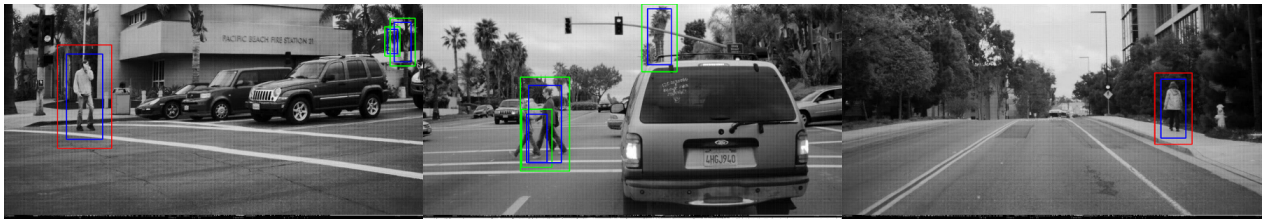
k	**Mean time per frame**
12	2.5 s
13	1.84 s
14	1.57 s
15	1.46 s
16	1.39 s
17	1.22 s

This work has mostly been concerned with lowering the number of false positives in the classic combination of a Haar cascade and HOG-SVM, so speed has not been a primary concern. First and foremost, several optimizations, such as using sparse HOG calculation, are presented in [4], so they could be implemented with little impact on performance.

The system presented here deals only with single-frame detection. A full pedestrian detection system would very likely benefit a lot from using tracking between frames to enhance the performance.

VII. Concluding remarks

In this paper, a part-based two-stage pedestrian detector has been presented. It builds on previous work by Geismann and Schneider [4], but extends it by introducing a part-based verification system instead of just a full body verification. The system works in two stages: A detection stage based on an AdaBoost cascade on Haar-like features. Its purpose is to find all pedestrian candidate patches in the input image. All these Regions Of Interest are sent on to a verification stage, where the Histogram Of Oriented Gradients (HOG) is computed for the entire person, the lower body, and the upper body. Each of the HOGs are then sent trough an SVM that computes a confidence value for which class (pedestrian or



(a) (b) (c)

Fig. 6. Example outputs of the detector. Input images are images captured using one of LISA's experimental cars. Red boxes indicate a detection, blue are the candidates from the detection stage, and green are the candidates with added padding. In (a) the pedestrian is detected, while a couple of false candidates from the detector are ignored. In (b) the pedestrians are not detected, since the detection stage does not find accurate enough candidate boxes. In (c) the pedestrian is detected and no false windows are found in the detection stage.

non-pedestrian) the part belongs to. These values are then passed into a second SVM-classifier, which performs the final verification. The system has been tested on the INRIA dataset and the results show that when compared with the original two-stage detector, it performs better across the full range of false positives per frame.

VIII. Acknowledgment

The authors would like to thank our colleagues in the LISA-CVRR lab and Vislab for feedback during the work. Especially valuable were the comments from Dr. Brendan Morris and Sayanan Sivaraman.

References

[1] D. Geronimo, A. Lopez, A. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 7, pp. 1239–1258, 2010.

[2] T. Gandhi and M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 506–511.

[3] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, and H. Jung, "A new approach to urban pedestrian detection for automatic braking," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 4, pp. 594–605, 2009.

[4] P. Geismann and G. Schneider, "A two-staged approach to vision-based pedestrian recognition using Haar and HOG features," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 554–559.

[5] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. Ieee, 2005, pp. 886–893.

[7] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi, "Pedestrian detection using infrared images and histograms of oriented gradients," in *Intelligent Vehicles Symposium, 2006 IEEE*. Ieee, 2006, pp. 206–212.

[8] T. Gandhi and M. Trivedi, "Pedestrian collision avoidance systems: A survey of computer vision based recent studies," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 2006, pp. 976–981.

[9] ——, "Pedestrian protection systems: Issues, survey, and challenges," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 3, pp. 413–430, 2007.

[10] M. Enzweiler and D. Gavrila, "Monocular pedestrian detection: Survey and experiments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 12, pp. 2179–2195, 2009.

[11] X. Yuan, X. Shan, and L. Su, "A Combined Pedestrian Detection Method Based on Haar-Like Features and HOG Features," in *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*. IEEE, 2011, pp. 1–4.

[12] W. Yongzhi, X. Jianping, L. Xiling, and Z. Jun, "Pedestrian Detection Using Coarse-to-Fine Method with Haar-Like and Shapelet Features," in *Multimedia Technology (ICMT), 2010 International Conference on*. IEEE, 2010, pp. 1–4.

[13] X. Mao, F. Qi, and W. Zhu, "Multiple-part based Pedestrian Detection using Interfering Object Detection," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 2. IEEE, 2007, pp. 165–169.

[14] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 90–97.

[15] ——, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, 2007.

[16] I. Alonso, D. Llorca, M. Sotelo, L. Bergasa, P. de Toro, J. Nuevo, M. Ocaña, and M. Garrido, "Combination of feature extraction methods for SVM pedestrian detection," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 2, pp. 292–307, 2007.

[17] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 829–836.

B

Jacobs Research Expo poster

The poster presented at the Jacobs Research Expo is included on the following page. This is to give an idea of its design and layout, and while the text is not readable at such a small reproduction, it does not contain any information not contained in *Learning to Detect Traffic Signs: Comparative Evaluation of the Roles of Real-world and Synthetic Datasets*, which can be found in appendix A.

UCSD Jacobs | Electrical and Computer Engineering

Detection of US Traffic Signs Using Computer Vision

LISA Laboratory for Intelligent and Safe Automobiles, UCSD

Andreas Møgelmose

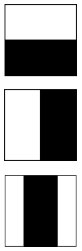
Introduction

This work investigates various types of training data to detect US traffic signs. Traffic Sign Recognition (TSR) has seen much work in the last decade, with the emergence of sophisticated driver assistance systems and research in autonomous cars. See [1] for a survey of current methods.

Most of the current research is concerned with European, Japanese, and Australian signs. Since US signs are significantly different from these, the detectors cannot be reused directly. Machine learning based detectors rely on a large set of training images that must be manually annotated by a person. This is time consuming, and the work presented here explores ways of training a traffic sign detector with synthetic data.

Methods

To evaluate the two types of training data, an **AdaBoost cascade on Haar-like features** was employed [2]. It is a machine learning-based approach, which evaluates linear combinations of a number of simple features (see figure to the right) over a number of stages. When the algorithm is trained, it attempts to find the Haar-like features that best describe the sought after object, in this case a traffic sign.



For training and test purposes, a database containing **7855 instances of US traffic signs in 48 different classes were collected**. For these tests, the signs Stop and Signal Ahead were chosen, since they are well represented in the data set.

Cascades were trained with a subset of the database and tested on a second subset. Another pair of cascades were trained using **synthetic training data** generated by applying 10 different transformations (such as changes in perspective, hue, and lightness) to an artificial template of the sign. Finally the variance of two central parameters, lightness and blur, of the synthetic data is compared.

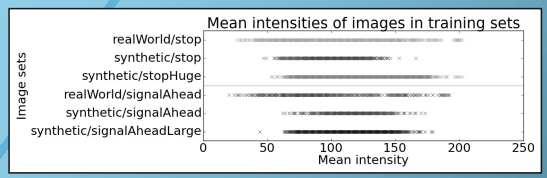
Results

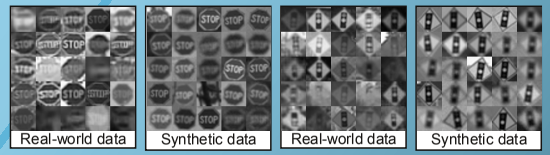
To the right are examples of real and synthetic traning data, respectively. In the table below, the detection rates for the two types of data are presented for varying numbers of training image. Clearly the **synthetic data performs significantly worse** than the real-world data.

At the bottom of this column are plots of the intensities and blur-values, showing a comparison of the ranges of appearances that real-world and synthetic data cover.

Table 1: Results of the comparative evaluations of detectors

Training type	Training images (positive/negative)	Stages	Signs to find	TP	FP	FN
Stop						
Real-world	1218/2500	20	103	76 (73.8%)	11	27
Real-world	1686/3000	20	103	75 (72.8%)	8	28
Synthetic	1218/2500	17	103	18 (17.5%)	2	85
Synthetic	5000/10000	19	103	26 (25.2%)	5	77
Synthetic	1218/2500	10	103	60 (58.3%)	1500	43
Pedestrian crossing						
Real-world	364/800	20	40	29 (72.5%)	10	11
Real-world	1044/2000	20	40	30 (75%)	2	10
Synthetic	364/800	14	40	11 (27.5%)	28	29
Speed limit 35						
Real-world	253/500	20	21	15 (71.4%)	1	6
Synthetic	253/500	7	21	5 (23.8%)	32	16
Synthetic	2000/4000	7	21	6 (28.6%)	6	15
Signal ahead						
Real-world	597/1500	20	56	42 (75%)	10	14
Real-world	859/2000	20	56	38 (67.9%)	4	18
Synthetic	597/1500	13	56	14 (25%)	117	49
Synthetic	2000/4000	13	56	16 (28.6%)	53	48





Discussion

With all signs, the real-world data performs significantly better than the synthetic data. Providing more training data in the synthetic case does help, but even a large increase - more than a doubling - of the traning data does not make the synthetic data perform comparably to the real-word data. One reason the synthetic detectors perform worse than their real-world counterpart could be that the training set, even if large, does not cover the same variance in appearance as the real-world data. However, plots of the mean intensities and the blur values of various datasets show that even in cases (like the stop sign detector with 5000/10000 training images) where the synthetic data spans nearly the same space as the real-world detector, the synthetic detector fails to achieve a detection rate anywhere near the real-world data. And that is with detectors where the real-world training set is much smaller than the synthetic training set (a factor of 20).

This work shows that **there is simply no sufficiently good substitute for real-world data**. The real-world training data performs significantly better than the synthetic training data, even in cases where the synthetic data seems to span a similar set of appearances.

References

[1] M.-Y. Fu and Y.-S. Huang, "A survey of traffic sign recognition," in International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), july 2010, pp. 119–124.

[2] P. Viola and M. Jones, "Robust real-time object detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, 2001.

May, 2012

Supervisors:
Mohan M. Trivedi
Thomas B. Moeslund

AALBORG UNIVERSITY

UC San Diego

C

Training a cascaded classifier with OpenCV

In order to train a cascaded classifier, positive sample image and negative sample images must be obtained. Positives should contain a picture of the object that is to be detected. Negatives should be guaranteed to *not* contain any instances of the object.

OpenCV comes with several tools to train a cascade. opencv_traincascade is the main program that actually performs the training and opencv_createsamples can help creating and organizing the training images. OpenCV requires the positive training images to be of the same aspect ratio, and if they are not the same size, OpenCV is able to resize them. However, the easiest thing is to resize them beforehand in order to know exactly what training goes into the cascade.

When the positive training images has been obtained, two ImageMagick commands can be used to resize and crop them into the proper size and aspect ratio (in this case resulting in images 18 pixels wide and 24 pixels high):

```
1 mogrify -resize 18x24^ *.png
2 mogrify -gravity center -crop 18x24+0+0 *.png
```

After that, a .dat-file (the extension does not really matter, it is a simple plain text file) should be produced, that contains a list of all the positive images in the format `[filename] [number of objects] [[x, y, width, height of object 1] [x, y, width, height of object 2] ...]`. If the images are pre-cropped to only contain one object, a line is simply `[filename] 1 0 0 [image width] [image height]`. For the negatives a list of files should be created. To create a .dat-file for pre-cropped positive images and one for negative images via the Unix command line, use:

```
1 find . -name '*.png' -exec echo \{\} 1 0 0 18 24 \; > positives.dat
2 find . -name '*.png' > negatives.dat
```

Based on the .dat-file, the positive image should be packed into a binary .vec-file. This can be done with the opencv_createsamples utility (the second command simply verifies that the .vec-file works):

```
1 opencv_createsamples -info positives.dat -vec positives.vec -w 18 -h 24
      -num 7000
```

```
opencv_createsamples -vec positives.vec -w 18 -h 24
```

Note that in the above, the -w and -h parameters are the width and height of the positive training images, so those numbers should correspond to the numbers used in the mogrify-commands executed earlier.

Now, the actual training can be performed with opencv_traincascade:

```
opencv_traincascade -data newCascade -vec positives.vec -bg negatives.¶
    dat -numPos 7000 -numNeg 10000 -numStages 20 -precalcValBufSize 512¶
     -precalcIdxBufSize 512 -featureType LBP -w 18 -h 24
```

The `-data` parameter defines where to put the resulting cascade. `-vec` should point to the file containing the positive training images and `-bg` to the list of negative training images. `-numPos` and `-numNeg` defines how many positive and negative images to train with, and `-numStages` how many stages the classifier should contain (however, if sufficient accuracy is obtained at an earlier stage, the training program will terminate). `-¶ precalcValBufSize` and `-precalcIdxBufSize` defines how much memory is allocated for the training. The more, the better, but the total should not exceed the amount of free memory on the machine running the training. `-featureType` can be either Haar (Viola and Jones, 2001; Lienhart and Maydt, 2002) or LBP (Liao et al., 2007) (faster).

When training, opencv_traincascade will save each stage independently and assemble them into one XML-file in the end. If you wish to assemble a detector before the full training is done, simply run opencv_traincascade again, but with `-numStages` set to a number equivalent to the number of stages that are done. It will load all the stages that are done and assemble the final XML-file.

D

Using Video Annotator

Video Annotator is the simplest of the two annotators. Its purpose is to split long source videos into smaller clips called *tracks*. When the program is started, the screen in fig. D.1 shows. From here, click `Load video...` to start annotating a video. A prompt will show, where video files can be opened.

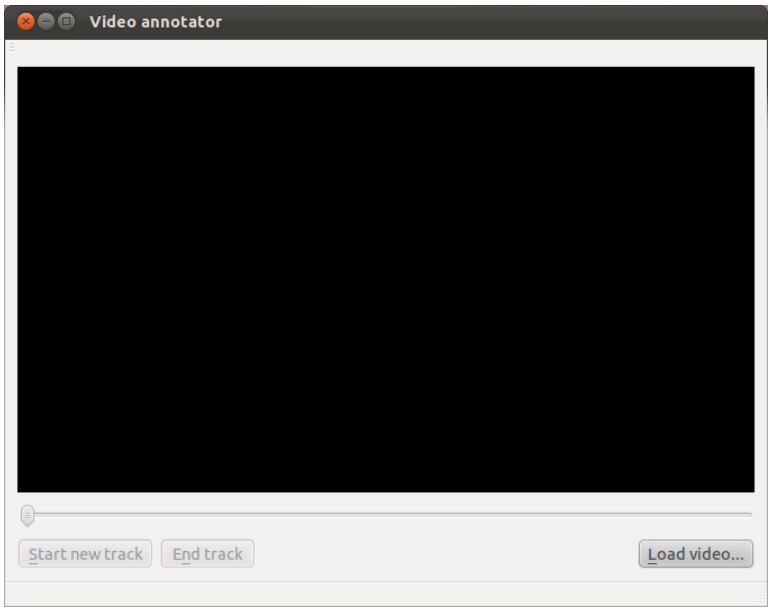


Figure D.1: The initial state of Video Annotator.

After the video has been loaded, it will begin playing immediately. A bar informs about the progress through the video, but the tool does not support searching to specific places in it. When an object deserving annotation shows up, click `Start new track` or press the hotkey `s`. This will open a dialog window as seen in fig. D.2. In the top input field, input the tag that will be associated with this track. The field below shows tags previously used in this session, as well as tags defined in the file `trackTypeList.txt`, located in the same directory as the Video Annotator executable. Video annotator provides auto complete on all entries in the list below the input field, but it is also possible to input a new tag. It will automatically be added to the list, so it has auto complete for any further annotations in this session. The tag will not be saved into

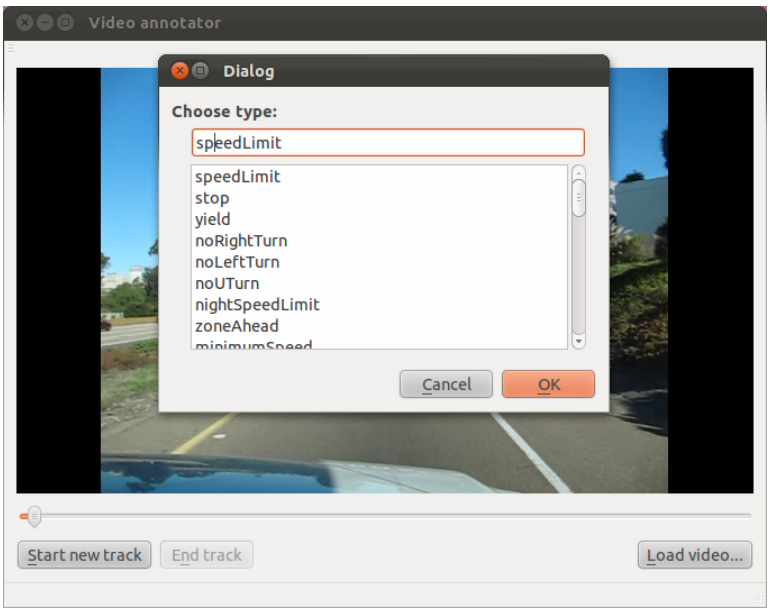


Figure D.2: Starting a new track.

`trackTypeList.txt` automatically, so to add it there, the file should be edited manually.

Multiple tracks can run simultaneously, so there is no requirement of stopping a track before starting a new one. This is relevant in case several interesting objects appears in overlapping segments of the source video.

When the annotated object disappears from view so the track should be ended, click `End track` or press the hotkey `n`. If only one track is currently running, it will be ended. If several are running, the user will be prompted to tell the program which track to end.

For each annotation ended, a video is saved containing the track, and a line is added to the file `<videoFileName>_annotations.csv`. The csv file contains information on each track: Its corresponding video file, its tag, and information on where the track originated, ensuring full traceability of all tracks. The video files and the csv files are all saved in the same directory as the source video is located in. They have a timestamp appended to their file names to prevent any file naming collisions.

E

Using Frame Annotator

Frame Annotator takes the output of Video Annotator and annotates it in further detail. It allows the user to annotate frames in tracks at a specific interval. When starting the program, the interface looks like fig. E.2. The black area is where the frames will show up for annotations, the left pane holds information about the current annotation, and the bottom contains various controls.

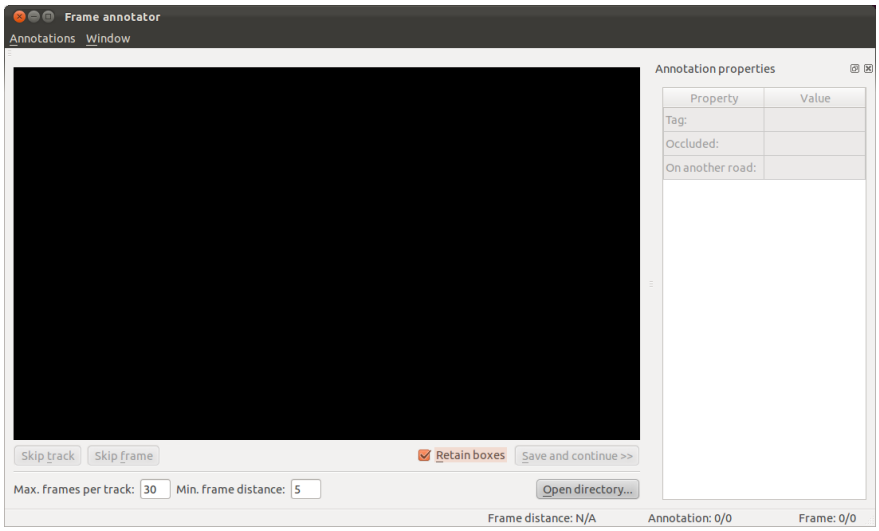


Figure E.1: The initial state of Frame Annotator.

Before annotation starts, there is the option to edit the meta data fields. This cannot be done after annotation has been initiated, since then the header of the output file has been written. Each annotation can carry an unlimited number of meta data fields. A meta data field is a boolean value that can be toggled independently for each annotation. In fig. E.2, it can be seen that the two meta data fields currently set are called `Occluded` and `On another road`. To edit the fields, go to `Annotations` → `Edit¶ meta data fields...` (see fig. E.2), or to change the fields more permanently, edit the file `metaDataList.txt`. It is also possible to change the interval of annotated frames by editing the field `Max. frame distance` and an upper limit of annotated frames from each track can be set by editing the field `Max. frames per track`. If these fields are changed during run time, they will not take effect until the next track is loaded.

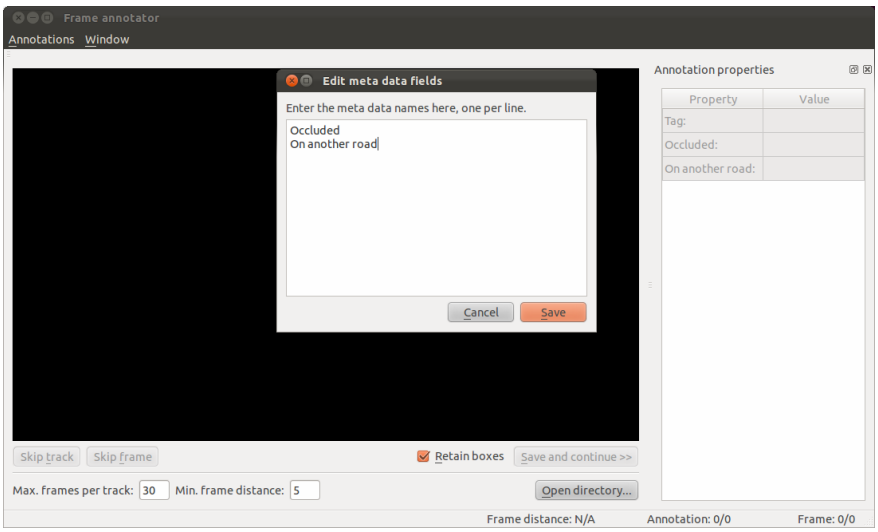


Figure E.2: Editing meta data settings.

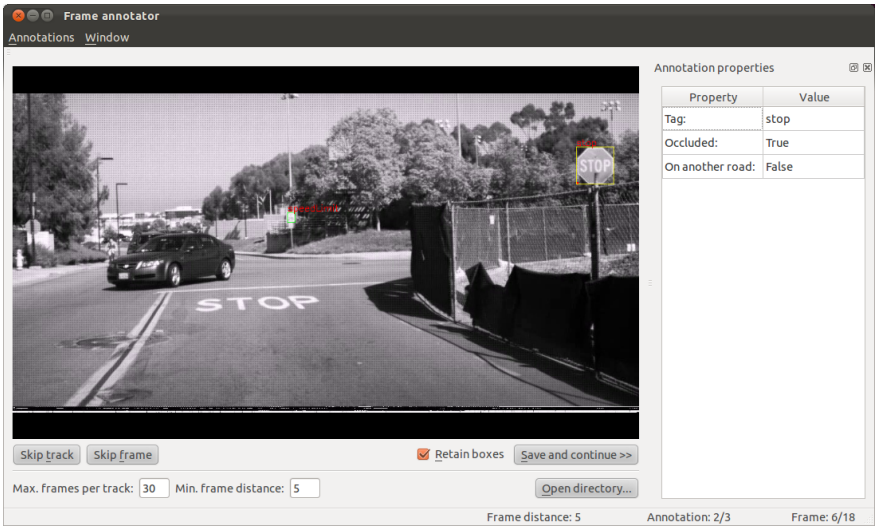


Figure E.3: Annotating with Frame Annotator.

When the fields has been set up, open a set of tracks for annotations by clicking `Open directory...` and opening the csv file output from Video Annotator, which contains a list of tracks. Frame Annotator will automatically open the first track in the set, and show the first frame up for annotation. Frame Annotator is set up for rapid annotation, so many functions are mapped to shortcut keys, which may appear confusing to learn at first, but will quickly start to make sense.

An annotation is started by clicking in one corner of the annotation and ended by clicking in the other. The annotation does not have to be completely accurate at first, since it can easily be adjusted using the keyboard shortcuts described in table E.1. After clicking to finish the box, the newly set annotation will be set as the active annotations, marked by a yellow outline. Other non-active annotations that may be in the image are marked with a green outline. The data for the current annotation can be seen in the property pane and adjusted using either keyboard shortcuts or the

Table E.1: Keyboard shortcuts for Frame Annotator

Operation	Shortcut
Move annotation up	w
Move annotation down	s
Move annotation left	a
Move annotation right	d
Shrink annotation vertically	i
Grow annotation vertically	k
Shrink annotation horizontally	j
Shrink annotation horizontally	l
Select next annotation	e
Select previous annotation	q
Set tag	F2
Toggle meta data field 1-8	F5 - F12
Save and continue to next frame	Enter

property pane itself by double clicking the property to change. If any of the meta data fields are toggled to true, a small red box is shown inside the annotation frame.

Starting a new annotation is as simple as clicking anywhere outside the present annotations. When all objects have been annotated, click `Save and continue >>`. This will save the current frame if it contains any annotations, and it will save info on the annotations to the `frameAnnotations.csv` file. All files are saved to a folder called `frameAnnotations-<annotationFileName>`, which is automatically created in the path that contains the original track csv file. It is required that all annotations has a tag, and when the user presses `Save and continue >>`, the program will prompt for a tag for any untagged annotations.

If `Retain boxes` is checked, any annotations on the current frame will be preserved for the next frame, so their position and size can simply be adjusted without the need to create a new annotation. Annotations are never retained across tracks.

The final options are to click either `Skip frame` or `Skip track`, which will skip to the next frame and the next track, respectively, without saving any annotations in the current frame. All annotation up until the current frame will still be saved, though.

If Video Annotator output is not relevant, Frame Annotator can work with any video, simply create a csv file containing the following two lines:

```
1 Filename;Track type;Origin file;First frame;Length
2 <videoFileName>;;;0;0
```

F

Various data handling methods

F.1 Show classes and their sizes

```
cat mergedFrameAnnotations.csv | awk -F \; '{print $2}' | sort | uniq -c
```

F.2 Black out signs present in the training

This procedure assumes that `annotation-split1.csv` contains the test and `annotation-split2.csv` contains the training annotations.

```
python ~/p10/code/annotationTools/annotateVisually.py -n annotation-split1.csv # Find test images.
awk -F ';' '{print $1}' annotation-split2.csv > trainingFiles.txt # Find the file names of all training files
grep --file=trainingFiles.txt annotation-split1.csv > trainingAnnotationsInTest.csv # Find the training annotations that exist in the test images.
python ~/p10/code/annotationTools/annotateVisually.py -b trainingAnnotationsInTest.csv # Black out the signs (overwriting relevant images)
```

Or just run the bash script `blackOut.sh`, which does all this automatically.

F.3 Evaluate the performance of a cascade against annotated data

The backbone in this operation is the Python script `evaluateDetections.py`, which is used like this:

```
$ python ~/p10/code/annotationTools/evaluateDetections.py -h
usage: evaluateDetections.py [-h] [-c] [-f dirname/] [-w] [-fn] [-s 20x20]
```

```
                              detections.csv annotations.csv

Print stats on detection, given a detection file and a ground truth file.

positional arguments:
  detections.csv        The path to the csv-file containing the detections.
  annotations.csv       The path to the csv-file containing the annotations
                        that are the ground truth.

optional arguments:
  -h, --help            show this help message and exit
  -c, --copyFP          Copy images with false positives to the falsePositive/
                        directory.
  -f dirname/, --saveFP dirname/
                        Path to the original test-images to extract the false
                        positive patches to. If not given, they are not
                        extracted.
  -w, --widthHistogram  Show a histogram of the widths of true positives.
  -fn, --falseNegativesOnly
                        Print only the false negatives.
  -s 20x20, --sizeMinimum 20x20
                        Disregard any annotation smaller than the specified
                        size. First number is width.
```

the previous

`evaluateDetections.py` requires that the detector has already been run and its output is readily available. To automate the entire process, use the bash script `testCascade.sh`:

```
# Input arguments: <cascade file> <test file> <reference file> <result destination>
# Example:
./testCascade.sh synthetic/stop/stopSynCascade.xml stopTest/stopTest.dat /media/Data/signDatabase/stop-split1.csv stopTestSyn
```

The above example runs the `stopSynCascade.xml` detector on the files specified in `stopTest.dat` and compares the detections to the ground truth in `stop-split1.csv` (the format of that file must be compatible with the output from Frame Annotator, see appendix E). The output is saved in the folder `stopTestSyn`, which will contain all the test images with markings of detections, a csv-file called `detections.csv` contains the detections and a text-file called `detectionResults.txt`, which contains the statistics of the detections compared to the ground truth.

F.4 Create ROC curves for cascades

ROC curves for cascaded classifiers can be created by changing the number of stages, thus altering the relationship between true positives and false positives. A couple of tools have been developed to automate this process.

First, cascades with a different number of stages mu be produced. This can be done with the shallowCascades.sh bash script, which wraps the `convert_cascade` tool of OpenCV. The usage is:

```
# Input arguments: <cascade path> <training image size> <starting ¶
    number of stages> <end number of stages> <output file name>
# Example:
./shallowCascades.sh realWorld/stop/stopCascade 20x20 20 5 stopCascade
```

The above will output a number of stopCasade1.xml, stopCasade2.xml, etc. files in the directory realWorld/stop/.

The next step is to evaluate each of these cascades. That can be done with the tool `testMultipleCascade.sh`, which wraps the `testCascade.sh` script and runs it automatically for several detectors:

```
# Input arguments: <start cascade number> <end cascade number> <cascade¶
    file pattern> <test file> <annotation file> <result destination ¶
    pattern> <minimum detection size>
# Example:
./testMultipleCascades.sh 20 5 realWorld/stop/stopCascade stopTest/¶
    stopTest.dat /media/Data/signDatabase/stop-split1.csv stopROC 21x21
```

This will output an number of folders called stopROC20, stopROC19, stopROC18, etc. and a file called `stopROCResults.txt`.

The final step is to generate the ROC curve. This is done with the Python script `createROC.py`:

```
python createROC.py stopROCResults.txt -t 'ROC curve for real-world ¶
    trained stop sign detector'
```

When running this, a window with the ROC curve will show up.

So, to recap, these three commands will generate an ROC curve for a cascade:

```
./shallowCascades.sh realWorld/stop/stopCascade 20x20 20 5 stopCascade
./testMultipleCascades.sh 20 5 realWorld/stop/stopCascade stopTest/¶
    stopTest.dat /media/Data/signDatabase/stop-split1.csv stopROC
python createROC.py stopROCResults.txt
```

Acronyms

TSR Traffic Sign Recognition

HOG Histograms of Oriented Gradients

MUTCD Manual on Uniform Traffic Control Devices

SHSM Standard Highway Signs and Markings

GTSRB German Traffic Sign Recognition Benchmark

CVRR Computer Vision and Robotics Research

LISA Laboratory for Intelligent and Safe Automobiles

UCSD University of California, San Diego

ITSC Intelligent Transportation Systems Conference

ICPR International Conference on Pattern Recognition

SVM Support Vector Machine

FRS Fast Radial Symmetry

EFRS Extended Fast Radial Symmetry