Development of Evolution Based Technology for Image Recognition Systems

Jens-Petter S. Sandvik

Cand. scient. thesis

November 4, 2005
Abstract

A traffic sign detection system in the vehicle can be of great help for the driver. The number of accidents can be reduced by 20% if the speed limits are followed. A system that warns the driver about speeding could therefore save lives if the driver reduces the speed.

This work focuses on the colour classification used in traffic sign detection methods. Existing methods are compared, and a Genetic Algorithm is used for optimising parameters used in the existing colour classification methods.

Cartesian Genetic Programming is used for evolving colour classifiers for traffic signs, and compared to the existing methods. The evolved classifier is tested with three different luminance adjustment algorithms.

The results show that the GA is able to find better parameters than the reported parameters, and some of the evolved colour classifiers were better than the existing methods. The CGP architecture did find better classifiers than the existing. The luminance adjustment algorithms did not result in better classification results.
## Contents

1 Introduction ........................................ 1  
  1.1 The quest for safer driving .......................... 1  
  1.2 Problem description .............................. 3  
  1.3 Notation and conventions .......................... 3  
  1.4 Organisation of the document ........................ 4  

I Background ........................................... 5  

2 Genetic algorithms and evolvable hardware .......... 6  
  2.1 Terminology ........................................ 6  
  2.2 Genetic algorithms ................................ 7  
  2.3 Evolutionary Strategies ............................ 9  
  2.4 Cartesian Genetic Programming .................. 9  

3 Digital image processing ............................. 13  
  3.1 Digital image processing and object recognition .................................................. 13  
  3.2 Colour spaces .................................... 15  
  3.3 Noise and artifacts ............................... 17  
  3.4 Low-level processing ............................. 18  
     3.4.1 Noise reduction ................................ 19  
     3.4.2 Luminance adjustments ......................... 19  
     3.4.3 Histogram equalisation ....................... 21  
  3.5 Intermediate level processing .................... 23  
     3.5.1 Classification and segmentation .......... 23  
  3.6 Digital image processing in hardware ............ 25  
  3.7 Evolvable methods used in digital image processing .................................................. 25  
     3.7.1 Noise reduction ................................ 25  

4 Intelligent Transportation Systems .................. 29  
  4.1 Intelligent Transportation Systems ................ 29  
     4.1.1 What is ITS? ................................ 29  
     4.1.2 ITS research ................................ 30  

5 Traffic sign detection

5.1 Norwegian speed limit signs
5.1.1 The sign standard
5.1.2 Speed limit signs
5.1.3 Rules governing the speed limit

5.2 Traffic sign detection
5.2.1 Traffic sign detection for Norwegian signs

5.3 Other colour segmentation algorithms used for traffic sign detection

II Evolution based methods for colour classification

6 Methods based on evolution for colour classification

6.1 Tuning of parameters using GA
6.2 Using CGP for colour segmentation
6.3 Luminance adjustments

7 General optimisations

7.1 A fast method for Value manipulation in the HSV colour space
7.1.1 Mathematical background
7.1.2 Histogram equalisation in the HSV colour space

7.2 A short note on the HSV colour mapping

8 Experiments

8.1 Description of the experiments
8.1.1 Framework and tools
8.2 Dataset
8.3 Evaluation methods

8.4 Experiment: Evolution of parameters
8.4.1 Objective
8.4.2 Parameters
8.4.3 Results and observations

8.5 Experiment: Using CGP for colour segmentation
8.5.1 Objective
8.5.2 Parameters
8.5.3 Result and observations
8.5.4 Comparison with the tested, existing methods

8.6 Experiment: Luminance adjustments
8.6.1 Objective
8.6.2 Parameters
8.6.3 Result and observations

8.7 Experiment: Optimised Value manipulation in the RGB colour space
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.7.1</td>
<td>Objective</td>
<td>77</td>
</tr>
<tr>
<td>8.7.2</td>
<td>Parameters and setup</td>
<td>79</td>
</tr>
<tr>
<td>8.7.3</td>
<td>Result and Observations</td>
<td>79</td>
</tr>
<tr>
<td>9</td>
<td>Conclusion</td>
<td>80</td>
</tr>
<tr>
<td>9.1</td>
<td>Conclusion</td>
<td>80</td>
</tr>
<tr>
<td>9.2</td>
<td>Future work</td>
<td>81</td>
</tr>
<tr>
<td>III</td>
<td>Appendices and bibliography</td>
<td>82</td>
</tr>
<tr>
<td>A</td>
<td>Acronyms used in the text</td>
<td>83</td>
</tr>
<tr>
<td>B</td>
<td>Description and usage of computer programs used.</td>
<td>85</td>
</tr>
<tr>
<td>B.1</td>
<td>Programs developed for this thesis</td>
<td>85</td>
</tr>
<tr>
<td>B.1.1</td>
<td>sigrec</td>
<td>85</td>
</tr>
<tr>
<td>B.1.2</td>
<td>garec</td>
<td>89</td>
</tr>
<tr>
<td>B.1.3</td>
<td>cgprec</td>
<td>90</td>
</tr>
<tr>
<td>B.1.4</td>
<td>teststuff</td>
<td>92</td>
</tr>
<tr>
<td>B.1.5</td>
<td>analyse_cgp</td>
<td>93</td>
</tr>
<tr>
<td>B.2</td>
<td>Third party tools</td>
<td>93</td>
</tr>
<tr>
<td>B.2.1</td>
<td>GALib</td>
<td>93</td>
</tr>
<tr>
<td>B.2.2</td>
<td>Condor</td>
<td>93</td>
</tr>
<tr>
<td>B.2.3</td>
<td>Gnumeric</td>
<td>94</td>
</tr>
<tr>
<td>B.2.4</td>
<td>Octave</td>
<td>94</td>
</tr>
<tr>
<td>B.2.5</td>
<td>LaTeX</td>
<td>94</td>
</tr>
<tr>
<td>B.2.6</td>
<td>Gimp</td>
<td>94</td>
</tr>
<tr>
<td>C</td>
<td>Evolved CGP configurations</td>
<td>95</td>
</tr>
<tr>
<td>C.1</td>
<td>ESA</td>
<td>96</td>
</tr>
<tr>
<td>C.2</td>
<td>ESB1</td>
<td>96</td>
</tr>
<tr>
<td>C.3</td>
<td>ESB2</td>
<td>97</td>
</tr>
<tr>
<td>C.4</td>
<td>ESC</td>
<td>98</td>
</tr>
<tr>
<td>C.5</td>
<td>ESD</td>
<td>99</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Traffic accidents in Norway 1995 – 2004. It has been decline in the number of accidents causing death or serious injuries the last years. ........................................... 2

2.1 A simple genetic algorithm ................................. 7
2.2 Crossover in SGA ........................................... 8
2.3 Mutation in SGA ........................................... 9
2.4 The structure of a Cartesian Program. ....................... 10
2.5 The layout of the genome for the Cartesian Program. The grey nodes are redundant nodes that are not connected to any output. The grey genes in the genome are the genes representing these redundant nodes. ............................. 11

3.1 The object recognition system. (From [GW93].) .............. 14
3.2 The missing triangle/ Kanizsa’s triangle ....................... 15
3.3 The blocking effect and colour bleeding in JPEG compression. Black and white are both red, since the red Hue is wrapped around at 255/0. The rectangular blocks of the same colour (grey tone) is a result of the blocking effect. The red areas stretches beyond the rim of the sign, and is called colour bleeding. ............................................. 18
3.4 The bright background and the dark foreground makes it difficult to see the signs at both sides of the road. The effect of the transfer functions shows that the signs can be more readable. ............................................. 20
3.5 An image segment from the test set. .......................... 21
3.6 The transfer function. Note that the steep slopes in the transfer function corresponds with high peaks in the original histogram. ............................................. 22
3.7 The histogram equalised image segment. ..................... 22
3.8 System used by Sekanina for evolving spatial image filters. (Figure from [ST02].) ............................. 26
3.9 One of the evolved sorting networks for 13 inputs. The horizontal lines are the signal path, and the vertical lines are a compare-swap between the signals. The signal in the middle on the right is the median. (From [Sek04a].)

5.1 Speed limit

5.2 End of particular speed limit

5.3 Speed limit zone

5.4 Informational signs

5.5 A private, two-coloured sign

5.6 The stages in the IDSL method developed by Sekanina, Tørresen and Bakke.

5.7 The transfer functions used for Hue and Saturation in [dIEAM03].

5.8 The RG and BY images created by Kim’s algorithm. Note the high values for the red circle in the RG image, and the high value for the blue rectangle in the BY image.

6.1 The inputs for the CGP architecture. The average value labelled 15, is the average of all colour channels in a 3x3 window.
8.14 Results for ESC with luminance adjustments. The dataset “CO” is the score without adjustment of the HSV Value channel.

8.15 The images that was most problematic when histogram equalised. Notice the effect of the JPEG compression in the image pre-processed with the adaptive histogram equalisation method (d).

8.16 The results of histogram equalisation using a trivial method and the optimised method for different image sizes.
# List of Tables

1.1 The prefix notation for binary multiples, as recommended by IEC. ........................................ 4

3.1 A confusion matrix ........................................ 24
3.2 The configurations of the CGP used by Sekanina. .......... 26
3.3 The functions used by Sekanina in [Sek04b]. ............... 26

5.1 Minimum viewable distance for signs. ........................ 34
5.2 The colour classification rules for IDS. R, G and B is the Red, Green and Blue components from the original image. . 37

6.1 The arithmetic functions used in the CGP. .................... 49

8.1 Overview of the experiments in chapter 8. SGA is the Simple Genetic Algorithm, ES is Evolution Strategy. ............. 56
8.2 The reference methods for colour segmentation. The ID in the first column is used as reference in the text to the different methods. ........................................ 61
8.3 Setup of the genetic algorithm. .............................. 62
8.4 These two parameter sets were the only ones reported in the original papers, and were tested for comparison. ............... 62
8.5 The results from GA tuning of parameters. ................. 63
8.6 Confusion matrix for A1 ................................. 63
8.7 Confusion matrix for A2 .................................. 63
8.8 Confusion matrix for B1 ................................. 64
8.9 Confusion matrix for B2 .................................. 64
8.10 Confusion matrix for C1 ................................. 64
8.11 Confusion matrix for D1 ................................. 64
8.12 Confusion matrix for E1 ................................. 64
8.13 Confusion matrix for F1 ................................. 65
8.14 Confusion matrix for G1 ................................. 65
8.15 ES parameters ........................................... 67
8.16 The parameters to the CGP used with ES. ................. 67
8.17 Confusion matrix for ESA ................................ 68
8.18 Confusion matrix for ESB1 .................................. 68
8.19 Confusion matrix for ESB2. ................................. 68
8.20 Confusion matrix for ESC. ................................. 68
8.21 Confusion matrix for ESD. ................................. 68
8.22 Setup of the experiment with histogram adjustments. . . 74
8.23 The parameters to the adaptive histogram equalisation algorithm. .............................................. 74
8.24 The result from AC. ........................................... 74
8.25 The result from AH. ........................................... 75
8.26 The result from AA. ........................................... 75
8.27 The result from CC. ........................................... 75
8.28 The result from CH. ........................................... 75
8.29 The result from CA. ........................................... 75
8.30 The results for Histogram equalisation using the trivial method, and the optimised method. ...................... 79
Chapter 1

Introduction

1.1 The quest for safer driving

From the beginning of the automobile era, the need for safety measures has been recognised. The first car accident happened in 1771, when Nicolas-Joseph Cugnot crashed his steam powered car into a wall. Great Britain passed a law in the 19th century that forced automobiles on the public roads “to be preceded by a man on foot, waving a red flag and blowing a horn.” This law was not repealed until 1896.

Up through the history more and more safety measures were added to the cars. E.g. Three point seat belts, better break systems and mirrors. Later came Antilock Breaking System (ABS), air bags and crumple zones, or zones that crumple on impact and absorbs the energy in a collision. The infrastructure has also seen huge improvements with e.g. better roads, better road markings and better driver education. Unfortunately these improvements have been followed by more powerful engines and more cars on the roads.

The evolution of car safety is still continuing. With the introduction of small computers, the possibilities for safety systems have increased with orders of magnitude. There has been research done in driver surveillance systems, that can monitor the driver and detect signs of drowsiness. There has also been much research in autonomous driving, where the car itself takes care of the driving.

In Norway, there are about 250-300 fatal accidents and about 1000 accidents causing serious injuries each year. Figure 1.1 shows the number of accidents causing death or serious injuries in Norway from 1995 to 2004. According to Statens Vegvesen (Norwegian Public Roads Administration) 60 fatal accidents and 200 accidents causing serious injuries could have been avoided if the speed limits had been followed. This is about 20 % of the accidents.

Intelligent Speed Adaption (ISA) is a name for systems that supervises the actual speed of the car and compares it with the speed limit. The system can warn the driver that he is driving too fast. The driver can be warned about speeding by some audiovisual warnings or a resistance in the accelerator pedal.

There are several approaches to implement ISA, each method has their own advantages and disadvantages. One method is to let the system in the car receive speed limit information from transmitters along the road. This would be simple to implement and the cost for a car owner is not that much. On the other hand, the infrastructure has to be built, and somebody has to maintain this infrastructure. This can be expensive for the maintainer.

Another approach is to use existing GPS navigation systems, and add speed limit information to the maps used by the system. This has the advantage that the infrastructure maintenance is reduced to updating the maps used, and anybody can develop and maintain such systems. Another advantage is that the route planner can use the speed limit information when calculating the fastest route. The drawback is that the maps are static after being uploaded to the system, and information about new speed limits due to e.g. road work or accidents are not integrated into the system before another update of the maps.

The third approach is to use a camera or similar sensor to recognise
speed limit signs in real time. The advantage is that this does not rely on any external sources of information (apart from the information from the sensors.) Another advantage is that this can be integrated into an even more complex solution that might detect e.g. potential dangerous events in the environment like objects on a collision course etc. The disadvantages is that such a solution will need more processing power, and it might be a considerable expense for the car owner. Another disadvantage is that even with a perfect image processing method, such a system will be prone to errors due to signs that have been tampered with, or other objects hiding the sign.

The first two methods (transmitters along the road and GPS) has been tested in Sweden, and they found that both methods worked, but both method had their advantages and disadvantages that made the methods suited for different environments. The different warning systems (i.e. audiovisual warnings and increasing resistance on the accelerator pedal) did result in the same speed reduction for the test cars[BL02].

1.2 Problem description

A system for traffic sign detection need to work reliably. If it has too many errors, the system can’t be trusted, and is of no use for a driver. So one of the main goals for such a system is that it has to work reliably.

The preliminary studies in this thesis showed that the many object recognition methods needed good segmentation algorithms. And a good segmentation method was dependent on good classification of the pixels. One of the most notable qualities with the signs are the colour information, and this is the focus area of this thesis.

The focus for this thesis is:

- Compare existing methods for colour classification used in traffic sign detection.
- Explore the use of evolution based methods for colour classification.
- General optimisations of the system.

1.3 Notation and conventions

Prefixes to units will follow the Système International d’unités (SI) system. For binary multiples, as are usually found in computer science, recommendations from International Electrotechnical Commission (IEC) will be followed. This recommendation has been proposed in order to avoid confusion between multiples of ten and multiples of two. Some of the prefixes
### Table 1.1: The prefix notation for binary multiples, as recommended by IEC.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Factor</th>
<th># of units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ki</td>
<td>$2^{10}$</td>
<td>1024</td>
</tr>
<tr>
<td>Mi</td>
<td>$2^{20}$</td>
<td>1048576</td>
</tr>
<tr>
<td>Gi</td>
<td>$2^{30}$</td>
<td>1073741824</td>
</tr>
</tbody>
</table>

are listed in table 1.1. The recommendations for binary multiples are supported by both International Committee for Weights and Measures (CIPM) and Institute of Electrical and Electronics Engineers (IEEE).

For multichannel images, the name of the given channel will have an uppercase first letter (or be identified with the first letter, as R in RGB,) the colours will be lowercase and a class representation in uppercase. E.g. “the red colours in a RGB image will have higher values in the Red channel, and most of the red colours will be classified as RED.”

### 1.4 Organisation of the document

The rest of this work is organised in three parts. The first part is theoretical background and a summary of some relevant research. Chapter 2 describes evolutionary algorithms, chapter 3 gives some background in digital image processing, chapter 4 is a short summary of Intelligent Transportation Systems and chapter 5 is a summary of research done in traffic sign detection.

The second part contains a description of the new research done in this master thesis. Chapter 6 is the work on using evolutionary methods in colour classification for traffic sign detection. Chapter 7 describes some optimisations discovered during this work, chapter 8 consists of the experiments and the results. The conclusion and some thoughts about future work is in chapter 9.

The last part is appendices and bibliography. Appendix A contains a list of acronyms used in the text, appendix B is a description of the programs developed during this work, and other tools used. Appendix C gives a detailed listing of the solutions evolved by the experiments.
Part I

Background
Chapter 2

Genetic algorithms and evolvable hardware

Evolutionary Algorithms (EAs) is an umbrella term for all computer-based algorithms that are based upon the idea of evolution\cite{HB00}. There exist a variety of evolutionary algorithms, but all share the conceptual basis of a population, reproduction, selection and environmental fitness. The most used EAs in evolvable hardware are genetic algorithms, genetic programming, evolution strategy and evolutionary programming. Genetic algorithms and evolution strategy are described in more detail below.

2.1 Terminology

Many of the expressions used in evolutionary algorithms are borrowed from the biology and especially genetics.

The genome is a term for all the genetic material, and it consist of one or more chromosomes. In EAs the genome usually consist of one chromosome and the terms are often used interchangeably.

The chromosomes itself consist of genes. In biology, a gene is the blueprint for a particular protein, and in EAs the gene describes a particular feature in the system.

A locus is a position in the gene, and the allele is the value at the particular locus. In a EA the meaning of “the allele at locus 5 is 1” is the same as “the value at position 5 is 1.”

The genotype is the specific representation of a genome, while the phenotype is the representation of the resulting properties of the genome. In many EAs the genotype and phenotype is the same. In biology the difference between these terms might be demonstrated by e.g. the eye colour: Brown eyes is one phenotype, but the genotype for brown eyes might be “Bb” or “BB”, where “B” is the dominant gene for brown eyes and “b” is the recessive gene for blue eyes.
2.2 Genetic algorithms

In Genetic Algorithms (GAs), the individual solutions in a search space is represented by the chromosome. This chromosome can be about anything, but the most basic representation is by a binary 1D array.

Simple GA

The Simple Genetic Algorithm (SGA) is the most basic GA. It uses a straightforward approach with no advanced operators. Therefore it is very easy to understand and analyse this algorithm.

In the SGA, the algorithm starts with a population of chromosomes (also called individuals). This population is usually chosen at random, but you can use some a priori knowledge to give the population a bias. Then the population is evaluated based on the fitness function, and the chromosomes with a better fitness is more likely to be chosen for reproduction. The reproduction is followed by crossover and mutation, and the offspring becomes the next population and this new population undergoes the same steps toward the next generation. (See fig. 2.1)

There are three operators that this SGA consist of which are the fundamental operators in genetic algorithms:

1. Selection
2. Crossover

3. Mutation

**Reproduction** is a process where the individuals are selected for mating according to how “good” they are. This “goodness” is a value that is returned from the *objective function*, and is a numerical value that is a measure of how good this particular chromosome is.

The objective function (also called fitness function) is an artificial simplification of the Darwinian natural selection in nature. In natural selection the population is determined by the individuals ability to survive illness, predators, and all the other aspects of life. But here it is merely a measure of how well it solves a given task.

After the individual chromosomes have been given this fitness, they are selected for mating. The easiest way to do this, is to produce a biased roulette wheel, and to give the individual chromosomes a slot sized in proportion to the fitness, and to spin the wheel as many times as individuals in the new population. This new population will then be the basis for the next operator, crossover.

**Crossover** can be performed in two steps. First the members of this newly produced population are mated at random. Then each pair of chromosomes are crossed. In SGA this crossover is performed by choosing a crossover point in the chromosomes, split the chromosomes at this point and swap the genes after this point. (See fig. 2.2.)

**Mutation** The mutation is performed by flipping a bit with a given probability. (See fig. 2.3.) A mutation rate of e.g. 0.01 means that each bit is flipped with a 1 percent probability. The larger this mutation rate is, the more random the search becomes.
2.3 Evolutionary Strategies

The Evolutionary Strategies (ES), or evolution strategy, was originally developed by Rechenberg in the sixties. The algorithm was originally developed for adjusting parameters in physical experiments[HB00].

The ES works in the following steps:

1. Create a random initial population of $\lambda$ genomes.
2. Evaluate fitness of each genome.
3. The $\mu$ best genomes are selected for mutation.
4. Create $\lambda$ mutations of the genomes selected in the previous step.
5. Return to step 2.

Two flavours of ES exist, namely plus strategy, $(\mu + \lambda)$-ES, and comma strategy, $(\mu, \lambda)$-ES. The difference between these two strategies, is that in the plus-strategy, both the $\mu$ parents and the $\lambda$ offsprings are considered for selection. In the comma-strategy, only the offspring are considered for selection. In GA terms it means that the plus-strategy uses a form of elitism while the comma-strategy has no elitism.

A $(1 + \lambda)$-ES means that only one parent is selected, and it produces $\lambda$ mutated genomes. Both the parent and the $\lambda$ offspring are evaluated and the best genome is then selected.

2.4 Cartesian Genetic Programming

Cartesian Genetic Programming (CGP) was first introduced by Miller and Thompson, as a variant of Genetic Programming[MT00]. In CGP the genome or configuration is mapped into a directed graph instead of e.g. a tree structure as is the case with Genetic Programming.

Cartesian Genetic Programming is expressed through a Cartesian Program, which can be expressed as a set $P = \{G, n_i, n_o, n_n, F, n_f, n_r, n_c, l\}$. $G$ represents the genotype, or the genome. $n_i$ is the number of inputs and $n_o$
Input 0
Input 1
Input 2
Input 3
Input 4
Input 5

Figure 2.4: The structure of a Cartesian Program.

is the number of outputs for the system. \( n_n \) is the number input connections per node. \( F \) is a set of \( n_f \) number of functions. \( n_r \) and \( n_c \) is the size of the node array, in number of rows and number of columns respectively. \( l \) is a levels back parameter that determines how many previous columns can connect to the nodes in the current column. Each input and internal node is assigned an address that is enumerated from the first input and column wise for the internal nodes. Nodes in the same column is not allowed to be connected to each other, so the Cartesian Program is a feed-forward structure. All nodes in the CGP can be connected to the input signals.

Fig. 2.4 shows the structure of the CGP with \( n_i = 6 \), \( n_o = 2 \), \( n_n = 2 \), \( n_r = 4 \) and \( n_c = 3 \).

Each input and node in the CGP are given a unique identification number (ID) that reflects the location of the node in the CGP. The ID start at 0 for the first input, and the last input is given the ID \( n_i - 1 \). The node at the first row and first column has the ID \( n_i \), and the numbering of the IDs continues column wise. The last node in the first column is therefore given the ID \( n_i + n_r - 1 \), and the last node in the last column is given the ID \( n_i + n_r n_c - 1 \).

The genotype, \( G \), of the Cartesian program is expressed as a list of the nodes, and the output connections. Each node contains \( n_n \) node IDs, that describes the nodes connected to this node, and a function ID. Fig. 2.5 shows a overview of the genome, \( G \). The length of the genotype is fixed, and is given by \( (n_n + 1)n_r n_c + n_o \). However, the size of the phenotype is dependent on the internal connections between the nodes in the Cartesian Program.

When the genotypes are created or mutated, they have to obey certain constraints in order to represent a valid program. The constraints for the node IDs that can be connected to the internal nodes, \( e_{\text{min}} \) and \( e_{\text{max}} \), can be
Genome = 0, 1, 1; 0, 2, 4; 1, 3, 2; 4, 1, 3; 5, 6, 2; 4, 6, 2; 7, 7, 1; 8, 9, 3; 9, 6, 4; 7, 12;

Figure 2.5: The layout of the genome for the Cartesian Program. The grey nodes are redundant nodes that are not connected to any output. The grey genes in the genome are the genes representing these redundant nodes.

expressed as:

\[
e_{\text{min}} = \begin{cases} 
n_i + (j - l) n_r & , j \geq l \\
0 & , j < l 
\end{cases} \quad (2.1)
\]
\[
e_{\text{max}} = n_i + j n_r - 1 
\quad (2.2)
\]
\[
e_{\text{min}} \leq c_{k_j}^n \leq e_{\text{max}} \quad (2.3)
\]

where \(c_{k_j}^n\) is the \(k\)-th input to a node in column \(j\), where the leftmost column is labelled 0.

The constraints for the output nodes, \(h_{\text{min}}\) and \(h_{\text{max}}\), can be expressed as:

\[
h_{\text{min}} = n_i + (n_c - l) n_r \quad (2.4)
\]
\[
h_{\text{max}} = n_i + n_c n_r - 1 \quad (2.5)
\]
\[
h_{\text{min}} \leq c_k^o \leq h_{\text{max}} \quad (2.6)
\]

where \(c_k^o\) is the gene representing the \(k\)-th program output.\(^1\)

The function ID for each internal node has to obey the following constraint:

\[
0 \leq c_k^f < n_f 
\quad (2.7)
\]

where \(c_k^f\) is the gene representing the function ID for the \(k\)-th node.

Miller et. al. used two evolutionary methods in [MT00]. One was a generational Genetic Algorithm with uniform crossover, where each gene in

\(^1\)The original paper by Miller ([MT00]) states that \(h_{\text{max}} = n_i + (n_c - 1) n_r\) and \(h_{\text{min}} \leq c_k^o < h_{\text{max}}\), but this will effectively disable output from the last column.
the offspring was randomly selected from the parents and a size two prob-
abilistic tournament selection, where the winner was selected with prob-
ability 0.7. The other method was a simple form of \((1 + \lambda)\) Evolutionary
Strategy with \(\lambda \approx 4\).
Chapter 3

Digital image processing

Digital image processing is a wide field and the span is considerable. It is used in many applications, from analysing satellite images, quality assurance in production lines, enhancing the holiday pictures, scanner and printer soft- and hardware, etc.

The topics presented here are the theories in digital image processing that are used in this work.

3.1 Digital image processing and object recognition

There are many applications where the ability to recognise an object in an image is very useful; about every task that is based completely or partially on visual information can be simplified if a computer can do the processing of the information.

The term “object recognition” will in this thesis mean the recognition of specific objects in images, where an object is a physical structure that may or may not consist of other objects. (I.e. a car is an object, and consist of other objects such as wheels, doors, etc.) Usually an object has a defined physical representation, either a specific shape, texture, colour and/or other qualities that separates the object from the environment.

Object recognition can be divided into three basic levels; low, intermediate and high level processing[GW93]. (Fig. 3.1.) In the low-level domain, the preprocessing of the image is done. That includes sampling, noise reduction, deblurring etc. These are algorithms that can be adaptive in some degree, but they don’t require knowledge about the objects to be recognised to accomplish their tasks. They do require some knowledge about how the image should be preprocessed though.
In the intermediate-level domain typically segmentation, representation and description are done. This requires some knowledge about the task, but there can still be applied many general algorithms. In the biological analogy, this is the image processing that is done in the visual cortex where neurons react to more complex shapes.

The last element is the high-level domain. Here are the recognition and interpretation parts. This requires very much knowledge about the task, and usually the systems used for this level is designed specific for the particular task. To investigate the biological analogy further, this can be viewed upon as the processing done in the temporal and parietal lobes, which are associated with recognition and spatial localisation of objects.

In image processing applications and algorithms, there is not necessarily a clear distinction between the different stages as outlined above.

One of the problems in object recognition is the human ability to recognise forms and shapes, and to fill in “missing” information. An example is the missing triangle in fig. 3.2. A human does not have a problem to see the “missing” triangle in the figure, but it is hard for a computer to spot this triangle. (The figure shows a modified form of Kanizsa’s triangle.)

Figure 3.1: The object recognition system. (From [GW93].)
3.2 Colour spaces

There are different ways to represent the colours in the images. Gray-scale images are the easiest to represent and to work with because there is only a value for luminance in the image. It is also easier to work with because there is only one value per pixel in the computations. Gray-scale images has typical 8 or 16 bits per pixel (bpp.)

In order to represent an arbitrary colour, we need at least three values. For such a three-value representation, the values are called tristimulus values\[Pra91\]. How these values are chosen, is a matter of expediency. The most used tristimulus representation is the Red, Green, Blue (RGB) colour space, where a colour is represented by its Red, Green and Blue spectral components. This is the normal output from a digital camera.

The different colour spaces usually contains the same information; the information that is needed to recreate the colour of the given pixel. The difference is the representation of the tristimulus values. By selecting the right representation (colour space) the image processing task can be simplified.

A colour might be represented in terms of its luminance and chromaticity, a luminance-chroma colour representation. Examples of this are “Hue, Saturation, Value (HSV)” and “Intensity, Hue, Saturation (IHS).” For HSV the Hue and Saturation channels describes the chromaticity, and Value channel describe the luminance. For IHS, the Intensity channel describe the luminance, and the Hue and Saturation channels describes the chromaticity.
One of the main advantages with the luminance-chroma representation in the HSV and IHS colour spaces, is that the representation is similar to how a human interprets a colour, and what a human refers to as a “colour” is embedded in the Hue component and not spread over three different components. The luminance-chroma representation is often used for e.g. finding human skin colours in images\[YKA02\] where the colour of interest is has little variance in the Hue component, and a huge variance in the Value and Saturation components. If this is done in the RGB colour space, it is a huge variance in all components.

This is also true for the colours in the signs. The Hue component has a small variance for the different colours, while the Saturation and Value components has a huge variance. The different colours in the traffic signs all have a small defining region in the Hue component that can be used for classification.

There are some differences between the IHS and HSV colour spaces. The transition functions are given below. The HSV colour space has a higher Value component than the Intensity in the IHS colour space. This is because the Value component is the maximum value of the RGB components while the Intensity component is the average value. Other notable differences is that the IHS mapping uses trigonometric functions (arctan) for the Hue component, and the euclidean distance between $B - G$ and $B - R$ for the Saturation. The calculation is explained in section 7.1. The HSV mapping uses the difference between the maximum and minimum components as the saturation and the Hue is calculated and mapped directly into different regions of the Hue scale. (Therefore the values 1,3 and 5 in eq. 3.8.)

The mapping from RGB to IHS is done by using the following mapping functions\[Pra91\]:

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ 0 \end{bmatrix} \begin{bmatrix} R_N \\ G_N \\ B_N \end{bmatrix}$$ (3.1)

$$H = \arctan \left( \frac{V_2}{V_1} \right)$$ (3.2)

$$S = \sqrt{(V_1)^2 + (V_2)^2}$$ (3.3)

$$= \sqrt{\frac{2}{6}} \sqrt{(B - R)^2 + (B - G)^2}$$ (3.4)

The mapping from RGB to HSV uses this mapping functions\[Smi78\]:

$$V = \max (R, G, B)$$ (3.5)

$$X = \min (R, G, B)$$ (3.6)

$$S = \frac{V - X}{V}$$ (3.7)
For both these functions, when \( R = G = B = 0 \), the Hue is undefined. For some image processing tasks it is expedient to let the Hue exist, even if it is undefined. In these cases the Hue can be e.g. set to the last defined Hue computed, or to a predefined value.

\[
H = \begin{cases} 
\frac{5+V-G}{b} & \text{if } R = V \land G = X, \\
\frac{1-V-G}{b} & \text{if } R = V \land G \neq X, \\
\frac{1+G-R}{b} & \text{if } G = V \land B = X, \\
\frac{3-G-R}{b} & \text{if } G = V \land B \neq X, \\
\frac{3+G-B}{b} & \text{if } B = V \land R = X, \\
\frac{5-B-R}{b} & \text{otherwise.}
\end{cases}
\] (3.8)

\[\text{For both these functions, when } R = G = B = 0, \text{ the Hue is undefined. For some image processing tasks it is expedient to let the Hue exist, even if it is undefined. In these cases the Hue can be e.g. set to the last defined Hue computed, or to a predefined value.}\]

### 3.3 Noise and artifacts

There are many types and sources of noise in images. Two of the noise sources in the photo detector are photon noise and thermal noise[Sol01]. The photon noise is also called shot noise, and is due to the random arrival of photons at the individual pixels in the sensor. This noise gives the upper bound for the Signal-to-Noise Ratio (SNR) at high signal levels in the sensor. This noise can be reduced by frame averaging and/or spatial median filtering. This is usually not a problematic noise source, because the SNR is quite high.

Another noise source in an image sensor, is the thermal noise, or dark current noise. This noise is a result of leakage currents in the photo diodes, also called dark current. This dark current is dependent on the temperature of the photo diodes. Thermal noise is the one of the main noise sources that limits the SNR at low signal levels. This is a more problematic noise source, especially in dark areas of the image, because the SNR can be very low with low signal levels.

For most photo detectors, and as long as the signal level is not very small, this noise can be modelled as additive Gaussian noise.

The SNR is usually better at higher signal levels, and much is done by the chip producers to minimise the noise. But at low signal levels (dark pixels) the SNR is often quite low, and must eventually be taken into account when processing the image.

A source of static artifact, is dead pixels and warm pixels. These are malfunctioning pixels in the sensor array. A dead pixel is always black, and a warm pixel can be white or another value.

Lossy compression, such as JPEG conversion, is the source of some artifacts in images. Lossy compression is popular for natural images, since it compresses better than lossless compression algorithms. The lost information is usually information that a human don’t notice very well anyway.
3.3 The blocking effect and colour bleeding in JPEG compression.

Black and white are both red, since the red Hue is wrapped around at 255/0. The rectangular blocks of the same colour (grey tone) is a result of the blocking effect. The red areas stretches beyond the rim of the sign, and is called colour bleeding.

Many inexpensive cameras have the images converted to JPEG before they are stored in the internal memory. This is because memory is expensive, and by compressing the image before it is stored, the cost can be cut.

Two of the artifacts found in images compressed with a lossy compression algorithm, such as JPEG, are the blocking effect and colour bleeding[SW04]. The blocking effect is due to the quantisation of individual blocks in the image, and colour bleeding is due to the suppression of high frequency coefficients in the chroma components. These effects are shown in fig. 3.3.

A human viewer doesn’t see these artifacts very well, since a human is not good at noticing the details in the chromatic components. A computer on the other hand is very good at these details, and these compression artifacts might be a problem if not taken into consideration.

3.4 Low-level processing

There are many low-level processing methods that exist, e.g. noise reduction, normalisation of values, edge detection etc. The common trait of the preprocessing methods is that the preprocessing suppress or enhance cer-
tain qualities in the image. The qualities that are enhanced or suppressed are dependent on the methods used later in the system.

### 3.4.1 Noise reduction

One of the fundamental preprocessing operations that are performed, are noise reduction. This operation should remove much of the shot noise and Gaussian noise in the image. In natural images, there are several sources of noise in the image acquisition process, and the noise might reduce the quality of the image processing by a considerable amount.

In order to minimise different noise patterns, different methods are needed. The typical shot noise can be reduced by an averaging neighbourhood filter or a median filter. These are spatial domain filters, that is, they operate in the spatial domain as opposed to e.g. Fourier filters that operate in the frequency domain.

Spatial filters may be of different sizes and with different convolution kernels. The general kernel of a typical spatial low pass filter is:

\[
H = \left[ \frac{1}{b+2} \right]^2 \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}
\]  

(3.9)

For \( b = 1 \) and \( b = 2 \) this gives the following commonly used kernels:

\[
H_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

(3.10)

\[
H_2 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}
\]

(3.11)

These filters remove high-frequency components in the image. The problem with the kernel in eq. 3.10, is that it “smears” out much of the details in the image. This is better in eq. 3.11 because the centre pixel is given more weight than the peripheral pixels.

Other methods for removing high frequency noise, is a non-linear method called median filtering. This sorts the components in the filter window and picks the median as the new value for the centre pixel. This method does not introduce “artificial” colours in the image, and it removes single pixel noise efficiently.

### 3.4.2 Luminance adjustments

If the image is acquired in a controlled environment (e.g. a production line) the light can be optimised for the image acquisition process. But often
the environment can’t be controlled, and the specific objects or qualities of interest in an image can be either too dark or too bright. The brightness (or luminance component) of the image can be adjusted in order to enhance the objects of interest. For a grey scale image, the luminance is the grey level. For a colour image, the luminance is found by translating the image to a luminance-chroma colour space (e.g. HSV.)

There are several functions that can be used in order to adjust the luminance. If it is known that the object is very dark, the transfer function for the luminance can be e.g. a cubic root function, \( A = \sqrt[3]{x} \), where \( A \) is the adjusted luminance component, and \( x \) is the original luminance component. Figure 3.4 shows a dark image, and the result of cubic root scaling.

The transfer function in this case is static, and will result in an even brighter image if the image is overexposed. So a better approach would be if the transfer function could be dynamic, and change according to the brightness of the image.

One of the disadvantages with such mappings, is that this is a many to one mapping of the values, and results in a loss of information in the image. Sometimes the lost information is not relevant for the rest of the image processing task, and the relevant information is emphasised.
3.4.3 Histogram equalisation

Histogram equalisation is a method for adjusting the brightness using a dynamic transfer function. The transfer function is calculated from the histogram of the original image. Peaks in the histogram will be stretched over a wider span of values, and flat areas in the histogram will be joined. This creates a new image with a more equal distributed histogram, and this tends to enhance the edges in uniform objects (or areas) in the image.

In order to calculate the mapping, or transfer function, between the original luminance level for each pixel, $F(x, y)$, and the new luminance level $G(x, y)$, the histogram of the first image is calculated [Pra91].

The next step is to create a table of the accumulated histogram values. This table is scaled such that the range of the values falls within the range of the grey tones, usually $[0, 255]$. This scaled table is then used as the mapping between the grey values. Figure 3.5 to 3.7 shows the mappings and histograms of these steps.

The last step is then to create the new image, given the original image and the transfer function.

For a colour image the same is done with the luminance component in a Luminance-Chroma colour space.
Figure 3.6: The transfer function. Note that the steep slopes in the transfer function corresponds with high peaks in the original histogram.

Figure 3.7: The histogram equalised image segment.
Adaptive histogram equalisation

One of the problems with the histogram equalisation method described above, is that this method only uses the histogram for the whole image to create the transfer function. If there are several large areas with low contrast within the areas, the contrast enhancements will be limited to a short interval on the grey scale.

The solution to this is to use an adaptive histogram equalisation. This method uses the histogram for a smaller area in order to calculate the transfer function. This way, an area another place in the picture will not interfere with the transfer function for this area.

One method is to use a sliding window, so that each pixel has its own transfer function calculated from the histogram of the surrounding pixels. This is very computer intensive, and is not very usable for a fast processing.

A method introduced by Pizer et. al. overcomes this limitation by calculating the histogram around grid points in the image. The transfer functions for pixels between these grid points are then just a linear interpolation between the transfer functions of the surrounding grid points [PAA+87]. The histogram window might or might not be wider than the grid point spacing. This method is not as computation intensive as the sliding window method described above. If the window size is about the same as the grid point spacing, it is about the same amount of calculations as the static histogram equalisation method. There is however an overhead connected to the interpolation between the 4 nearest grid points and their mappings.

3.5 Intermediate level processing

Intermediate level processing is the stage where the image is segmented and described. The segmentation can be as simple as a thresholding of the results from the preprocessing stage, or it might be a more complex algorithm. The description of the segmented image is a labelling of the areas found in the segmentation.

3.5.1 Classification and segmentation

Segmentation is to divide the image into regions (or segments) that shares one or more features. Haralick and Shappiro stated that the “regions of an image segmentation should be uniform and homogeneous with respect to some characteristic such as grey tone or texture, Region interiors should be simple and without many small holes. Adjacent regions of a segmentation should should have significantly different values with respect to the characteristic on which they are uniform. Boundaries of each segment should be simple, not ragged, and must be spatially accurate [HS85].”
There are many features that can be used for segmentation, e.g. luminance information, colour information, texture, region growing, etc. The methods for segmentation of these features does also span a huge range from static thresholding, adaptive thresholding, region growing, etc.

Classification is to assign a class to an element of some kind. The element might be a pixel, or a region found by segmentation, or something entirely different. A pixel can e.g. be classified based on its colour information.

The results for a classifier is usually given as a confusion matrix. This is a KxK matrix, where K is the number of classes. Figure 3.1 shows a confusion matrix. From this matrix, the True Positive rate (TP) and True Negative rate (TN) can be calculated as:

\[
TP = \frac{d}{c + d} \quad (3.12)
\]
\[
TN = \frac{a}{a + b} \quad (3.13)
\]

There are also three other values that are calculated from these, the precision, \(P\), that is the ratio of predicted positive classifications that were actual positives, and the geometric mean (g-mean_1 and g-mean_2):

\[
P = \frac{d}{b + d} \quad (3.14)
\]
\[
g - \text{mean}_1 = \sqrt{TP \cdot P} \quad (3.15)
\]
\[
g - \text{mean}_2 = \sqrt{TP \cdot TN} \quad (3.16)
\]

If we assume that the classification is not perfect, there will always be pixels that are classified erroneously. If more pixels are classified as positive, more pixels will be false negatives and vice versa. A good classification method will therefore minimise the number of erroneously classified pixels. It is not certain that all pixels will be classified correctly, even with the best classifiers practically possible. It is therefore a trade off between how many false negatives versus how many false positives that are better for the given task. For some tasks both false classifications should be minimised, and for other tasks it might be better if there are more false positives than false negatives.
3.6 Digital image processing in hardware

Many image processing methods and systems are very computing intensive. The methods are often parallelisable since they often operate in different data independently. E.g. grey scale amplitude thresholding can be parallelised into a number of tasks equal to the size of the image.

General purpose processors today have a quite good performance when it comes to SIMD instructions. (Single Instruction, Multiple Data.) But a general purpose processor is somewhat limited, in that it has to decode each instruction, and the SIMD registers are usually a bit limited. The SSE2 registers in the Intel architecture are 128 bits wide, this means that a single instruction can operate on 4 32 bit integers at the same time.

If implemented in e.g. a FPGA, the number of simultaneous calculations can be even more. The FPGA can also do different calculations simultaneously. For a real time system, a hardware implementation can therefore be advantageous.

3.7 Evolvable methods used in digital image processing

3.7.1 Noise reduction

Sekanina has done some research in evolving spatial filters with evolvable hardware: One approach, was to evolve spatial filters using Cartesian Genetic Programming (CGP) (see section 2.4) and evolution at the functional level[Sek02, Sek04b]. The filters were evolved by using a development consisting of images with a particular noise scheme; a predefined amount of noise of a predefined type. (E.g. Gaussian noise with a mean of 0 and standard deviation of 16, or shot noise with probability of 0.08 for a corrupted pixel, either completely white or completely black.) The fitness value is compiled based on the difference between the filtered image and the original. The evolution was performed in hardware, using a system as depicted in fig. 3.8.

The CGP configuration that was used in the experiments performed is given in table 3.2. Table 3.3 shows the functions that were tested. The Cartesian Program used by Sekanina was slightly modified version of Millers original program. Only the nodes in columns \( \leq l \) can be connected to the input nodes in Sekaninas version, as opposed to Millers where all nodes can connect to the input nodes.

The inputs for the CGP were a 3x3 window that was passed over the noisy image.

The EA used in the experiments was a form of \((4 + 12)\)-ES, where the 4 best individuals in each generation were chosen for reproduction, and
Figure 3.8: System used by Sekanina for evolving spatial image filters. (Figure from [ST02].)

Table 3.2: The configurations of the CGP used by Sekanina.

<table>
<thead>
<tr>
<th>n_i</th>
<th>n_o</th>
<th>n_n</th>
<th>l</th>
<th>n_f</th>
<th>n_r \cdot n_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>1, 2, 4 or n_c</td>
<td>8 - 32</td>
<td>20-60</td>
</tr>
</tbody>
</table>

Table 3.3: The functions used by Sekanina in [Sek04b].
the 12 other genomes were mutations of these four. The mutation rate was fixed at 2 mutations per genome. The mutation was performed by mutation of either the input address or the function, but not both, in the given node.

The evolution was run for 500000 generations, or was stopped if no increase in the maximum fitness was detected for 50000 generations.

The results indicated that the evolved filters could match the conventional filters. One of the evolved filters could match a conventional median filter, but with an order of magnitude less equivalent gates.

The EHW design was also tested with a dynamic environment. The filters were evolved for a given noise scheme, and then applied and continued evolution under another noise scheme. The results from these experiments showed that the evolved filters showed that at least one of the evolved filters had a lower difference between filtered image and noiseless image than the best conventional filter for every noise scheme, and one of the evolved sequences was better in average than any of the conventional filters. The results also showed that a good filter design under the time constraints in the experiments was probably dependent on a good individual in the original population.

Another approach Sekanina followed, was to evolve a median filter using CGP and a Compare-Swap approach [Sek04a]. In the first experiment, the median filter was evolved by CGP and by utilising only \( \text{min} \) and \( \text{max} \) operators. The results showed that it was difficult to evolve perfect median filters for filter sizes above 5 inputs, no optimal solution with regard to number of subcomponents were obtained for filter sizes 9x9 and 11 inputs, and no perfect median filters were found for filter sizes 13 inputs and above.

The compare-swap approach uses a network of compare-swap elements. The chromosome in this case is a sequence of compare-swap elements. The chromosome was chosen to be sufficiently large for a given filter size. When a perfect median filter was found, the maximum length of the chromosome was decreased by two, and the evolution continued. The results showed that the evolved filters found solutions with the same amount of compare-swap elements as the best conventional filters for sizes up to 11 inputs. The evolved filters also found solutions for filter sizes up to 25 inputs in contrast to the CGP approach that could only find solutions up to and including 11 inputs. Figure 3.9 shows one of the evolved median circuits.

The median circuits were also tested if the number of components was reduced below the lowest number of components for a perfect circuit. This showed that the circuits still operated correctly for the majority of input sequences. This can be utilised in e.g. image filters, where it is not important that the filter is perfect since the a human will not discern between a perfect and near-perfect filter.
Figure 3.9: One of the evolved sorting networks for 13 inputs. The horizontal lines are the signal path, and the vertical lines are a compare-swap between the signals. The signal in the middle on the right is the median. (From [Sek04a].)
Chapter 4

Intelligent Transportation Systems

4.1 Intelligent Transportation Systems

4.1.1 What is ITS?

Intelligent Transportation Systems (ITS) is an umbrella term for technology designed for more effective and safer multi-modal transportation systems. The term “multi-modal” in ITS means that all kinds of transport are included in the term. (Land transport, boat, air transport, etc.) ITS includes technology for both vehicles and infrastructure.

One of the first coordinated European research programme in ITS was called PROMETHEUS (“PROgraMme for a European Traffic with Highest Efficiency and Unprecedented Safety.”) This project was launched in 1986 as a collaboration between European car manufacturers, and the main objectives were to reduce road accidents and to improve traffic efficiency[Wil92]. The project ended in 1994.

ERTICO\(^1\) is a European non-profit organisation that is working for implementation of ITS in Europe, and it was founded in 1991 by the European Commission, key members in the ITS industry and national governments. Their vision for 2010 is to increase the effective road capacity by 20% without new constructions, to reduce the causalities by 50%, save travel time amounting to one year over an average lifetime, to significantly reduce the vehicle CO\(_2\) emissions and to have a market for ITS equipment and services worth 21 billion euros. Among the partners, we find the Norwegian Public Road Administration (Statens Vegvesen), BMW Group, Daimler-Chrysler, Fiat etc. Other countries/regions have their own organisations; in USA

\(^{1}\)http://www.ertico.com
there is ITS America\(^2\), in Japan there is ITS Japan\(^3\), and other countries have their own organisations.

In Norway there is a new organisation called ITS Norway. It was started in July 2004, and is the national equivalent of ERTICO. Its purpose is to establish an ITS community in Norway, to facilitate research, development and deployment of ITS systems and to have means of communication between private companies, research institutions and the government.

4.1.2 ITS research

There has been a number of research projects in Intelligent Transportation Systems, both in Europe, USA and Japan and elsewhere in the world. In this thesis, the focus will be on the image processing research.

ARGO

One of the most spectacular ITS project has been the ARGO vehicle, coordinated by Prof. Alberto Broggi at the University of Parma, Italy.\(^4\) In this project, the team built a car that was able to drive autonomously without much interference from the driver. The first part of the project ended in 1998 with a 2000 km tour through Italy, where the vehicle drove autonomously between 85 and 95 percent on the different legs[BBF99].

The ARGO vehicle uses a system called Generic Obstacle and Lane Detection (GOLD) for autonomous driving. This system consist of the data-acquisition, processing and output subsystems.

The data-acquisition system uses two cameras placed at the top corners of the windshield and a speedometer for recording the current speed. The placement of the cameras is chosen in order to have a maximum distance between them to facilitate stereo vision. The cameras are small low-cost cameras that have a resolution of 320 lines, and they can be synchronised by an external signal. The cameras were calibrated by using a grid with known size on the ground.

The processing system consists of a standard 200 MHz MMX Pentium processor. The MMX capabilities were used in order to speed up the processing. The team has also developed a special purpose system, called PAPRICA-3 (PArallel PRocessor for Image Checking and Analysis, version 3.)

The output system consist of an actuator on the steering wheel for autonomous steering, loudspeakers for warning the driver about dangerous

\(^2\)http://www.itsa.org/
\(^3\)http://www.its-jp.org/
\(^4\)The homepage for the ARGO project is located at http://millemiglia.ce.unipr.it/ARGO/english/index.html
situations, a monitor that shows the left image with lane markings and obstacles displayed, and a control panel that shows the vehicle’s offset from the centreline of the road.

The driver has the possibility to select one of three driving modes:

*Manual:* The system monitors the driving and warns the driver if potential dangerous situations are detected.

*Supervised:* The system monitors the driving and takes control in situations that it recognises as dangerous.

*Automatic:* The system drives without intervention of the driver. The system can change lanes and avoid obstacles.

If the driver sees that the vehicle will do dangerous manoeuvres, the driver can override the system and take control with an emergency button.

GOLD is a system that can find lane markings and obstacles, and steer the vehicle clear of obstacles and hold the vehicle on the road based on the lane markings. After the images are grabbed from the cameras and loaded in the RAM of the computer, the detection is done in 3 stages.

First the perspective is removed in order to get a 2-D birds-eye view of the road ahead. This is done by a method called Inverse Perspective Mapping (IPM)[BBF98b]. This method assumes a flat road, and marks the deviations from the flat road.

The second stage in the process is to segment the lane markings. It is assumed that the markings are brighter than the surrounding road and that the markings forms approximately vertical and constant-width lines in the remapped image. The image is enhanced through some iterations of geodesic morphological dilation. This is an algorithm that take advantage of the vertical correlation between the markings. Because of different illumination conditions, the system uses a adaptive threshold in a 3x3-pixel neighbourhood.

The result is then thinned and the image is scanned row-wise and non-zero pixels are joined into a line. The lines are then approximated and joined to form a *polyline* that contains two or more of the lines. The polylines are joined themselves if the distance between them is short enough and the orientation is the same. The line that best approximate the previous processed line is chosen as the centreline. This gives the system a good reliability in that it can detect the lane markings even if they are partially occluded. The processed image can then be used for computing both the road geometry and the vehicles position.

The knowledge about position and geometry is used for the automatic steering and warnings. The system had no difficulties in processing 25 stereo images pr. second.

The experience from the MilleMiglia in Automatica tour showed that the system performed well, but there were some cases where the system
could not drive autonomously. In these cases the system changed to manual driving and alerted the driver. The system was robust under different lighting conditions, but it had some problems with fast changing intensities like driving in and out of tunnels. The team believes that this should be better with more expensive cameras with a faster gain control. Other problems were reflections in the windshield that rendered the acquired images over-saturated. The vehicle did tend to oscillate inside the lane at high speeds (>95 km/h.)

The ARGO team has also developed some extensions to the system, namely the use of stereo inverse perspective mapping[BBF98b], where IPM is used for simplifying the search for homologous points in the stereo-images. Another extension to the system is called Extended Inverse Perspective Mapping (EIPM), and is an extension that removes the flat road assumption from IPM by using stereo images[BBF98a].
Chapter 5

Traffic sign detection

5.1 Norwegian speed limit signs

There are some rules regarding the speed limit, and the layout of the signs in Norway. I will in this chapter give an overview over these topics.

5.1.1 The sign standard

The technical specifications of the traffic signs in Norway are given in “skiltnormalen” (the sign standard.) This is a detailed standard spanning several hundreds of pages. It is under revision, and will be updated the next couple of years. This standard is based on both international regulations and scientific research.

For a circular sign, there are three sizes that are used. The smallest signs have a diameter of 60 cm. These signs are typically set up in built-up areas where the speed limit is low. The medium sized signs have a diameter of 80 cm. These signs are typically placed on roads where the speed limit is higher. The largest signs have a diameter of 100 cm, and are placed on highways where the speed is very high.

The sign standard does also give the minimum distances in which the signs should be seen. (i.e. the distance between the sign and obstacles that hides the sign.) These distances are given in table 5.1. It should be noted that there may exist special circumstances and/or environments such that these distances may be shorter. The distance between signs in the same direction should be at least 50 metres, and preferably 100 metres.

The font used on the signs is called “trafikkalfabetet” (the traffic alphabet.) This font is developed for use in Norwegian traffic signs, and all Norwegian signs with text use this font. The font is designed to be clear and easily read by a human under all conditions.

The signs uses a reflective film and colours that make them easy to spot and read, and gives the signs the best possible contrast. It should be noted
that the reflective film and colours do wane with the age of the sign.

### 5.1.2 Speed limit signs

Fig. 5.1 to 5.4 show the signs that are relevant for speed limits. (The captions in the figures are the numeric codes for the signs.) There are 7 speed limit signs, 5 signs that mark end of a particular speed limit, one speed limit zone sign, and one end of speed limit zone. There are also some informational signs that decides the speed limit.

In addition to these signs, there is also a sign with a speed limit of 100 km/h, presently being tested. There is also signs that are made by LEDs (light emitting diodes.) These consist of red LEDs around the rim on a black background and with the white LEDs that form the digits. This type is used in tunnels and other places where the speed limit may be variable. These signs are in a group called “variable signs” due to their ability to change. These are also specified in the sign standard. There are also private signs that are set up on e.g. parking lots and private roads. Fig. 5.5 is a private sign. These private signs do not resemble any official sign.
Figure 5.2: End of particular speed limit

Figure 5.3: Speed limit zone

Figure 5.4: Informational signs

Figure 5.5: A private, two-coloured sign
5.1.3 Rules governing the speed limit

The default speed limits on Norwegian roads are given in “vegtrafikkloven”\(^1\) § 6[Lov04]. For built-up areas the speed “must not exceed 50 km/h,” and elsewhere it “must not exceed 80 km/h.” The signs in fig. 5.4(a) and 5.4(b) denote a yard, and the speed should be held at an minimum within this area. The signs in fig. 5.4(c) and 5.4(d) denote a built-up area, and the rules for driving in a built-up area should be followed.

The law also states that the driver “shall adapt the speed of the vehicle according to the local conditions, road conditions, visibility and traffic conditions [...]” and it continues; “the driver shall always be in full control of the vehicle.” This means that even if the speed limit is 80 km/h, other conditions may lower the actual speed.

5.2 Traffic sign detection

There has been some work on traffic sign detection algorithms. There has been quite a plethora of methods used. I will give a short overview over some of the research done in this area. The theoretical background for the most common digital image processing methods are given in section 3.1. Most of the proposed ideas starts with a colour segmentation algorithm.

Most of the work on traffic sign detection have used the colour information as a basis for the detection system. The colour information is valuable information in traffic sign detection, and studies has shown that the colour information increase the detection rate[PN00].

5.2.1 Traffic sign detection for Norwegian signs.

Sekanina et. al. developed a system for recognition of Norwegian speed limit signs, called Image Detection of Speed Limits (IDSL)[ST02]. This method consists of three stages as depicted in figure 5.6. The first stage is preprocessing and colour segmentation, where the image is segmented into three new binary images that represent the red white and black regions. The second stage is the localisation of the speed limit sign by template matching. The last stage is interpretation of the speed limit information in the sign.

If the average pixel value in the image is very dark, the image is brightened. This is done by first checking the average value. If this value is below a threshold, \( T_{\text{dark}} \), all channel values for the pixels are increased by the difference between the average and threshold value.

Then the image is segmented into three binary images; one for each of the three colour classes, \( \text{RED} \), \( \text{WHITE} \) and \( \text{BLACK} \). The segmentation

\(^1\)The road traffic act
method operates in the RGB colour space, and the classification rules are given in table 5.2. Notice that a pixel may be a member of more than one of the colour classes.

The red areas in a speed limit sign are surrounding a white area in the middle of the sign, and to isolate these some spatial rules for the classification are applied: A pixel is only considered WHITE if it belongs to a 2x2 group of WHITE pixels. RED pixels are found by looking at all 2x2 clusters of WHITE pixels, and search for neighbouring RED pixels. If there is, the rest of the RED pixels are found by growing the RED region, i.e. mark the surrounding pixels recursively as RED if they fit into the RED class in table 5.2.

The output image from the previous stage will now be an image of RED pixels that have WHITE neighbours in the original image. Circular shapes are then found by a template matching algorithm. All templates are moved over the entire image, and for every pixel that are set (i.e. having a RED colour) on an even row and column, the percent of matching RED pixels are computed, and is set as the score. If the score is lower than 50 %, the area is discarded. If there are some RED pixels in the middle of the sign, the score is set to 0. This is done in order to avoid signs with a red crossing
line. The same happens if the RED pixels are not distributed symmetrically. If there are more than 95% or less than 10% WHITE pixels, the score is also set to 0. If this score is better than previous scores, the best score is updated, and the search continues.

The last stage consist of detecting and interpreting the digits in the potential sign. This is done by first removing the red and other surrounding colours in the area defined by the template. Then the width and height of the numbers are determined, and the first digit is down-scaled to 4x6 pixels. This matrix is then used for identification of the digit. The reason for only using the first digit, is that the second digit is always zero.

Bakke made some improvements to this algorithm [Bak04]. These improvements includes some optimisations in the implementation, has expanded the algorithm to look at image sequences, and done adjustments based on a bigger test-set with a huge span in the image quality. The light adjustment algorithm has been made to also adjust very bright images. The implementation of the light adjustment adjust the threshold values in table 5.2 instead of the individual pixel values. If the image is too bright or too dark, the amount of adjustment is the absolute value of the difference between the average value and a preselected “adjustment constant.” This adjustment constant is set to 131 for dark images and 141 for bright images. The new threshold values are then given by subtracting (for dark images) or adding (for bright images) this amount from the normal threshold values.

The recognition rate for the new data set increased from 50% to 90%. The improvements to the algorithm included; a test for yellow pixels, a better light adjustment algorithm for very bright images, earlier rejection of templates based on the non-existence of black and white in the middle of the area and red colours to the left side in the area.

5.3 Other colour segmentation algorithms used for traffic sign detection

A. de la Escalera et. al. describes a method for traffic sign detection in three stages [dlEMSA97]. First there is a colour segmentation stage that operates in the RGB colour space. The next stage is corner detection and extraction, where a sign is located by looking at matching corners. The corner detection is done by convolution masks designed for corner correlation. For circular signs, the same method as the corner detection is used, but the convolution masks are designed to find segments of a circle instead of corners. The last stage is a classification stage that uses neural networks for classification of the different sign classes.

The colour segmentation is done by using the ratio between the colours in the RGB colour space. By using the Red component as a reference, the
following new colour channels are used:
\[
\begin{align*}
R_n &= R \\
G_n &= \frac{G}{R} \\
B_n &= \frac{B}{R}
\end{align*}
\]
Where the subscript, \( n \), denotes the new channel value, and \( \{R, G, B\} \) are the red, green and blue values for the pixel.

These values are then thresholded by using static, global threshold values. It is mentioned that the thresholding can be implemented using a 16-bit lookup table. There is no mention of what threshold parameters that were used.

A later work by de la Escalera et al. use another approach to traffic sign detection\[dlEAM03\]. This starts with a colour segmentation algorithm that uses the HSI colour space, followed by a edge detection algorithm. The next step is to find shapes that resembles a sign. This is done by template matching, where the template is rotated, scaled and transposed in order to find the best match. The last step is the interpretation of the sign, and this is done by a neural network.

The first stage in the approach is the colour segmentation method. The interesting approach here is that they use a HSI colour space, and have a transfer function that generates two lookup tables, one for the hue and another for the saturation channel. This transfer function would emphasise red, saturated values, and gradually suppress less red hues and less saturated areas.

The transfer functions are as follows:
\[
\begin{align*}
H(i_h) &= \begin{cases} 
255 \frac{i_{s,min} - i_s}{i_s,min} & 0 \leq i_s \leq i_{s,min} \\
0 & i_{s,min} \leq i_s \leq i_{s,max} \\
255 \frac{i - i_{max}}{255 - i_{max}} & i_{max} \leq i \leq 255
\end{cases} \\
S(i_s) &= \begin{cases} 
i_s & 0 \leq i_s \leq i_{s,min} \\
255 & i_{s,min} \leq i_s \leq 255
\end{cases}
\end{align*}
\]
where \( i_h \) and \( i_s \) is the value of the corresponding channel, \( H(i_h) \) is the hue emphasis and \( S(i_s) \) is the saturation emphasis.\(^2\) These two values are multiplied and normalised to a maximum value of 255. The resulting image is then handed over to the recognition part. Figure 5.7 shows graphical representation of the transfer functions.

For the interpretation stage, the same colour segmentation method as was used in [dlEMSA97], was used for creating input to the neural network.

\(^2\)The original paper states that the transfer function for \( H(i) \) when \( i_{max} \leq i \leq 255 \) is \( 255 \frac{i - i_{h,max}}{i_{h,max} - i_{h,min}} \), but this would result in a value below 255 for \( i = 255 \). This is probably a typo in the original paper.
P. Paclik et. al. have also done some studies in road sign detection, using Laplace kernel classifiers[PNPS00]. This approach starts with a pixel based colour segmentation, which is performed by thresholding in the HSV colour space. This classifies the pixels into 6 different colour classes, namely; white, black, red, blue green and yellow. White and black were thresholded using the Value component, while the other four colours were thresholded using the Hue component. The threshold values were chosen by tuning the threshold values in accordance with real-world traffic signs under different illumination conditions. Unfortunately, there is nothing mentioned about the threshold values used in the experiments.

The next stages consist of extraction of spatial moments, and these moments are used as a feature vector to a kernel classifier system.

The experiments performed showed that this approach gave good detection results, even for hard classification problems, but one of the disadvantages with using kernel classifiers is that the whole training dataset is used for each computation of the probability density. Since the training set is supposed to grow while the system is operational, this means that the computation of the probability density will use more and more resources.

The group has also done some research on using grey scale images instead of colour images. With grey scale images the spatial moment analysis is done directly on the grey scale image without segmentation[PN00]. By using grey scale images, the result was slightly lower than with colour images. (The error rate was 4 percent for the grey scale images while it was about 1 percent for colour images.)
Piccoli et. al. use a system that first selects a region of interest based on localisation and colour information, then a geometrical analysis of these regions in order to find sign shaped regions, and at last a template matching of the inner regions of the sign[PMPC96].

The colour segmentation is done in the HSV colour space, and pixels are classified as red if the hue is $0^\circ \pm 30^\circ$ and the saturation is greater than 20%. The 512x512 image is then split into 16x16 regions and each region is classified as red if there is more than 80 red pixels in the region. Adjacent red regions are then joined into clusters, and the bounding box around the clusters defines the search regions for the geometrical analysis.

Hsu et. al developed a system that consisted of two main phases; detection and recognition[HH01]. The detection phase consists of colour segmentation, template matching and geometrical reasoning. The recognition phase uses Matching Pursuit filters for classifying the signs found in the first phase.

The colour classification is done in a chroma-luminance colour space. The exact colour space used is not mentioned, but it can be inferred that it is HSV or a similar colour space. The global threshold values used is $\pm 30^\circ$, if red hue is given as 0, and the Saturation should be at least 20%. These values are equal to the values used by Piccoli et. al. in an earlier study[PMPC96].

Vitabile et. al. have done some research in traffic sign detection, and they have developed a system called Automatic Road Signs Recognition System (A(RS)$^2$)[VS98]. This system consists of a colour segmentation stage, a shape recognition stage, and a sign interpretation stage.

The colour segmentation is done in the HSV colour space, and instead of using one set of threshold values, the Value component is split into 6 segments, and each of the different Value segments has its own Hue and Saturation threshold values. There is no information about the specific threshold values other than that they were chosen.

The next stages are template matching in order to find the shape of the sign, and a neural network for interpretation of the sign.

In a later study, the colour segmentation algorithm is refined to include a region growing algorithm for colour segmentation[VPPS01, VGS02].

The first step in this segmentation approaches is to convert the image into HSV colour space. The image is then filtered using “theoretical” values of the signs. I.e. the image is first segmented using global thresholds in order to find interesting regions.

The original image is then split into segments that are 32x24 pixels, and for each of these segments, the average values of the pixels found in the
first segmentation step are calculated, and used as a basis for the region growing algorithm.

The threshold values for deciding similarity in the growing algorithm, are given as the Euclidean distance in the HSV colour space from the average values. This distance is calculated by the following equation:

\[ a = k - \sin(s_{\text{seed}}) \]  \hspace{1cm} (5.6)

where \( a \) is the threshold distance, \( k \) is a normalisation parameter and \( s_{\text{seed}} \) is the average saturation value found in the previous step.

N. Kehtarnavaz and A. Ahmad developed a system for traffic sign detection that first transforms the image to the YIQ colour space, and uses a neural network for segmentation. The next step is to recognise and interpret the sign in a neural network. As input to this NN, the objects are translated to a polar coordinate system. (This method was originally used by Kang et al. [KGK94].)

When selecting colour space to use, several colour spaces were considered. These were \( uvY, XYZ, YIQ, ISH, RGB \) and \( uvh \). Of these, the YIQ colour space gave the most compact and separated colour clusters than the other spaces.

These colour components were then fed into an ART2 Neural Network (NN). This network found all colour signs, but it had problems with black and white signs. (This behaviour was also present in all other colour spaces.)

L. Estevez and N. Kehtarnavaz introduced another approach to colour segmentation[EK96]. The approach consist of four stages; Colour segmentation, edge detection, edge analysis and recognition based on the edge analysis.

In this work, they point out the problems with colour bleeding and desaturation effects, and that these effects will compromise the quality of edge detection. Instead they use a sparse colour classification routine, that classifies only every fourth pixel. The equation used for this step is:

\[ R_n = R - \max(G, B) - \alpha|G - B| \]  \hspace{1cm} (5.7)

where \( R_n \) is the redness of the pixel; \( R, G \) and \( B \) are the Red, Green and Blue values of the pixel and \( \alpha \) is a sensitivity factor. The threshold is then set to 0, and all positive values are interpreted as RED.

In order to find edges, each segmented pixel is XOR’ed with the previous classified pixel. If an edge is detected here, the algorithm continues to locate the edge more precisely. This is done by looking at the eight pixels surrounding the detected edge. For each pixel, the RGB difference is calculated with the following equation:

\[ E_p = |R - R_p| + |G - G_p| + |B - B_p| \]  \hspace{1cm} (5.8)
where $E_p$ is the difference for the previous pixel, $R$, $G$ and $B$ is the Red, Green and Blue channels respectively. The $p$ subscript indicates the previous pixel, i.e. the pixel at location $x - 1$, for the pixel located at $x$. The maximum difference value, $\max(E)$ is then selected as the edge.

J. H. An and T. Y. Choi uses a method called “dominant colour transform” for their traffic sign detection algorithm[AC98]. This segmentation algorithm first calculates the difference between the maximum channel value and the medium channel value. If the difference is larger than a given threshold value, the dominant colour value is used as the new pixel value and the other channel values are discarded. If the difference is smaller than the threshold, the pixel is set to a predefined value, e.g. grey or black. This can be summarised with the following equation if $\max(R, G, B) = R$:

$$F = \begin{cases} 
  (R, 0, 0), & \text{if } \max(R, G, B) - \text{med}(R, G, B) \geq T_0 \\
  (c, c, c), & \text{else}
\end{cases}$$

(5.9)

where $F$ is the new pixel triplet; $R$, $G$ and $B$ is the channel values for the given pixel; $T_0$ is the threshold value and $c$ is a constant.

The problem with this method, was that it was difficult to find a good threshold value. A better approach might be to use an adaptive threshold value instead of the static used.

S.-K. Kim used a traffic sign detection algorithm that first used colour and texture information to find the signs, then found the shape of the sign analysing the bounding box and grid lines inside this box[Kim98]. The last step is the identification of the individual signs, which this is done with an eigenspace representation.

The colour segmentation is done by converting the RGB image to a Red-Green and a Blue-Yellow colour opponent image (see below for description,) and then classify the colours with threshold values specific for a given colour.

The transformation from RGB to R-G and B-Y colour opponent images is given by the following equations:

$$L(x) = 105 \log(x + 1 + n)$$

(5.10)

$$I = L(G)$$

(5.11)

$$RG = L(R) - L(G)$$

(5.12)

$$BY = L(B) - \left( \frac{L(G) + L(R)}{2} \right)$$

(5.13)

where $n$ is a noise factor with uniform distribution in the range $[0, 1)$. The constant 105 in eq. 5.10 is a scaling constant chosen such that the function
Figure 5.8: The RG and BY images created by Kim’s algorithm. Note the high values for the red circle in the RG image, and the high value for the blue rectangle in the BY image.

$L$ will be in the range $[0, 254)$. The RG and BY images are thresholded in order to find red-orange, green, blue and yellow signs. Red areas will have high values in the R-G image, while green areas will have low values in the R-G image. Colours between red and green will have a medium value. Blue and yellow areas will have respectively high and low values in the B-Y image. Figure 5.8 shows the RG and BY images of an image from the test set. The values are scaled to fit in a range from 0-255.

The intensity image from eq. 5.11 uses only the green channel because the red and blue channels from some cameras have lower spatial resolutions. The intensity image is used for texture filtering. The texture filtering is done by first smoothing the image with a median filter, and then subtracting it from the original image. This new image is then median filtered again and passed to a filter that looks for the texture of signs.

The last steps are then to erode and dilate the thresholded images, and to label the regions.
Part II

Evolution based methods for colour classification
Chapter 6

Methods based on evolution for colour classification

Most of the different speed limit sign recognition methods use some kind of colour segmentation in order to find the sign. There are exceptions that use e.g. textures and grey scale images for the segmentation[PN00]. When a method is depending on the colour information of the object, the need for a good colour classification method is important. If the colour classification does not detect colours that are in the objects, the later stages of the object recognition system will suffer from incomplete data. In these experiments, the performance of the different colour classification methods are tested with the reported parameters, and the parameters are tuned with a GA.

In this work, two different methods that use evolution are considered. The first method is to use evolution for tuning of parameters. This method uses existing algorithms and evolves the parameters to the algorithms. An advantage to this method is that the evolution is objective with regard to which method are considered, and should therefore be able to give comparable results between the methods.

The other method is to use Cartesian Genetic Programming to evolve a good colour segmentation method. Instead of just evolving the parameters for existing algorithms, the CGP can evolve the algorithm itself. The CGP was chosen because it is a simple architecture and easy to implement in hardware.

6.1 Tuning of parameters using GA

Most of the parameters used in the literature is found by either an educated guess, or by adjusting the parameters by trial-and-error, or a combination of the two. The trial-and-error approach is time consuming, and for algorithms that have a complex interaction between the parameters it is difficult to find the optimal parameter set.
When comparing the different algorithms, it is important to compare them using the best possible parameter sets. In most research the colour classification algorithms are described in detail, but often the parameters used is not mentioned. This is probably due to the fact that most work on traffic sign detection focus on algorithms on a higher level and the colour classification is only a minor point in the work.

The idea of using evolutionary methods for tuning parameters is not new, it was in fact one of the reasons Evolution Strategies was created. The GA is used here to compare the different methods, and to get an optimised set of parameters. For methods with many parameters or complex interactions between parameters (e.g. Sekanina and Bakkes method) the GA could find solutions that are difficult to find using educated guesses and trial-and-error.

The methods that are considered for GA tuning of parameters are described in section 5.3. In the cases where the parameters used for the colour segmentation part is described, these parameters are also tested.

6.2 Using CGP for colour segmentation

The CGP framework is described in section 2.4. This architecture has been used in image processing tasks. Sekanina used this architecture for creating noise reduction filters[Sek04b, Sek04a].

The CGP architecture is a simple architecture in itself, and resembles the way FPGAs work. A logic cell in the CGP is much like a Configurable Logic Block (CLB) in the FPGA. On the other hand, the routing is more restricted in the CGP than in a typical FPGA.

The fact that CGP resembles a typical hardware implementation makes it an interesting architecture to use in real-time image processing tasks. The direct mapping from the CGP architecture to a hardware implementation makes an implementation very simple.

The EA used in this work is a simple $(\mu + \lambda)$-ES. The ES is described in section 2.3. The parameters for the ES was chosen to be $\mu = 4$ and $\lambda = 12$, which is the same as Sekanina used in his experiments. This $\mu$ does also mean that up to four different configurations can theoretically exist at any time during the evolution.

The inputs used in this CGP is a set of 15 pixels, 5 from each colour channel, and a 3x3 average value for all colours. The output is a single number, and represents RED if the value is $\geq 128$ and NOT RED otherwise. Figure 6.1 shows the inputs to the CGP.

Two sets of functions are also evaluated. The first set is the same functions Sekanina used for noise reduction in images. (See section 3.7.1.) This set of 34 functions is given in table 3.3.

The second functions set consist of 16 more or less arithmetic functions.
The functions consist of ordinary arithmetic functions and some other special functions that can be useful in image processing tasks. The functions have been crafted and selected especially for this work. These functions are at a higher abstraction layer that Sekaninas functions. The advantage is that more can be done in each logic cell, and this might reduce the size of the CGP. On the other hand, it might be more difficult to map the functions directly in hardware if the functions would take too much die space. The functions are listed in table 6.1. The hypothesis is that these functions should be able to find good solutions with a smaller Cartesian Program. Note that there are no constants in this function set.

The subscript $s$ to the operators in the function sets, is to denote a saturated operation. The operator $+_s$ will then return the value 255 if the addition overflows. The operator $-_s$ will return 0 when the result is negative. The operator $/_s$ will return -1 (that is; 255 for unsigned, 8-bit values) when the divisor is 0. The operators $<<$ and $>>$ denote a shift left and shift right respectively.

Some of the functions in table 6.1 might need an explanation. Function number 2 will return the overflow bit from an addition. Function 3 is a saturated add, which means that it will return 255 for all results $\geq 255$. Function 4 is the (positive) difference between the inputs. Function 6 is a saturated subtraction, i.e. it will return 0 for all results $\leq 0$. Function 7 and
Table 6.1: The arithmetic functions used in the CGP.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$x + y$</td>
</tr>
<tr>
<td>1</td>
<td>$(x + y) &gt;&gt; 8$</td>
</tr>
<tr>
<td>2</td>
<td>$x + s \cdot y$</td>
</tr>
<tr>
<td>3</td>
<td>$</td>
</tr>
<tr>
<td>4</td>
<td>$x - y$</td>
</tr>
<tr>
<td>5</td>
<td>$x - s \cdot y$</td>
</tr>
<tr>
<td>6</td>
<td>$xy \pmod{255}$</td>
</tr>
<tr>
<td>7</td>
<td>$xy &gt;&gt; 8$</td>
</tr>
<tr>
<td>8</td>
<td>$\max(x, y)$</td>
</tr>
<tr>
<td>9</td>
<td>$\min(x, y)$</td>
</tr>
<tr>
<td>10</td>
<td>$x$</td>
</tr>
<tr>
<td>11</td>
<td>$x / s \cdot y$</td>
</tr>
<tr>
<td>12</td>
<td>$2x$</td>
</tr>
<tr>
<td>13</td>
<td>$x / 2$</td>
</tr>
<tr>
<td>14</td>
<td>$x^2 &gt;&gt; 8$</td>
</tr>
<tr>
<td>15</td>
<td>$x \pmod{y}$</td>
</tr>
</tbody>
</table>

8 is the low and high byte from a multiplication, respectively. Function 11 is just a pass through of the first input. Function 12 is a saturated divide, and it will return 255 for any result $\geq 255$, even for division by 0.

The saturated functions are chosen because many operations are not necessarily dependent on an exact value, and these functions will not overflow and is not undefined for any permutations of the inputs. (E.g. dividing by zero is a defined value.)

### 6.3 Luminance adjustments

Luminance adjustments as described in section 3.4.2 are often performed in order to enhance certain qualities in the image. It might be to increase the luminance in dark areas or enhance the contrast in homogeneous areas.

The images in the test set are acquired under varying illumination conditions and varying environments. It is therefore difficult to know exactly how the image should be processed in order to enhance the colours in the sign. The sign might be very dark or very bright, and it might be illuminated by yellow lights, it might be dark against a light background, or dark against an even darker background. It would therefore be of help if the colour information in the signs can be enhanced.

The freedom the CGP architecture has when evolving a colour classification method can make the architecture too specific. It is therefore useful to see if the architecture can correctly classify a huge span of colours. This can be tested by using different luminance preprocessing methods. The luminance preprocessing does change the RGB values, but preprocessing does not change the ratio between the R, G and B components for each pixel.

There are therefore two properties that are tested with the luminance adjustment methods in this work. The first is to see if the colours in the sign are enhanced so that the colour classification is improved. The other is to test the robustness of the evolved colour classification methods.
Three luminance adjustment algorithms are considered here; cubic root scaling of HSV Value component, histogram equalisation of the HSV Value component and adaptive histogram equalisation of the HSV Value component.

Cubic root scaling of the Value component does increase the brightness in the image. The histogram is skewed towards higher luminance levels. This might help detecting dark colours in the sign.

Histogram equalisation does enhance the contrasts in the image, and the details in homogeneous regions is therefore easier to discern. One of the problems that can arise from this preprocessing, is when the colours in the sign are darker than the homogeneous areas. In this case the colours in the sign will be darker after the preprocessing stage.

When there are several huge homogeneous regions with different span in the Value component, the histogram equalisation will try to spread the histogram equally for the whole picture. The different regions will therefore have less contrast. This can be fixed by using an adaptive histogram equalisation method. The adaptive histogram equalisation is calculated based on a subsection of the image, and different regions in the image will have different transition tables. The contrast should therefore be better than by using the static histogram equalisation. The same disadvantage as described above does also apply for this method.
Chapter 7

General optimisations

7.1 A fast method for Value manipulation in the HSV colour space

Manipulation of the Value in the HSV colour space is used for manipulating the brightness of the image without adjusting the colour information.

The ordinary way to do this, is to first convert the image to the HSV colour space, then adjusting the Value component and convert the image back to the original colour space. The original colour space is often RGB, and this is the basis of this method. The conversion between RGB and HSV is described in section 3.2.

It is interesting to note that if the individual channel values in the RGB colour space are adjusted by the same ratio, the Hue and Saturation components in the HSV colour space are preserved.

The method described here uses 1 max operation, one addition/subtraction, 1 division and 3 floating point multiplications when manipulating the Value. The original method uses 1 max operation, 1 min operation, on average 6 comparisons if the colours are evenly distributed, 3 divisions, 1 addition and 2 subtractions for converting the image to HSV. The conversion back is about the same amount of calculations.

If other image operators are used in the HSV colour space, there is a overhead using the proposed method, but if the only operator in the HSV colour space is value modification, this method should be faster.

7.1.1 Mathematical background

The inequality between two (or three) variables are unchanged when scaled: \( x \leq y \leq z \iff cx \leq cy \leq cz \) for \( c \leq 0 \).

First, the Saturation (eq. 3.7) is invariant for scaling of the RGB values:

\[
V = \frac{sV - sX}{sV} = \frac{s(V - X)}{sV} = \frac{s}{s} \frac{V - X}{V} \quad (7.1)
\]
where $X$ is $\min(R, G, B)$.

Next, the equation for the Hue component in eq. 3.8 can also be shown to be invariant for scaling in the RGB colour space:

$$
\frac{V - Y}{V - X} \iff \frac{c(V - Y)}{c(V - X)} \iff \frac{cV - cY}{cV - cX}
$$

(7.2)

where $V$ is the maximum value, $X$ is the minimum value, and $Y$ is the median.

The Value in the HSV colour space is given as $\max(R, G, B)$, and a scaling of each RGB channel will result in a scaling of this value.

### 7.1.2 Histogram equalisation in the HSV colour space

This method for Value manipulation can be used in colour image histogram equalisation. This can be done in the following way:

1. Make a grey scale image of the maximum channel values in each pixel. This is the same as creating a Value image.
2. Histogram equalise the Value (grey scale) image.
3. For each pixel, calculate the ratio between the original Value, and the histogram equalised Value. This ratio is used as the scaling factor for the RGB image, and is applied to each RGB channel.

By using the Value in the HSV colour space, the scaling of the RGB values will never overflow. This is because the Value channel is composed of the maximum of the RGB channels. Since the V value can only be scaled up to 255 (or the maximum channel value,) and this is the maximum value, all other values are less than or equal the V value.

If the average channel value has been used, as e.g. the Intensity channel in the IHS colour space where the Intensity, $I$ is given by $I = \frac{R + G + B}{3}$, the scaling often result in a higher value than the maximum value. This happens when there is e.g. the R value is 255, and $G = B = 0$. The average value is then 85, and if this is scaled with a factor $> 1$, the R value will become higher than 255.

It should also be noted that some Luminance-chroma colour representations does change the chroma components when scaling the RGB values. One example of this is the IHS colour space. The Hue component would be the same, but the Saturation component would change when the RGB channels are scaled. This can be shown mathematically by using the equations from section 3.2, especially equations 3.1 to 3.3. The scaled values would be:

$$
I_s = \frac{cR + cG + cB}{3} = c \frac{R + G + B}{3} = cI
$$

(7.3)
\[ V_{1,s} = \frac{2cB - cR - cG}{\sqrt{6}} = cV_1 \] (7.4)

\[ V_{2,s} = \frac{cR - cG}{\sqrt{6}} = cV_2 \] (7.5)

\[ H_s = \arctan \left( \frac{cV_2}{cV_1} \right) = H \] (7.6)

\[ S_s = \sqrt{(cV_1)^2 + (cV_2)^2} \] (7.7)

\[ = \sqrt{c^2((V_1)^2 + (V_2)^2)} \] (7.8)

\[ = c\sqrt{(V_1)^2 + (V_2)^2} = cS \] (7.9)

where the subscript ‘s’ denotes the scaled value, and c is the scaling factor.

The Hue component is the same, but the saturation is changed.

As a side note, the Saturation in the IHS colour space is the euclidean distance between \( B - R \) and \( B - G \). This can be shown by expanding eq. 3.3:

\[ S = \sqrt{\left( \frac{2B - R - G}{\sqrt{6}} \right)^2 + \left( \frac{R - G}{\sqrt{6}} \right)^2} \] (7.10)

\[ = \sqrt{\frac{4B^2 + 2R^2 + 2G^2 - 4BR - 4BG}{6}} \] (7.11)

\[ = \sqrt{\frac{2(B - R)^2}{6} + \frac{2(B - G)^2}{6}} \] (7.12)

\[ = \frac{\sqrt{2}}{6} \sqrt{(B - R)^2 + (B - G)^2} \] (7.13)

### 7.2 A short note on the HSV colour mapping

Section 3.2 describe the HSV functions, but as a reference, the Saturation is given by:

\[ S = \frac{V - X}{V} \] (7.14)

where \( V = \max(R, G, B) \) and \( X = \min(R, G, B) \).

For the special case \( R = G = B \), the hue is undefined. When \( V = 0 \), the expression for \( S \) can be found by using L’Hôpital’s rule. This rule states that if we have two expressions \( \lim_{x \to a} f(x) = \lim_{x \to a} g(x) = 0 \), then the proportion between these functions are the same as the proportions between the derivatives:

\[ \lim_{x \to a} \frac{f(x)}{g(x)} = \lim_{x \to a} \frac{f'(x)}{g'(x)} \] (7.15)

When \( V = 0 \) then, by definition, \( X = 0 \). And by using L’Hôpital’s rule, we get \( S = \frac{1 - 0}{1} = 1 \). This special case is not mentioned in the original paper.

53
by Smith[Sm78], and some implementations if RGB to HSV mapping set $S = 0$ if $V = 0$. 
Chapter 8

Experiments

8.1 Description of the experiments

The experiments performed can be divided into three sets; the experiments with the different colour classification schemes with and without GA tuning of parameters, experiments on colour segmentation using Cartesian Genetic Programming and the experiment with optimisation.

For the colour classification experiments, two classes are used. These are RED for the red colours in the sign and NOT RED for all other colours.

The first set of experiments is done in order to compare different known methods for colour segmentation, and to get an overview of the performance of the different methods. The experiments with GA tuning of the parameters are done in order to see how the GA performs in choosing these parameters, and to see if the parameters found are better than the hand-tuned parameters. The methods tested are described in section 5 and 6.1.

The next set of experiments is focused on the Cartesian Genetic Programming. This approach will be tested with different function sets, and different EAs in order to see how this method performs with regard to existing methods. The experiments will also investigate how different preprocessing methods will affect the evolved solutions. The CGP architecture is described in section 2.4, Sekaninas setup of the CGP is described in section 3.7.1 and the setup developed for this work in section 6.2.

The last set of experiments will focus on the optimisation described in chapter 7.

Table 8.1 is an overview of the experiments in this chapter.

8.1.1 Framework and tools

The programs and tools used for these experiments are described in detail in appendix B.
<table>
<thead>
<tr>
<th>Method</th>
<th>Synopsis</th>
<th>EA</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA tuning</td>
<td>Tuning of parameters for existing methods using evolution.</td>
<td>SGA</td>
<td>Sec. 8.4</td>
</tr>
<tr>
<td>CGP classifier</td>
<td>Using CGP for evolution of classifier.</td>
<td>ES</td>
<td>Sec. 8.5</td>
</tr>
<tr>
<td>Luminance</td>
<td>Testing the CGP evolved method with luminance adjustment methods.</td>
<td>–</td>
<td>Sec. 8.6</td>
</tr>
<tr>
<td>adjustment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimisation</td>
<td>Optimisation of histogram equalisation of V component in HSV colour space.</td>
<td></td>
<td>Sec. 8.7</td>
</tr>
</tbody>
</table>

Table 8.1: Overview of the experiments in chapter 8. SGA is the Simple Genetic Algorithm, ES is Evolution Strategy.

8.2 Dataset

In these experiments several input images are used, both with and without signs. These images have been chosen semi-random from a larger dataset, and were chosen in order to see different colours on the signs. The images are acquired when it is day and in either sunny or cloudy weather. The images are stored in PNG format as RGB images with 24 bits/pixel (8 bits/colour).\(^1\) The original images were in JPEG format, and have undergone a lossy compression. See section 3.3 for a description of the noise introduced in JPEG images. On the other hand, a method for retrieving traffic sign information in real-time should be immune to noise, and an image with these artifacts can be seen as a noisy image.

The training set consists of 3 images of 300x300 pixels, that contains 9 segments each. Of these 27 segments, 23 contain a speed limit sign, and the other 4 contain road scenes without signs. The reason for using several signs in each image, is that the processing of the images does take some time and by using smaller and less images the evolution could run faster. Figure 8.1 shows the images that constitute the training set.

The test set contains 48 image, 47 images with speed limit signs and one without any signs. Because the main objective was to see how the colour segmentation performed, there was no need to use many images without signs. All images in the test set are 640x512 pixels, and all speed limit signs in the images are at least 30 pixels in diameter. Apart from this, the size of the traffic signs are varying. The images in the test set are from actual road scenes. Figure 8.2 shows some of the images in the test set.

\(^1\)See http://www.libpng.org/pub/png/ for more information about the Portable Network Graphics format.
Figure 8.1: The images used in the training set
Figure 8.2: A selection of the images used in the test set.
The training set and test set has corresponding sets of images where the actual RED and NOT RED pixels are marked. These images are grey scale PNG images, and are stored in a separate folder. The actual RED pixels are white and the actual NOT RED pixels are black. Figure 8.3 shows an image and the corresponding image with the actual classes. These images have been manually created and the actual class has been marked by hand for all pixels.

8.3 Evaluation methods

The evolution is done on the training set mentioned above, and the performance of the methods are recorded using the test set consisting of 48 images. The scores and numbers in the confusion matrices is therefore not from the same images that has been used for the evolution, but from an independent test set.

Each of the experiments using evolution has been evolved 5 times, and the best result from these 5 evolutions is used. This is done in order to ensure that a bad initial population will not affect the result.

In order to calculate how good a particular segmentation method is, each segmented image is compared to a reference image describing the red areas in the sign. The pixels are grouped in four sections, namely; pixels correctly classified as RED (True Positive, P), pixels correctly classified as NOT RED (True Negative, N), pixels incorrectly classified as RED (False Positive, FP) and pixels incorrectly classified as NOT RED (False Negative, FN.) These numbers are accumulated for all images in the test set, and reported in a confusion matrix.

Figure 8.3: An image of a road scene with the corresponding image with the actual classes.
The two following ratios would then give an indication for how good the segmentation is:

True Positive rate/ positive score \((R_P)\): The ratio of correctly classified RED pixels. This is given by \(R_P = \frac{P}{P+FN}\)

True Negative rate/ negative score \((R_N)\): The ratio of correctly classified NOT RED pixels. This is given by \(R_N = \frac{N}{N+FP}\)

The results from the segmentation are given in both tabular form, and as clustered bar charts. The best solutions would be the methods with both a high positive score and a high negative score. Since the nature of the segmentation task is to interpret red colours in order to later recognise the sign, it is also important that the worst image is quite good. This has not been taken into account in the evolution, but is mentioned in the discussion of the results.

The only areas that give a positive score, are the red areas in the signs. The best images should therefore have a positive score as near 1 as possible and a negative score between 0.7 to 1.0. This is because there are other red colours in the image apart from the sign, and these would count as a false positive in the evaluation. The draw-back is that in the images, there are other areas that have similar red colours and these are counted as errors. The advantage is that this is a simple and understandable approach. The segmentation should find most of the red colours in the sign and suppress all other colours.

As mentioned in section 3.5.1, there is always a trade off between the number of false positives and false negatives for a classification method. In the experiments using evolution, the \(g\)-mean value is used as fitness function. This means that equal weight is given to the ratio of false positives and false negatives.

The running time has not been taken into account in these tests, because the algorithms and data structures have not been optimised, and these optimisations are dependent on the type of hardware they are going to be implemented in.

### 8.4 Experiment: Evolution of parameters

#### 8.4.1 Objective

The objectives for this experiment, are to measure the different colour segmentation methods reported in the literature and to see how effective tuning of parameters using GA is.

The methods tested in this experiment are listed in table 8.2, and described in section 5.3.
<table>
<thead>
<tr>
<th>ID</th>
<th>Researcher(s)</th>
<th>Method</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1+2</td>
<td>Sekanina, Bakke</td>
<td>Tristimulus difference in the RGB colour space</td>
<td>[ST02, Bak04]</td>
</tr>
<tr>
<td>B1+2</td>
<td>Paclik, Piccioli, Hsu</td>
<td>HSV segmentation with global threshold values</td>
<td>[PNPS00, HH01, PMPC96]</td>
</tr>
<tr>
<td>C1</td>
<td>Vitabile</td>
<td>HSV segmentation with 6 sets of global threshold values</td>
<td>[VS98]</td>
</tr>
<tr>
<td>D1</td>
<td>de la Escalera</td>
<td>Segmentation based on RGB ratios</td>
<td>[dlEMSA97]</td>
</tr>
<tr>
<td>E1</td>
<td>de la Escalera</td>
<td>HSV segmentation based on LUT</td>
<td>[dlEAM03]</td>
</tr>
<tr>
<td>F1</td>
<td>An</td>
<td>Dominant colour transform in RGB</td>
<td>[AC98]</td>
</tr>
<tr>
<td>G1</td>
<td>Kim</td>
<td>RG - BY colour opponent filtering</td>
<td>[Kim98]</td>
</tr>
</tbody>
</table>

Table 8.2: The reference methods for colour segmentation. The ID in the first column is used as reference in the text to the different methods.

8.4.2 Parameters

The same GA setup was used for all methods. For the methods that did mention the parameters used in the original papers, these parameters have also been tested. Table 8.3 shows the GA setup, and table 8.4 shows the parameters that were supplied in the literature. The parameters describing the GA was chosen based on some preliminary tests of different parameter sets. It should be noted that it was not any huge differences in the performances of the different GA parameter sets that were tested in the preliminary experiments, and therefore the GA parameters were held constant during these experiments.

All GA tuning experiments were performed 5 times, and the best of these 5 evolutions were reported.

8.4.3 Results and observations

The GA tuning experiments were labelled “1,” and the experiments with the original parameters were labelled “2.” Table 8.5 shows the evolved parameters for the different methods.

The confusion matrices for the different parameter sets are shown in tables 8.6 to 8.14 together with the true positive and true negative rates. Figure 8.4 shows a comparison of the parameter sets together with the true positive and negative rates for the worst individual image.
<table>
<thead>
<tr>
<th>Population size</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of generations</td>
<td>1000</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.7</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.05</td>
</tr>
<tr>
<td>Elitism</td>
<td>Yes</td>
</tr>
<tr>
<td>Selection scheme</td>
<td>Roulette wheel</td>
</tr>
</tbody>
</table>

Table 8.3: Setup of the genetic algorithm.

<table>
<thead>
<tr>
<th>A2</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta RG_{\text{max}}$</td>
<td>17</td>
</tr>
<tr>
<td>$\Delta RB_{\text{max}}$</td>
<td>17</td>
</tr>
<tr>
<td>$\Delta GB_{\text{min}}$</td>
<td>50</td>
</tr>
<tr>
<td>$R_{\text{min}}$</td>
<td>77</td>
</tr>
<tr>
<td>$T_{\text{bright}}$</td>
<td>150</td>
</tr>
<tr>
<td>$T_{\text{dark}}$</td>
<td>125</td>
</tr>
<tr>
<td>Bright adjustment</td>
<td>143</td>
</tr>
<tr>
<td>Dark adjustment</td>
<td>131</td>
</tr>
<tr>
<td>$R_{\min}$</td>
<td>77</td>
</tr>
<tr>
<td>$T_{\text{bright}}$</td>
<td>150</td>
</tr>
<tr>
<td>$T_{\text{dark}}$</td>
<td>125</td>
</tr>
<tr>
<td>Bright adjustment</td>
<td>143</td>
</tr>
<tr>
<td>Dark adjustment</td>
<td>131</td>
</tr>
</tbody>
</table>

Table 8.4: These two parameter sets were the only ones reported in the original papers, and were tested for comparison.

Figure 8.4: A comparison of the results for the different methods and parameter sets.
A1
- \(\Delta RG_{\text{max}}\): 14
- \(\Delta RB_{\text{max}}\): 4
- \(\Delta GB_{\text{min}}\): 40
- \(R_{\text{min}}\): 12
- \(T_{\text{bright}}\): 190
- \(T_{\text{dark}}\): 61
- Bright adjustment: 113
- Dark adjustment: 164

B1
- Hue: 225 – 14
- Saturation: 25 – 255
- Value: 10 – 248

C1

<table>
<thead>
<tr>
<th>Value</th>
<th>0 - 42</th>
<th>43 - 85</th>
<th>86 - 127</th>
<th>128 - 170</th>
<th>171 - 212</th>
<th>212 - 255</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hue</td>
<td>45 - 29</td>
<td>208 - 13</td>
<td>64 - 13</td>
<td>194 - 170</td>
<td>196 - 120</td>
<td></td>
</tr>
<tr>
<td>Saturation</td>
<td>140 - 255</td>
<td>75 - 255</td>
<td>182 - 109</td>
<td>35 - 223</td>
<td>33 - 245</td>
<td>11 - 135</td>
</tr>
</tbody>
</table>

D1

<table>
<thead>
<tr>
<th>(R_{\text{min}})</th>
<th>17</th>
<th>(h_{\text{min}})</th>
<th>39</th>
<th>(T_0)</th>
<th>70</th>
<th>(RG)</th>
<th>133 - 249</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_{\text{max}})</td>
<td>228</td>
<td>(h_{\text{max}})</td>
<td>198</td>
<td></td>
<td></td>
<td>(BY)</td>
<td>60 - 132</td>
</tr>
<tr>
<td>(G_{\text{min}})</td>
<td>(-2.32405 \cdot 10^{-7})</td>
<td>(s_{\text{min}})</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(G_{\text{max}})</td>
<td>(3.87969 \cdot 10^{-2})</td>
<td>Threshold</td>
<td>156</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B_{\text{min}})</td>
<td>(-6.07399 \cdot 10^{10})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B_{\text{max}})</td>
<td>0.232612</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E1

| \(T_{\text{dark}}\) | 61    |                      |       |        |       |        |           |

F1

| \(T_{\text{bright}}\) | 190   |                      |       |        |       |        |           |

G1

| Threshold | 156   |                      |       |        |       |        |           |

Table 8.5: The results from GA tuning of parameters.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Actual</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT RED</td>
<td>15073432</td>
<td>5605</td>
<td>True Negative</td>
<td>0.960975</td>
</tr>
<tr>
<td>RED</td>
<td>342627</td>
<td>37476</td>
<td>True Positive</td>
<td>0.869896</td>
</tr>
</tbody>
</table>

Table 8.6: Confusion matrix for A1

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Actual</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT RED</td>
<td>15342932</td>
<td>12544</td>
<td>True Negative</td>
<td>0.978157</td>
</tr>
<tr>
<td>RED</td>
<td>342627</td>
<td>30537</td>
<td>True Positive</td>
<td>0.708828</td>
</tr>
</tbody>
</table>

Table 8.7: Confusion matrix for A2
### Table 8.8: Confusion matrix for B1

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>15127672</td>
<td>557887</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>4902</td>
<td>38179</td>
</tr>
</tbody>
</table>

### Table 8.9: Confusion matrix for B2

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>15179552</td>
<td>506007</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>8012</td>
<td>35069</td>
</tr>
</tbody>
</table>

### Table 8.10: Confusion matrix for C1

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>14753369</td>
<td>932190</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>19808</td>
<td>23273</td>
</tr>
</tbody>
</table>

### Table 8.11: Confusion matrix for D1

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>10115535</td>
<td>5570024</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>2920</td>
<td>40161</td>
</tr>
</tbody>
</table>

### Table 8.12: Confusion matrix for E1

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>15065870</td>
<td>619689</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>4473</td>
<td>38608</td>
</tr>
</tbody>
</table>
It is interesting to note that for the two methods that had the parameters reported, the GA was able to find parameters with a better score. Method A2 had a positive average on 0.71, while the GA tuning gave a positive average on 0.87. The negative average scores were approximately equal. For B2 the average positive score were 0.81, while the GA tuning gave a positive score of 0.89. The negative average scores were also here approximately equal.

An interesting observation is that the RGB based methods had most problem with the sign in the image shown in fig. 8.5(a), while the HSV based methods had problems with the sign in the image shown in fig. 8.5(b). For the sign the RGB based methods had problems with, the signs were very dark, and the dominant colour is in fact blue. For the sign the HSV based methods had problems with, the sign had a Hue value \( > 30^\circ \), and a medium to low saturation and value. This is a light brown to orange colour.

The GA tuning of the parameters of method A1 gave a varying result in the different runs when tested on the test set. The ranges in the average scores varied from 0.071446 for the worst run to 0.861183 for the best. The method has some dependencies between the parameters, especially the light/ dark adjustment parameters and the light/ dark threshold parameters. This might create a ragged solution space with many local optima well below the best solutions.

There was a huge difference in the scores and parameters for method C. This is because the different Value ranges have independent Hue/Saturation boundaries. This, combined with the relatively limited training set, seems to indicate that it is not sufficient samples of the colours in each value range to ensure that the evolution found good solutions.
Method E gave varying parameters resulting in the same scores on the test set. This is because the two lookup tables are multiplied, and the product is thresholded. The $s_{\min}$ was around 22 for all runs, and high $h_{\min}$ values did correspond with low $h_{\max}$-values and high threshold values. In three of the runs, the $h_{\min}$-value was higher than the $h_{\max}$-value.

The best methods and parameter sets are A1, B1 and E1. They had all a quite good overall score. The different methods did all have their problematic images, and it is impossible to say that one of these is definitely better than the other two.

8.5 Experiment: Using CGP for colour segmentation

8.5.1 Objective

The first objective in this experiment is to establish how well the Cartesian Genetic programming architecture performs on colour segmentation tasks. The CGP is tested with different architecture sizes and different levels-back parameter in order to decide the best setup. The levels back parameter is adjusted to decide how much the performance is degraded by restricting the interconnects. At last, two different function sets are tested; the first is the function set used by Sekanina[Sek04b], and the other consist of arithmetic functions, and is described in table 6.1.
Parameter | Setting
--- | ---
Strategy | $(\mu + \lambda)$
$\mu$ | 4
$\lambda$ | 12
Mutation rate | 0.005
# of generations | 10000

Table 8.15: ES parameters

<table>
<thead>
<tr>
<th>Experiment</th>
<th>$n_n$</th>
<th>$n_c$</th>
<th>$l$</th>
<th>$n_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESA</td>
<td>16</td>
<td>10</td>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td>ESB</td>
<td>4</td>
<td>40</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td>ESC</td>
<td>16</td>
<td>10</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>ESD</td>
<td>4</td>
<td>40</td>
<td>40</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 8.16: The parameters to the CGP used with ES.

8.5.2 Parameters

Table 8.15 shows the parameters used for the Evolution Strategy in this experiment. The mutation rate is chosen such that on average two genes are mutated in each individual. Preliminary tests indicated that this mutation rate gave better results on average than higher mutation rates, but the difference was not very huge. A mutation rate of 0.005 means that on average 2.4 mutations are performed in each genome.

The CGP was set up as described in table 8.16. The number of nodes were held constant at 160 nodes. For all runs, the number of inputs was 16, and the number of outputs was 1.

8.5.3 Result and observations

Tables 8.17 to 8.21 shows the confusion matrices together with the true positive and true negative rate for the experiments. Figure 8.6 is a graphical representation of the ratios, together with the positive and negative rates for the worst images.

An analysis of the result in ESA, showed that only 17 nodes in the CGP architecture was actually used. This is 10.6% of the available nodes.

ESB1 did find a quite good solution, but there were many false positive red pixels. The most interesting part of the ESB solution, was that it used only 4 nodes in the CGP. All inputs was the pixel below the centre pixel.
<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>15159678</td>
<td>525881</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>3711</td>
<td>39370</td>
</tr>
</tbody>
</table>

Table 8.17: Confusion matrix for ESA.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>13734441</td>
<td>1951118</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>5657</td>
<td>37424</td>
</tr>
</tbody>
</table>

Table 8.18: Confusion matrix for ESB1.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>13742240</td>
<td>1943319</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>6315</td>
<td>36766</td>
</tr>
</tbody>
</table>

Table 8.19: Confusion matrix for ESB2.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>14792574</td>
<td>892985</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>3523</td>
<td>39558</td>
</tr>
</tbody>
</table>

Table 8.20: Confusion matrix for ESC.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED</td>
<td>15259524</td>
<td>426035</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td>4421</td>
<td>38660</td>
</tr>
</tbody>
</table>

Table 8.21: Confusion matrix for ESD.
The method for red segmentation found in ESB can be expressed as:

\[ \frac{((-R \land B) \lor G) + \frac{R+B}{2}}{2} \geq 128 \]  

(8.1)

where “\( \land \)” and “\( \lor \)” are the bitwise “and” and “or” operator, respectively. “\( \neg \)” is the bitwise “not” operator.

Even if this method gave the best score, it was not the optimal solution found. Figure 8.7 shows the image with lowest score. Even when almost 35\% of the red pixels were classified right, there sign in the image is unrecognisable. This happens because the segmentation finds many false positives around the sign.

A better (visual) solution was found in another run. This run has been named ESB2. The scores are slightly lower, but there was less noise around the signs. The worst image is shown in figure 8.8. The circular shapes of the signs can still be seen in the segmented image.

For ESB2, 25 different nodes were used, and these nodes used 18 different functions. The inputs used where two red pixels (left and right of centre pixel,) all green pixels and the blue centre pixel.

The best solution in ESC used 21 nodes, and 11 inputs. The inputs were the same 4 pixel locations from the Red and Green channel, and 3 from the Blue channel. The blue inputs were the same pixel locations as R and G, missing one. This was also the best solution in the experiment. It had
Figure 8.7: The result from segmentation using the method found in ESB1. Even when almost 35% of the pixels in the sign are classified correct, the sign is unrecognisable.

Figure 8.8: The result from segmentation using the method found in ESB2.
the highest positive score with the lowest standard deviation, and the best worst case image. It did also have a decent negative score.

The best solution in ESD used 10 nodes and 6 inputs. The inputs were 2 pixels from each channel. The same pixels were used from the R and G channel ($(x - 1, y)$ and $(x, y - 1)$) and the inputs from the blue channel were $(x, y - 1)$ and $(x, y + 1)$.

The image with the best result from these experiments is shown in figure 8.9. The positive score for this image is 0.998024, and the negative score is 0.991597. A more typical image is shown in figure 8.10. For this image the positive score is 0.917603 and the negative score is 0.925107.

When looking at the resulting segmented images, there is a distinction between the two function sets used. The function sets in ESA and ESB did
find many edges in the images, and did also mark many single pixels in
the image. The blocking effect in the JPEG images was also more noticeable
with this function set. The arithmetic (and saturated) functions used in ESC
and ESD, did usually result in more continuous RED regions. Some edges
was found by this function set, and some square blocks were marked, but
not nearly as much as with the function set used in ESA and ESB. Figure
8.11 is an example on these differences.

There were some differences between the different CGP setups, but the
average performance of the different runs of the different setups were quite
near each other. The only exception to this is the setup in ESB. The negative
scores were in all runs about 7-8 % lower than in the other runs.

It is also interesting to note that the methods with a higher levels back
parameter (ESB and ESD,) did tend to create smaller solutions. ESA and
ESC used on average 24 nodes each run, and ESB and ESD used on average
14 nodes each run. This is because the methods with a small levels back
parameter are forced to use more nodes. The methods with a small levels
back parameter did also tend to create better solutions on average.

A detailed listing of the evolved CGP configurations can be found in
appendix C.

8.5.4 Comparison with the tested, existing methods

The CGP architecture evolved some good results. Figure 8.12 shows the
three best results from the experiments with GA tuning and CGP evolution.
The differences are not very huge except that the most difficult images had
a better score with the CGP architecture.
Figure 8.12: A comparison of the three best methods from the GA tuning experiment and the CGP evolution experiment.

The methods evolved with the CGP architecture had, with the exception of ESB, a better positive score. The negative scores were on the same level. The worst case images on the other hand, shows that the CGP outclassed the existing methods on the positive scores. But the existing methods out-classed the CGP architecture on the negative worst case images.

One of the reasons the CGP architecture was a bit better on average than the existing methods, is that the existing methods uses static threshold values. With the exception of ESB, the CGP architecture evolved methods that did not include any constants at all. This would indicate that the methods evolved are at least somewhat adaptive.

8.6 Experiment: Luminance adjustments

8.6.1 Objective

The objective of this experiment is to see how effective the luminance adjustments are. A better score would indicate that the luminance adjustments are good. A score about the same as the not preprocessed methods would indicate a robust method, while a lower score would indicate that the preprocessing makes it either very difficult for the methods or that the methods are very fragile with respect to changing luminance conditions.

The methods are described in section 3.4.2.
Table 8.22: Setup of the experiment with histogram adjustments.

<table>
<thead>
<tr>
<th>ID</th>
<th>Configuration</th>
<th>Adjustment method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>ESA</td>
<td>Cubic root</td>
</tr>
<tr>
<td>AH</td>
<td>ESA</td>
<td>Histogram equalisation</td>
</tr>
<tr>
<td>AA</td>
<td>ESA</td>
<td>Adaptive histogram eq.</td>
</tr>
<tr>
<td>CC</td>
<td>ESC</td>
<td>Cubic root</td>
</tr>
<tr>
<td>CH</td>
<td>ESC</td>
<td>Histogram equalisation</td>
</tr>
<tr>
<td>CA</td>
<td>ESC</td>
<td>Adaptive histogram eq.</td>
</tr>
</tbody>
</table>

Table 8.23: The parameters to the adaptive histogram equalisation algorithm.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Window size</td>
<td>65</td>
</tr>
<tr>
<td>Horizontal grid spacing</td>
<td>64</td>
</tr>
<tr>
<td>Vertical grid spacing</td>
<td>64</td>
</tr>
</tbody>
</table>

8.6.2 Parameters

The results from the previous experiments will be compared. Table 8.22 is the setup of the experiment together with the CGP configurations that are tested.

The parameters for the adaptive histogram equalisation algorithm is described in table 8.23.

8.6.3 Result and observations

Tables 8.24 to 8.29 shows the confusion matrices for the experiments. Figures 8.13 and 8.14 is a graphical representation of the same data, also including the images with the worst score.

Two of the methods evolved in the previous experiment were tested here. The two were the methods with the best scores.

For the most part, the luminance adjustments done here in HSV have a negative impact on the scores.

Table 8.24: The result from AC.
<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED</th>
<th>RED</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED 15195003</td>
<td>RED 490556</td>
<td>True Negative 0.968726</td>
</tr>
<tr>
<td></td>
<td>RED 5191</td>
<td>NOT RED 37890</td>
<td>True Positive 0.879506</td>
</tr>
</tbody>
</table>

*Table 8.25: The result from AH.*

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED 15018515</th>
<th>RED 667044</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED 5329</td>
<td>RED 37752</td>
<td>True Negative 0.957474</td>
</tr>
<tr>
<td></td>
<td>RED 5329</td>
<td>NOT RED 37752</td>
<td>True Positive 0.876303</td>
</tr>
</tbody>
</table>

*Table 8.26: The result from AA.*

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED 14153539</th>
<th>RED 1532020</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED 5174</td>
<td>RED 37907</td>
<td>True Negative 0.902329</td>
</tr>
<tr>
<td></td>
<td>RED 5174</td>
<td>NOT RED 37907</td>
<td>True Positive 0.879901</td>
</tr>
</tbody>
</table>

*Table 8.27: The result from CC.*

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED 14782912</th>
<th>RED 902647</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED 4858</td>
<td>RED 38223</td>
<td>True Negative 0.942454</td>
</tr>
<tr>
<td></td>
<td>RED 4858</td>
<td>NOT RED 38223</td>
<td>True Positive 0.887236</td>
</tr>
</tbody>
</table>

*Table 8.28: The result from CH.*

<table>
<thead>
<tr>
<th>Predicted</th>
<th>NOT RED 14056990</th>
<th>RED 1628569</th>
<th>Rate / score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>NOT RED 5949</td>
<td>RED 37132</td>
<td>True Negative 0.896174</td>
</tr>
<tr>
<td></td>
<td>RED 5949</td>
<td>NOT RED 37132</td>
<td>True Positive 0.861911</td>
</tr>
</tbody>
</table>

*Table 8.29: The result from CA.*
Figure 8.13: Results for ESA with luminance adjustments. The dataset “AO” is the score without adjustment of the HSV Value channel.

Figure 8.14: Results for ESC with luminance adjustments. The dataset “CO” is the score without adjustment of the HSV Value channel.
The cubic root scaling had a positive impact on the negative scores, in ESA, but not in ESC, where the score was a bit lower than the non-preprocessed method. Since the average positive score decreased in all runs, this means that fewer pixels were classified as RED, and this applies to both RED and NOT RED pixels.

The histogram equalisation methods did make classification difficult with some signs, and the positive worst case scored much lower than the non-preprocessed. This was also the preprocessing that had the least impact on the average scores, both positive and negative.

When looking at the worst images, the image labelled “aadz” was reported as the worst case in 4 of the methods (ESA, AH, ESC, CH.) All static histogram equalisation methods made the classification more difficult with this image. The image labelled “aabq” was reported as the worst image in 1 case (AC.) The image “aadb” was reported worst in 2 cases (AA and CA,) all adaptive histogram equalisation. Method CC reported image “aaag” as the worst. Figure 8.15 shows two of the images that were problematic to classify correctly with the histogram equalisation preprocessing.

The figure clearly shows the problem with the histogram equalisation methods. In the image labelled aadz, there is many bright colours, e.g. the sky and a lot of colours with a medium luminance. The sign that are dark, is therefore much darker when histogram equalised.

The sign in the image labelled “aabd” does in fact consist of a wide range of luminance values, and the white area inside the sign does not have a HSV Value component on more than about 128. This means that the histogram for the sign is quite flat, as opposed to the bright area around which has a high peak for the Value at 220. This means that colours below 220 will be very dark, and this is unfortunately all colours in the sign.

Some of the colours in the signs is more difficult to find after applying this kind of preprocessing. The scores were a bit lower than without preprocessing, but the fact that they are not dramatically lower would indicate that the evolved methods is not fragile with regard to varying luminance.

8.7 Experiment: Optimised Value manipulation in the RGB colour space

8.7.1 Objective

The objective in this experiment is to time the optimised Value manipulation described in section 7.1 and the trivial approach, i.e. translation from RGB to HSV and back.
Figure 8.15: The images that was most problematic when histogram equalised. Notice the effect of the JPEG compression in the image pre-processed with the adaptive histogram equalisation method (d).
<table>
<thead>
<tr>
<th>Image size</th>
<th>Trivial (s)</th>
<th>Optimised (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>1.391</td>
<td>0.726</td>
</tr>
<tr>
<td>1600x1200</td>
<td>10.254</td>
<td>4.286</td>
</tr>
</tbody>
</table>

Table 8.30: The results for histogram equalisation using the trivial method, and the optimised method.

Figure 8.16: The results of histogram equalisation using a trivial method and the optimised method for different image sizes.

### 8.7.2 Parameters and setup

In order to test the Value manipulation technique, the Value channel is histogram equalised. The timer started right before the histogram equalisation function in the code, and were stopped right after. The three results were then added together. Each image was processed four times, and the last three runs were timed. This was done 14 times, and the average run time calculated. Two different image sizes were tested; a 640x480 pixels image and a 1600x1200 pixels image.

### 8.7.3 Result and Observations

Table 8.30 shows a table of the results, and figure 8.16 is the same data represented in a bar chart. It should be noted that the times reported is for three consecutive runs. The time used for one image is therefore the reported time divided by three.

The results shows that the optimised method is in fact faster than the trivial approach. It should be noted that if the image are going to be processed in the HSV colour space anyway, it would be better to do the histogram equalisation after the RGB $\rightarrow$ HSV translation.
Chapter 9

Conclusion

9.1 Conclusion

This thesis has looked on some of the existing colour classification methods used in traffic sign detection systems, and used evolutionary methods for colour classification.

During the preliminary tests of different traffic sign detection algorithms, the need for good classifiers were recognised. Many existing methods focus on the higher level recognition and interpretation algorithms, and the colour classification they are based upon, are often only mentioned as a short note.

Many of the colour classification methods were not very good when tested against the test set used in this thesis. In order to test the classifiers in an objective setting, the parameters used for the methods were evolved using a SGA. The GA was able to find better parameter sets than the reported parameter sets.

The Cartesian Genetic Programming architecture was used for evolving a colour classification method. The CGP architecture were able to find better colour classifiers than the existing methods with evolved parameter sets.

The preprocessing of the Value channel in the HSV colour space showed that none of the preprocessing methods made it easier to classify the colours in all images. If the most difficult images are kept out of the equation, the evolved CGP solutions were quite robust against changes in the Value channel. The experiments showed that it is difficult, if not impossible to apply one Value manipulation technique that works for all images that are captured under different environments.

This work also showed that scaling the individual RGB components with the same factor is mathematically the same as scaling the Value component in the HSV colour space. An application of this is a faster method for histogram equalisation than the trivial approach that first converts the
RGB image to HSV, histogram equalises the Value channel and converts the image back to RGB. The timing analysis showed that the histogram equalisation directly on the RGB components was 50% faster than converting the image to HSV and back.

9.2 Future work

There are still much that can be explored with the methods used here. The CGP architecture has been used to evolve colour classifiers, and this architecture can be explored further in image processing tasks in general and traffic sign detection in particular.

There are some limitations with the colour classification. One of these limitations is that the colours in sign often are very dark, or have a low saturation. This is a problem because the colour classification is very inexact under these circumstances. The solution to this is to use more of the sign’s qualities for the detection. It could have been interesting to use CGP for texture classification. Other qualities that can be explored, is the relative orientation of the edges in the sign.

There are also other evolution based methods that can be explored for classification tasks. E.g. and-or gate arrays can be configured, or GP can be used. The possibilities are many, and some of these methods might be better when e.g. die area, time usage or power usage are taken into account.

Another interesting approach could have been to use the three dimensions in video streams fully. The dimensions would be the horizontal and vertical dimension and the third being the time dimension. All three dimensions are discrete, the vertical and horizontal dimension is bound by the sampling resolution, and the time is bound by the sampling speed. The horizontal and vertical dimensions are finite, and the time dimension is practically infinite.

It should also be researched how such a system can be integrated with other safety systems, like existing ISA and ITS systems in order to take into account several sources of information. The next generation of ISA systems should be able to suggest a proper speed based on e.g. the current speed limit, tyre friction, the curvature of the road, children playing on the side of the road, etc.
Part III

Appendices and bibliography
Appendix A

Acronyms used in the text

ABS  Antilock Breaking System
\{A(RS)^2\}  Automatic Road Signs Recognition System
CGP  Cartesian Genetic Programming
CIPM  International Committee for Weights and Measures
EA  Evolutionary Algorithm
EIPM  Extended Inverse Perspective Mapping
ES  Evolutionary Strategies
GA  Genetic Algorithm
GOLD  Generic Obstacle and Lane Detection
HSV  Hue, Saturation, Value colour space
IDSL  Image Detection of Speed Limits
IEC  International Electrotechnical Commission
IEEE  Institute of Electrical and Electronics Engineers
IHS  Intensity, Hue, Saturation colour space
IPM  Inverse Perspective Mapping
ISA  Intelligent Speed Adaption
ITS  Intelligent Transportation Systems
LUT  Lookup Table
NN  Neural Network
RGB Red, Green, Blue colour space
SGA Simple Genetic Algorithm
SI Système International d’unités
SNR Signal-to-Noise Ratio
Appendix B

Description and usage of computer programs used.

B.1 Programs developed for this thesis

The framework for the methods used here is five programs written in C
and C++. The programs below are written from scratch for this thesis.

**sigrec**: A program for testing different configurations and methods.

**garec**: A program for GA tuning of parameters. This program uses
GALib from MIT.

**cgprec**: A program for evolving a colour segmentation algorithm using
the CGP framework. This program uses GALib and a ES library writ-
ten for this thesis.

**teststuff**: A program for converting between different colour spaces,
preprocessing etc. This is used to verify that the algorithms are cor-
rect and for timing analysis.

**analyse_cgp**: A program for analysing the CGP architecture based on a
given configuration.

B.1.1 sigrec

Name

**sigrec** – Sign recognition.

Synopsis

```
sigrec [-s] [-c file] image(s)
```

Description

sigrec is a program that analyses different colour classification methods and parameters. All methods used in this work is implemented, and the selected method and its parameters are described in a configuration file. The program were originally written for testing traffic sign detection. But as the work progressed the program was used for testing colour classification methods instead. Therefore the program might seem a bit strange with some unused blocks etc.

The program works by reading the configuration file, and decides what method and parameters to use. The program then reads the correct templates from the template directory listed in the configuration file. At last the program classifies the list of files given at the command line, and for each file, compares the result with the template and log the results.

The output is the time used for each image, and a summary of the results. The last list in the output is detailed scores for each image classified.

A naming convention was made in order to facilitate detection result logging, but as another method was used for colour classification result logging, this is not actually used for anything now. It was not removed because the testset was already encoded like this. The naming convention of the input files are:

1. A four letter name
2. One digit for the number of speed limit signs.
3. Sign descriptions, one for each sign. This description consist of the following information:
   (a) 3 digits for the speed limit
   (b) 8 digits for upper left corner of bounding box (xxxx,yyyy).
   (c) 8 digits for lower right corner of bounding box (xxxx,yyyy).

Options

-s
   Test of segmentation method. (This has to be present.)

-c file
   The configuration file

Configuration file

# Example configuration file for sigrec
[general]
# Red segmentation method. Possible methods are:
# hsvcube - A cube in the hsv colour space
# diff_r - Sekanina and Bakkes method
# cgp_r - CGP
# rg_by - The rg-by method.
# domcol - Dominant colour method.
# hsv6 - HSV with 6 different SV sets.
# rgbratio - Method based on the ratio between the channels
# hslut - The LUT in HSV based method.

```python
segm_r = diff_r
```

# White segmentation method. Only one is implemented:
# diff_w - Method described byu Sekanina

```python
segm_w = diff_w
```

# The preprocessing done. Possible methods are:
# average - Neighbourhood averaging filter
# histeq - Histogram equalisation
# ahisteq - Adaptive histogram equalisation
# cubic - Cubic scaling of V component in HSV.
# croot - Cubic root scaling of V component.
# none - No preprocessing

```python
preprocess = none
```

# The neighbourhood averaging window size
```python
avgfiltersize = 3
```

# The postprocessing method. Possible methods are
# none - nothing
# grow - Region growing as described by Sekanina

```python
postprocess = none
```

# Directory where red templates are located
```python
rdir = path/to/templates
```

# Working directory, where all new files are created
```python
wdir = path/to/results
```

# Name of score file for matlab
```python
mfile = results.m
```

# The name of the log file
```python
logfile = log.txt
```

# Only the section below corresponding to segm_r above is needed.

```python
[hsvcube]
h_min = 230
h_max = 30
s_min = 10
s_max = 255
v_min = 5
v_max = 255
```
[diff_r]
rg_diff = 14
rb_diff = 4
gb_diff = 40
r_min = 12
t_min = 61
t_max = 190
dadj = 164
ladj = 113

[cgp]
cgp.n_i = 16
cgp.n_o = 1
cgp.n_n = 2
cgp.n_c = 10
cgp.n_r = 16
cgp.l = 4
cgp.n_f = 16

# The binary file with the configuration
configstring = cstringb0.dat

# The function set used
functionset = arithmetic

[rg_by]
rg_min = 133
rg_max = 249
by_min = 60
by_max = 132

[domcol]
T = 70

[hsv6]
h_min0 = 45
h_max0 = 29
s_min0 = 140
s_max0 = 255
h_min1 = 208
h_max1 = 13
s_min1 = 75
s_max1 = 255
h_min2 = 86
h_max2 = 138
s_min2 = 182
s_max2 = 109
h_min3 = 64
h_max3 = 13
s_min3 = 35
s_max3 = 223
h_min4 = 194
h_max4 = 170
s_min4 = 33
s_max4 = 245
B.1.2 garec

Name

garec – GA tuning of parameters.

Synopsis


Description

garec is a program that tunes the parameters for a given colour classification method. The program works by first reading the images and the corresponding templates. The templates are black images with white areas for the red colours. The program then reads the configuration file, selects the right classification method. At last the Genetic Algorithm is run, using the parameter structure as the genome. The resulting parameters can by pasted directly into the configuration file for sigrec.

Options

-l file
    Log file

-q
    Quiet, suppress some information.
-s file
    File containing the scores for each generation.

-d dir
    Working directory. All created files are put here.

-r dir
    Red template directory.

-m
    Method for colour classification. This can be one of:
    • hsvcube
    • rgbdiff
    • hsv6
    • domcol
    • rgy
    • hsvlut
    • rgbratio

-o float
    Crossover rate for GA.

-u float
    Mutation rate for GA.

B.1.3 cgprec

Name

.cgprec – Cartesian Genetic Programming for colour classification.

Synopsis

eatype] image(s)

Description

cgprec will evolve a CGP colour classifier. It works mostly the same as
garec, but it can select to use a GA or ES as the evolutionary algorithm.
The configuration file consist of the CGP setup parameters.
Options

-c file
  Configuration file.

-l file
  Log file

-s file
  File containing the scores for each generation.

-d dir
  Working directory. All created files are put here.

-r dir
  Red template directory.

-o float
  Crossover rate for GA.

-u float
  Mutation rate for GA and ES.

-t file
  Configuration string for CGP.

-e eatype
  Type of EA. Possible values are:
  es: Evolution Strategy.
  ga: Genetic algorithm.

Configuration file

# Configuration file example for cgprec

# The parameters are the same as described for CGP.
n_i = 16
n_o = 1
n_n = 2
n_r = 4
n_c = 40
n_f = 16
l = 40

# The ea can be one of:
#  es - Evolution strategy
#  ga - Genetic Algorithm
ea = es
B.1.4 teststuff

Name
teststuff – Program for testing purposes.

Synopsis
teststuff [-c method] [-p method] [-a size] [-v] image(s)

Description
teststuff is a program for testing the different functions developed for the other programs. It also does the timing analysis.

Options
- `-c method`
  Convert an image. The possible conversion methods are:
  - `lll`: Convert RGB image to the lll colour space.
  - `rgb`: Convert RGB image to scaled rgb image. \( r = \frac{R}{R+G+B} \) etc.
  - `hsv`: Convert RGB image to HSV.
  - `ihs`: Convert RGB image to IHS.
  - `rgbly`: Convert RGB image to RG and BY images.
  - `domcol`: Convert RGB image to dominant colour image.

- `-p method`
  Preprocess image. Valid preprocessing methods are:
  - `mean3`: Neighbourhood average with 3x3 window.
  - `mean5`: Neighbourhood average with 5x5 window.
  - `croot`: Cubic root scaling of Value channel.
  - `hcube`: Cube scaling of Value channel.
  - `hist1`: Histogram equalisation of the RGB channels individually.
  - `hist3`: Histogram equalisation of the Value channel.
  - `histad`: Adaptive histogram equalisation of the Value channel.

- `-a size`
  Timing analysis of the trivial vs. fast neighbourhood averaging methods with the given size of the window.

- `-v`
  Timing analysis of the trivial vs. optimised histogram equalisation method.
B.1.5 analyse_cgp

Name
analyse_cgp—Analyse the nodes of a Cartesian Genetic Program.

Synopsis
analyse_cgp fset configfile

Description
analyse_cgp creates an analysis of the CGP, and prints only the nodes and functions that are actually used.

Options
fset
This is the function set used for the CGP. Valid options are:
   sek: Sekanina’s function set.
   arit: Arithmetic functions.

configfile
This is the configuration string, as written by cgprec with the option -t.

B.2 Third party tools
This is the third party tools that are used in this work.

B.2.1 GALib
GALib is a C library for Genetic Algorithms. The homepage is at http://lancet.mit.edu/ga/.

B.2.2 Condor
Condor is a system for distributed computing, using the computing power of idle computers. The programs that are to be run under Condor can be linked against the condor libraries at compile time.

In order to get galib to work with condor, the #define GALIB_USE_PID has to be commented out in gaconfig.h. The reason for this, is that get_pid() will return 0 when linked with the condor-libraries, and GARandomSeed() will be caught in an endless loop.
B.2.3 Gnumeric

Gnumeric is a spreadsheet, and was used for creating the bar plots. The reason for choosing this program, was that it had the ability to export the plots to a graphics file.

The Gnumeric homepage can be found at http://www.gnome.org/projects/gnumeric/.

B.2.4 Octave

Octave is a free Matlab clone. It has some shortcomings with regard to Matlab and image processing.

The homepage can be found at http://www.octave.org/.

B.2.5 \LaTeX

This work is typeset in \LaTeX, using palatino 12 pt font.

B.2.6 Gimp

The GNU Image Manipulation Program was used to create the images with actual classes for the classification.
Appendix C

Evolved CGP configurations

This is the detailed configurations evolved for the CGP architecture. Only the nodes that are actually used is given here. The prefix ‘N’ indicates a node (N030 is the node with id 30,) and the prefix ‘F’ indicates a function ID (F23 is the function with ID 23.)

The first block contains the following information:
1. Number of nodes actually used.
2. The next lines beginning with Nxxx, is the listing of nodes. The line contains the following information:
   Nxxx: The ID for current node.
   (x): The number of paths containing this node.
   in1, in2: The node IDs for the inputs.
   (Fxx): The Function ID.
   <some text>: A short description of the function.
3. The node ID for the output node.

The second block contains the description of the inputs used. The information is ordered as such:
xx: The input node ID.
Colour: The colour channel of the input.
x,y: The spatial location of the input pixel.

The third block contains a more detailed analysis of the functions used. This block is ordered this way:
Fxx: The function ID.
(xx): The number of nodes using this function.
<some text>: A short description of the function.
C.1 ESA

Function set: Sekaninas functions.

Number of nodes used: 17
N016: (1): in1 = N007, in2 = N009: (F21) (in1 + in2) >>
N019: (1): in1 = N009: (F05) << 2
N024: (1): in1 = N009, in2 = N010: (F25) (not in1) and in2
N025: (2): in1 = N014, in2 = N002: (F14) (in1 + in2 + 1) >>
N027: (1): in1 = N004, in2 = N008: (F20) +
N031: (1): in1 = N002: (F03) not
N033: (1): in1 = N016, in2 = N025: (F30) ((in1 + in2) >>) + 1
N039: (1): in1 = N025: (F03) not
N041: (1): in1 = N024, in2 = N027: (F33) min
N046: (1): in1 = N019: (F02) >> 4
N052: (1): in1 = N046, in2 = N012: (F30) ((in1 + in2) >>) + 1
N068: (1): in1 = N031, in2 = N012: (F30) ((in1 + in2) >>) + 1
N077: (1): in1 = N039, in2 = N033: (F30) ((in1 + in2) >>) + 1
N082: (1): in1 = N041: (F06) << 4
N100: (1): in1 = N082, in2 = N052: (F25) (not in1) and in2
N123: (1): in1 = N100, in2 = N068: (F13) max
N134: (1): in1 = N077, in2 = N123: (F27) not (in1 or in2)
Output: N134

Inputs used:
02: Red : x, y
04: Red : x+1, y
07: Green: x, y
08: Green: x, y+1
09: Green: x+1, y
10: Blue : x-1, y
12: Blue : x, y
14: Blue : x+1, y

Functions used:
F02: ( 1): >> 4
F03: ( 2): not
F05: ( 1): << 2
F06: ( 1): << 4
F13: ( 1): max
F14: ( 1): (in1 + in2 + 1) >>
F20: ( 1): +
F21: ( 1): (in1 + in2) >>
F25: ( 2): (not in1) and in2
F27: ( 1): not (in1 or in2)
F30: ( 4): ((in1 + in2) >>) + 1
F33: ( 1): min

C.2 ESB1

Function set: Sekaninas functions.

Number of nodes used: 4
N018: (1): in1 = N004, in2 = N014: (F21) (in1 + in2) >>
N019: (1): in1 = N004, in2 = N014: (F25) (not in1) and in2
N022: (1): in1 = N019, in2 = N009: (F27) not (in1 or in2)
N029: (1): in1 = N022, in2 = N018: (F21) (in1 + in2) >>
Output: N029

Inputs used:
04: Red : x+1, y
09: Green: x+1, y
14: Blue : x+1, y

Functions used:
F21: ( 2): (in1 + in2) >>
F25: ( 1): (not in1) and in2
F27: ( 1): not (in1 or in2)

C.3 ESB2

Function set: Sekaninas functions.

Number of nodes used: 25
N017: (5): in1 = N009, in2 = N005: (F23) and
N018: (2): in1 = N012, in2 = N007: (F13) max
N020: (3): in1 = N003, in2 = N017: (F11) xor not
N024: (3): in1 = N006, in2 = N001: (F25) (not in1) and in2
N027: (2): in1 = N003, in2 = N018: (F23) and
N028: (1): in1 = N024: (F03) not
N029: (1): in1 = N020, in2 = N027: (F18) (in1 and 0xcc) or (in2 and 0x33)
N030: (1): (F10) 0xff
N031: (2): in1 = N024: (F06) << 4
N038: (2): in1 = N017, in2 = N008: (F32) saturated add
N051: (2): in1 = N020, in2 = N031: (F23) and
N053: (1): in1 = N030, in2 = N027: (F32) saturated add
N054: (1): in1 = N038, in2 = N029: (F23) and
N064: (1): in1 = N053: (F01) >> 2
N065: (1): (F09) 0x33
N074: (1): in1 = N054: (F02) >> 4
N081: (1): in1 = N051: (F31) in1
N092: (1): in1 = N028, in2 = N074: (F27) not (in1 or in2)
N095: (1): in1 = N038, in2 = N051: (F27) not (in1 or in2)
N096: (1): in1 = N095: (F02) >> 4
N109: (1): in1 = N065, in2 = N081: (F22) or
N135: (1): in1 = N109: (F05) << 2
N139: (1): in1 = N064, in2 = N135: (F17) (in1 and 0xf) or (in2 and 0xf0)
N159: (1): in1 = N139, in2 = N096: (F30) ((in1 + in2) >>) + 1
N166: (1): in1 = N092, in2 = N159: (F32) saturated add
Output: N166

Inputs used:
01: Red : x, y-1
03: Red : x, y+1
05: Green: x-1, y
06: Green: x, y-1
07: Green: x, y
08: Green: x, y+1
09: Green: x+1, y
12: Blue : x, y

Functions used:
F01: (1): >> 2
F02: (2): >> 4
F03: (1): not
F05: (1): << 2
F06: (1): << 4
F09: (1): 0x33
F10: (1): 0xff
F11: (1): xor not
F13: (1): max
F17: (1): (in1 and 0xf) or (in2 and 0xf0)
F18: (1): (in1 and 0xcc) or (in2 and 0x33)
F22: (1): or
F23: (4): and
F25: (1): (not in1) and in2
F27: (2): not (in1 or in2)
F30: (1): {{in1 + in2} >>} + 1
F31: (1): in1
F32: (3): saturated add

C.4 ESC

Function set: Arithmetic functions.

Number of nodes used: 21
N020: (1): in1 = N011, in2 = N011: (F01) overflow bit add
N021: (1): in1 = N008, in2 = N001: (F03) difference
N023: (2): in1 = N006: (F13) halve
N024: (1): in1 = N011, in2 = N011: (F02) saturated add
N026: (1): in1 = N008, in2 = N009: (F07) multiplication, high byte
N027: (1): in1 = N012: (F13) halve
N030: (2): in1 = N002, in2 = N007: (F05) saturated subtract
N034: (1): in1 = N026, in2 = N020: (F15) modulo
N036: (1): in1 = N009, in2 = N003: (F00) add
N038: (1): in1 = N023, in2 = N024: (F04) subtract
N042: (1): in1 = N021, in2 = N004: (F00) add
N043: (1): in1 = N011, in2 = N027: (F07) multiplication, high byte
N045: (1): in1 = N023, in2 = N030: (F11) saturated divide
N056: (1): in1 = N045, in2 = N038: (F07) multiplication, high byte
N070: (1): in1 = N013, in2 = N042: (F08) max
N071: (1): in1 = N036, in2 = N034: (F11) saturated divide
N087: (1): in1 = N043: (F13) halve
N090: (1): in1 = N030, in2 = N070: (F07) multiplication, high byte
N091: (1): in1 = N071, in2 = N056: (F11) saturated divide
N098: (1): in1 = N091, in2 = N087: (F08) max
N142: (1): in1 = N098, in2 = N090: (F06) multiplication

Output: N142

98
Inputs used:
01: Red : x, y-1
02: Red : x, y
03: Red : x, y+1
04: Red : x+1, y
06: Green: x, y-1
07: Green: x, y
08: Green: x, y+1
09: Green: x+1, y
11: Blue : x, y-1
12: Blue : x, y
13: Blue : x, y+1

Functions used:
F00: (2): add
F01: (1): overflow bit add
F02: (1): saturated add
F03: (1): difference
F04: (1): subtract
F05: (1): saturated subtract
F06: (1): multiplication
F07: (4): multiplication, high byte
F08: (2): max
F11: (3): saturated divide
F13: (3): halve
F15: (1): modulo

C.5 ESD

Function set: Arithmetic functions.

Number of nodes used: 10
N016: (1): in1 = N006, in2 = N005: (F00) add
N017: (1): in1 = N000, in2 = N001: (F07) multiplication, high byte
N018: (1): in1 = N013, in2 = N013: (F02) saturated add
N019: (1): in1 = N000, in2 = N005: (F03) difference
N021: (1): in1 = N016, in2 = N011: (F04) subtract
N022: (1): in1 = N018, in2 = N019: (F11) saturated divide
N027: (1): in1 = N011, in2 = N017: (F05) saturated subtract
N029: (1): in1 = N022, in2 = N027: (F09) min
N101: (1): in1 = N021, in2 = N000: (F04) subtract
N132: (1): in1 = N101, in2 = N029: (F00) add
Output: N132

Inputs used:
00: Red : x-1, y
01: Red : x, y-1
05: Green: x-1, y
06: Green: x, y-1
11: Blue : x, y-1
13: Blue : x, y+1

Functions used:
F00: ( 2): add  
F02: ( 1): saturated add  
F03: ( 1): difference  
F04: ( 2): subtract  
F05: ( 1): saturated subtract  
F07: ( 1): multiplication, high byte  
F09: ( 1): min  
F11: ( 1): saturated divide
Bibliography


Symposium on Image Analysis and Interpretation, pages 95–100, 1996.


