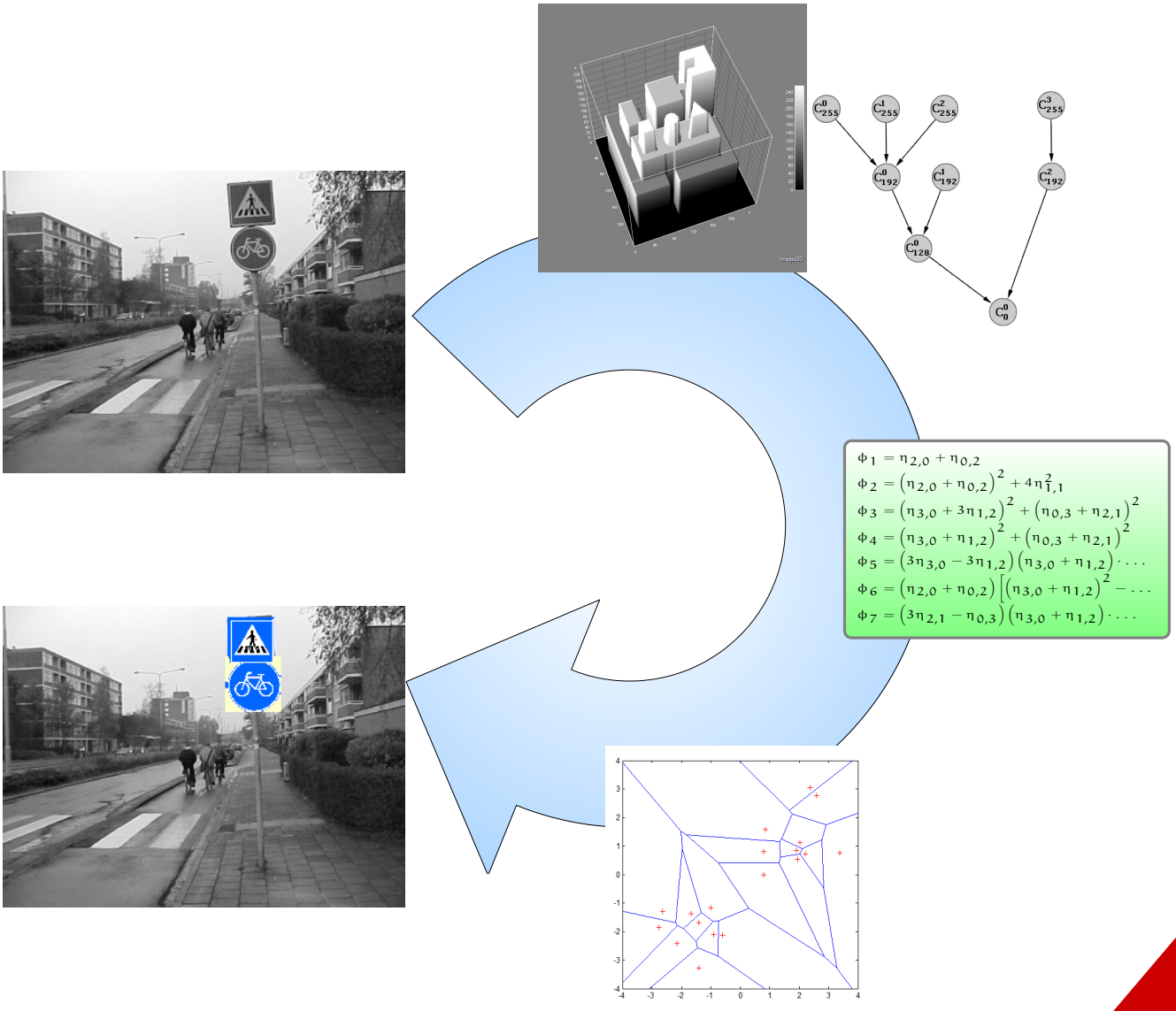
university of groningen

faculty of mathematics and natural sciences

AUTOMATIC TRAFFIC SIGNS RECOGNITION

FABIO BRACCI

Master of Science in Computing Science



University of Groningen

Faculty of Mathematics and Natural Sciences

Computing Science

June 2010

::

Fabio Bracci: *Automatic Traffic Signs Recognition,*
Master of Science in Computing Science,
Mastertrack Intelligent Systems,
© Fabio Bracci, June 2010
f.bracci@student.rug.nl
f.bracci@alumnus.rug.nl
fabio@fmf.nl

SUPERVISORS:
Michael H. F. Wilkinson
m.h.f.wilkinson@rug.nl
http://www.cs.rug.nl/~michael/
Michael Biehl
m.biehl@rug.nl
http://www.cs.rug.nl/~biehl/

SECONDARY SUPERVISORS:
Paris Avgeriou
paris@cs.rug.nl
http://www.cs.rug.nl/~paris/

LOCATION:
University of Groningen,
Groningen, The Netherlands

DEPARTMENT:
Faculty of Mathematics and Natural Sciences,
Computing Science

"The lyf so short, the craft so long to lerne."

— Geoffrey Chaucer (born 1340/44, died 1400),
The Parliament of Fowls

Dedicated to
my family, sometimes patient but not feeling with me,
sometimes feeling with me but not patient
Michel, a beloved friend who took me as a brother
Stefano, a far away friend who stayed close
and all my other close and far away friends.

ABSTRACT

Nowadays a growing number of sources is producing complex digital images; the need for object detection arises also in the automotive world. In particular driver's assistance systems aim at a continuous analysis of the road looking for obstacles, lane marks and traffic signs.

In this research the aim is to automatically detect the traffic signs along the road based on their shape, irrespective of brightness, position, size, orientation. To this end theoretical tools from Image Analysis, Mathematical Morphology, Pattern Recognition and Machine Learning are linked together to form a modular detection framework.

Images are first decomposed by looking at connected components of constant intensity and the Max-Tree is set up; the Max-Tree is interpreted as a tree of nested peak components from which the shape features are extracted by means of image moments.

The moments are collected and used to train supervised classifiers as CCM, kNN and LVQ. In order to have a useful training data set a labeling step turns out to be necessary; the labeling is performed by using a self-made application realized during a precedent stage.

The labeling stage led to the construction of a labeled data set (ground truth); during this phase the limits of a naive reference-based filtering became apparent and the usefulness of contrast-based filtering emerges very clearly.

Later also a preprocessing step turned out to be necessary; this because of the huge number of peak components deriving from each of the gray images. Preprocessing revealed difficulties in the data set reduction by considering the peak component area and the moment similarity to ideal traffic sign and results in sampling as viable solution. The use of different Minkowsky metric looks marginal as well.

Training the classifiers on the sampled data set led to other insights. First the problem seems to be learnable by kNN and LVQ but not by CCM; this confirms preprocessing difficulties and indicates the distribution of the classes are not hyperspherical and well separated. The performances of kNN reveal the added value of considering higher order moments. The LVQ classifier obtains good performances after a key statistical data reshaping and reveals limits deriving from the skewed distribution of the traffic sign examples. One-class classification aspects implied by the background patterns emerge as well; also the use of several prototypes hits cross-validation limits given by this dataset.

A proof-of-concept application is realized demonstrating real-time detection of traffic signs in digital images of urban roads.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [Knuth, 1974]

ACKNOWLEDGMENTS

I am heartily thankful to my supervisors, Michael H. F. Wilkinson and Michael Biehl, whose encouragement, guidance and support from the initial to the final stage enabled me to develop an understanding of the subject.

I am thankful to my readers Mohammad and Barend for helping me in finding poorly written and unclear parts.

Also a special thank goes to my lab fellows Natasja, Ka-Wing and Leo, who helped in making my stay at the computing science laboratory pleasant.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

CONTENTS

1 Introduction 1

I THEORY FUNDAMENTALS 5
2 Mathematical Morphology concepts 7
3 Max-Tree 13
4 Invariant Moments 17
4.1 Distance Measures 19
5 Applying Machine Learning 21
6 The Learning Algorithms 23
6.1 k Nearest Neighbours 23
6.2 Class-Conditional Means 24
6.3 Learning Vector Quantization 25
6.4 The ROC curve 26

II FEATURE EXTRACTION FROM IMAGES 29
7 Starting Data 31
7.1 Image Set 31
7.2 Target images 35
8 Labeling System 37
8.1 Labeling System Structure 37
8.2 Labeling Application 42
9 Labeling Analysis 49
9.1 Pure reference-based labeling 49
9.2 Supervised reference-based labeling 51
9.3 Supervised contrast-based labeling 51
9.4 Labeling considerations 53
10 Image Moments Data set 57

III DATA SET PREPROCESSING 59
11 Area-based pre-processing 61
12 Reference-based pre-processing 65
12.1 Plain Data 66
12.2 Normalized Data 71

IV APPLYING THE LEARNING ALGORITHMS 75
13 kNN Classification 77
13.1 Plain data 77
13.2 Normalized data 79
14 CCM Classification 81
14.1 Plain data 81
14.2 Normalized data 81
15 LVQ1 Classification 83
15.1 Plain data 89

V DISCUSSION 95
16 Demonstration 97
17 Conclusions 101
17.1 Labeling 101
17.2 Pre-processing 102
17.3 Classification 102

17.4 Overall 103
18 Future Work 105
18.1 Labeling 105
18.2 Pre-processing 106
18.3 Classification 106
18.4 Overall 106

BIBLIOGRAPHY 109

VI SUPPLEMENTARY MATERIAL 117
A Additional Pre-processing graphs 119
A.1 plain data graphs 119
A.2 ROC and class histograms examples 124
A.3 iqr-normalized data graphs 126
B Additional kNN Classification results 131
C Additional CCM Classification results 135
D Additional LVQ Classification results 139

1 INTRODUCTION

Image analysis is an ever developing field of modern science; the need to process images of any kind grows every day as the number of imaging sensors and displays increases continuously, together with the number and the variety of those imaging devices.

Likewise there is a growing interest in research and development of different techniques with increasing efficiency and flexibility within image analysis.

A particular sub-field of image analysis takes care of „understanding“ the (content of) images by means of detection and extraction of parts of images corresponding to interesting objects; this class of problems involves Image Segmentation (the process of partitioning a digital image into multiple regions), Object Recognition (the task of finding given object in an image or video sequence) and the Image Understanding (the task oriented reconstruction of a scene by means of images) among the others.

Mathematical morphology, a theory for the analysis of spatial structures started in the late 1960's by Georges Matheron and Jean Serra (see Sonka et al. [2007]), offers interesting tools with nice properties. In particular the extension to gray-level images of mathematical morphology's openings and closings deliver tools to partition a gray-level image into regions of pixels of the same gray intensity (the so-called *flat zones*). Salembier in Salembier et al. [1998] proposed an efficient method for partitioning an image by constructing the so-called *Max-Tree*, a component tree of flat zones.

Recently the Max-Tree has been thoroughly studied at the Intelligent Systems group[1] of the University of Groningen[2], and several applications and variations were developed. In particular several ways to evaluate the components of the Max-tree and to filter the tree itself were developed by thinking of sensible attributes and filtering rules; this allows to distinguish the flat zones on a sensible way.

A class of such attributes stems from an application of the mechanical moments to the Image Analysis domain; here extensive research led to different Invariants Moments which aim to describe the same features irrespective of translation, scaling, rotation and for some of them also irrespective of shear and other affine transformations which may be applied to any object (image parts).

The first and most known among those invariant moments are the Hu Moments developed by Hu Hu [1962], but more recently Flusser and Suk developed the Complex Moments Flusser and Suk [2003] and the Affine Moments Flusser and Suk [1993].

A new way to use such image moments as attributes for the Max-Tree's flat zones was developed mainly by Wilkinson and Urbach (see for instance in Urbach and Wilkinson [2001], Urbach et al. [2004, 2005, 2007], Urbach and Wilkinson [2002]). By applying those moments on selected reference images, attribute vectors are computed which can be seen as prototypes; the Max-Tree's flat zones are then accepted or rejected as being similar to the reference image by means of a small set of thresholds on parameters like the distance of their attribute vector from the reference vector and the minimal area of the flat zones.

This approach looks very promising, but still there is room for improvements. It is not clear which kind of moments is the most adequate for image and object representation or recognition. More generally, it is not clear which kind of image

1 http://www.rug.nl/informatica/onderzoek/
2 http://www.rug.nl

moments is the best one to express shape information without being sensitive to transformations like translation, scaling and rotation, or other issues like noise.

Furthermore, the required order of the image moments necessary to perform pattern recognition is not clear either, as well as the most appropriate (dis-) similarity measure within the attribute space(s); moreover, the necessary but undesirable burden of looking for the appropriate threshold parameter values for each new image by a human user is far from ideal.

Parallel to this research, at the very same Intelligent Systems group broad research is carried out about Machine Learning and its applications. Machine Learning (ML) is a field of science aiming to have automatic systems „learning“ to solve problems based on some (possibly growing) „experience“. Several tasks are involved, and classification is an important one.

A rather natural idea is then to use some Machine Learning technique to overcome the parameters choice problem of the morphological attribute filtering technique by learning the optimal settings for this problem. The learner algorithm will then lead to a classifier for traffic signs detection within the images.

In the Machine Learning field of science two major characterizations of techniques appear: *supervised* vs *unsupervised* learning. This characterization is implied by the presence of a supervisor providing feedback (a set of examples which are already labeled or classified) while learning, therefore the learner generalizes from sets of examples, or by the lack of any prior knowledge.

Within the Machine Learning group, the supervised learning algorithm Learning Vector Quantization (LVQ) [Kohonen, 1990, 1995, 1997, Somervuo and Kohonen, 1999] has been the subject of detailed analysis, regarding its dynamics and learning power [Biehl et al., 2005a,b, 2007b], different extensions and variations [Ghosh et al., 2005, Biehl et al., 2006, Schneider et al., 2007a,b, Bunte et al., 2008], and applications to complex classification problems like DNA microarray or spectral data classification [Biehl et al., 2007a, Schneider et al., 2008]. LVQ is a typical supervised offline learning technique.

As LVQ turns out to be a powerful and robust ML tool to deal with complex high-dimensional data, even in the case of small datasets, the choice to couple LVQ to the morphological attribute filtering seems convenient: the moment data is expected to be complex and medium- or high-dimensional, depending on the moment order.

The goal of this research is to achieve the coupling of these two techniques within the same pilot research problem where the morphological attribute filtering technique already demonstrates its power: the recognition of traffic signs within ordinary camera pictures of urban roads (see an example picture in figure 1).

A direct implication of this novel classification problem is that the morphological attribute filtering technique needs to be adapted in order to produce tagged datasets which are required to train the learning algorithm. Of course, as it often happens with supervised learning, training stages are coupled to test stages and in the end the performance is evaluated by a validation stage. All those stages need the very same kind of data, a tagged set of patterns which in this case is the expression of the Max-Tree's flat zones in terms of attributes.

To this end preparing work was carried out in [Bracci, 2008], where an existing prototype application implementing the morphological attribute filtering technique is adapted in order to produce suitable training datasets. There a human supervisor drives the system to create a *ground truth*, a data set where all the connected components features are correctly tagged by a class label.

Continuing from this work, the first goal of this project is to process the given set of urban road images and to produce a feature vectors data set which represents the shapes of the connected components of the real images.

Figure 1: example of urban road picture

Once the feature data set is available, extensive study by means of Pattern Recognition techniques like kNN (k Nearest Neighbors) and Kohonen's LVQ [Somervuo and Kohonen, 1999] becomes possible. Here the second goal arises: to study the behavior of these learning techniques in order to assess the learnability of the traffic signs by means of shape features.

As the recognition problem turns out to be learnable, the last research question is how to put these pieces together to automate the "manual decision making" into a seamless process; only with a fully automated process, which autonomously tags the traffic signs within images, real world applications come within sight.

The relevance of this project derives from Computer Vision applications within the automotive industry: an example could be a driving support system which reminds the driver of unnoticed traffic signs. Even more advanced would be a robotic driving system, which would rely on road signs as well. Whenever this technique will be successful, even more challenging applications will come in sight, like advanced processing of biomedical data produced by CAT[3], MRI[4] and PET[5] scans.

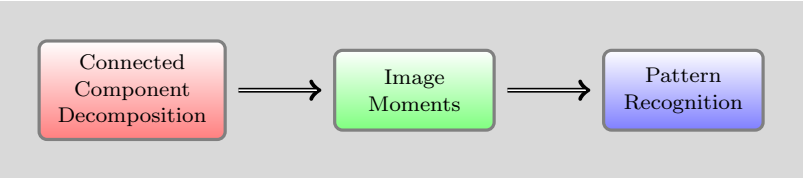
This thesis is structured as follows: first the theory layers are introduced in part I on page 6; then the construction of a labeled data set is described in part II on page 30; the arisen pre-processing of the data set is described in part III on page 60; the detection and recognition are described in part IV on page 76; finally a demonstration prototype, the conclusions and future work are described in part V on page 96.

3 Computed Axial Tomography
4 Magnetic Resonance Imaging
5 Positron Emission Tomography

Part I

THEORY FUNDAMENTALS

The automated traffic sign recognition relies on different layers of theory, namely the Max-Tree for the connected components decomposition, the (Invariant) Moments for the shape evaluation of the connected components and the Pattern Recognition techniques for the shape detection of the connected components; see the figure below for a diagram representation of the theory layers.



The three theory layers.

In this part the basics of the different theory layers are introduced: first some image analysis concepts are introduced in chapter 2, then the Max-Tree and its computation are explained in chapter 3, the different kinds of (invariant) moments are treated in chapter 4; the learning problem context is set out in chapter 5 and finally the learning techniques are explained in chapter 6. Chapters 3, 4 and 5, are a revised and extended versions of material already set out in [Bracci, 2008].

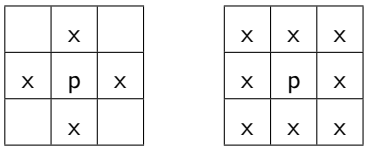
2 MATHEMATICAL MORPHOLOGY CONCEPTS

In image analysis an image is thought of as a (continuous) two-dimensional function $f(x,y)$, where x and y are *spatial* coordinates, and the amplitude of f at any pair of coordinates (x,y) is called the *intensity* or *gray level* of the image at that point [Gonzales and Wintz, 1987, Gonzalez and Woods, 2002, 2006].

This function definition considers a continuous domain and co-domain; in practice images are sensed and processed by *digital* devices, therefore in digital image processing the coordinates x and y become discrete as well as the function values; this happens because of the involved discretization steps (sampling and quantization) and consequently f becomes a discrete lattice and can be seen as a matrix of discrete values.

For this reason from now on an image f is regarded as a $M \times N$ matrix of picture elements or *pixels* and $f(x,y)$ is regarded as the pixel value at row and column indexes x and y.

Adjacency between pixels is then a fundamental concept which simplifies further definitions. To this end the following *adjacencies* of a pixel p are introduced: two pixels are said to be 4-*adjacent* if they belong to the set $N_4(p)$, the set of the four horizontal and vertical neighbour pixels. Similarly, two pixels are said to be 8-*adjacent* if they belong to the set $N_8(p)$, the set of the 4-adjacent pixels and the four diagonal neighbour pixels. See figure 2 for an illustration of both the adjacencies.



(a) 4-adjacency (b) 8-adjacency

Here p represents a pixel while x are a neighbour pixels belonging to the relative adjacency.

Figure 2: Pixel adjacencies.

Based on the concept of adjacency, it is possible to define a path within an image: a (*digital*) *path* or *curve* P in an image f for a given adjacency relation N is the sequence of neighbouring points $p_0 = (x_0, y_0), \ldots, p_n = (x_n, y_n)$, of which any consecutive pair belongs to the adjacency relation N. The usual digital path formally is

$$P = \{(x_i, y_i)\}_{i=0}^{n} \quad \text{where} \quad [(x_k, y_k), (x_{k+1}, y_{k+1})] \in N \tag{2.1}$$

In mathematical morphology it is customary to consider an image as a set of points; within this context image subsets (e.g. X) are used to indicate image parts or to act as selection masks. It is also customary to refer to a subset X of an image f also as a selector of a part of the whole digital image.

Below a number of basics concepts are introduced based on the notation used in [Urbach and Wilkinson, 2002, Urbach et al., 2005, 2007, Ouzounis and Wilkinson, 2007, Salembier and Wilkinson, 2009, Wilkinson, 2009a]; these concepts stem from Mathematical Morphology literature and are useful later on.

The usual digital path leads to the idea of connected components: for any pixel $p_0 = (x_0, y_0)$ within a subset X of an image f, the maximal set of pixels $p_i = (x_i, y_i)$ which are connected to p_0 by a digital path P within S is said to be

a *connected component* or *grain* of X. If X contains a single connected component, it is said to be a *connected set*. Formally, the connected component of an image subset X for point p_0 is the set of pixels

$$C(p_0, X) = \{(x, y) \mid P[(x_0, y_0), (x, y)] \cap X \neq \emptyset\} \tag{2.2}$$

whenever the subset X coincides with the whole image, it can be left out.

Considering the pixel gray level, the idea of flat image parts is proposed in [Serra and Salembier, 1993, Salembier and Serra, 1995]: the *flat zone* L_h^k is a maximal connected component of intensity h. If only the gray level h is considered, the set of all points at a certain *level* (gray intensity) h in a (gray) image, called the *Level Set*, is defined:

$$L_h = \{(x, y) \mid f(x, y) = h\} \tag{2.3}$$

Multiple flat zones may be present at level h and are indexed in the level set L_h by the natural number k: $L_h^k = C_k \subseteq L_h$.

In [Serra and Salembier, 1993, Salembier and Serra, 1995] the following definitions related to a partition are put forward: a *partition* $\mathcal{P}$ of a *space* X is a set of connected components $\{C_i\}$ which are disjoint ($C_i \bigcap C_j = \emptyset, i \neq j$) and the union of which is the entire space ($\bigcup C_i = X$). Each C_i is called a partition class.

Let us denote a partition by $\mathcal{P}$ and the region of $\mathcal{P}$ that contains pixel n by $\mathcal{P}(n)$. A partial order relationship among partitions can be created: $\mathcal{P}_1$ is *finer than* $\mathcal{P}_2$ (written as $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$), if $\forall n, \mathcal{P}_1(n) \sqsubseteq \mathcal{P}_2(n)$. The analogous *coarser than* partial order relationship among partitions is written as $\mathcal{P}_1 \sqsupseteq \mathcal{P}_2$. Note that the level sets L_h of an image induce a partition, the partition of the flat zones.

In [Salembier et al., 1998] a *binary connected operator* is proposed through the use of partitions: a binary operator ψ is connected if and only if its associated partition is finer than the original one ($\mathcal{P}_{\psi(X)} \sqsubseteq \mathcal{P}_X$).

Similar to the level set is the *threshold set*, the set of all the points at a certain level h or higher in an image, which is formally given by

$$T_h = \{(x, y) \mid f(x, y) \geqslant h\} \tag{2.4}$$

Similar to the flat zone, the threshold set T_h can be subdivided in smaller sets, the *peak components* P_h^k which are connected components of minimal intensity h. Multiple peak components may be present at level h and indexed within the threshold set T_h by the natural index k: $P_h^k = C_k \in T_h$. Note that the peak components of a threshold set T_h are contained by the peak components of a lower threshold set $T_{k<h}$.

It is useful to characterize interactions of a set operator ψ on an image subset X of an image f as *increasing* if $Y \subseteq X \Longrightarrow \psi(Y) \subseteq \psi(X)$, *idempotent* if $\psi(\psi(X)) = \psi(X)$, *extensive* if $\psi(X) \subseteq X$ and *anti-extensive* if $X \subseteq \psi(X)$. With these definitions it is possible to characterize a *morphological filter* as an increasing and idempotent operator; an *opening* is then a anti-extensive morphological filter while a *closing* is a extensive morphological filter.

Connected openings are treated in [Serra and Vincent, 1992]; a *connected opening* Γ_p of a image subset X with grains C_i at point p is

$$\Gamma_p(X) = \begin{cases} C_i & p \in C_i \\ \emptyset & p \notin X \end{cases} \tag{2.5}$$

this operator is idempotent, anti-extensive and increasing; its application selects precisely the grain C_i of X containing the point p.

In the used litterarture also an alternative definition is present, based on connectivity classes. A *connectivity class* $\mathcal{C}$ on an image subset X is any family in $\mathcal{P}_X$ for which

$$\begin{cases} \emptyset \in \mathcal{C}, & \forall p \in X \{p\} \in \mathcal{C} \\ \forall \{C_i\} \in \mathcal{C} & \bigcap_i C_i \neq \emptyset \implies \bigcup_i C_i \in \mathcal{C} \end{cases} \tag{2.6}$$

Now a *connected opening* Γ_p of a image subset X with connectivity class $\mathcal{C}$ and grains C_j within $\mathcal{C}$ can be constructed as

$$\Gamma_p(X) = \bigcup_j \{C_i \in \mathcal{C} \mid p \in \mathcal{C} \wedge C_j \subseteq X\} \tag{2.7}$$

where the point p selects all the grains in the connectivity class $\mathcal{C}$ containing it; the union of all such grains returns the maximal connected component containing p.

Substituting the point selection by more general criteria leads to the trivial opening: a *trivial opening* Γ_T of a image subset X with grains C_i for increasing binary criterion T is

$$\Gamma_T(C_i) = \begin{cases} C_i & \text{if} \quad T(C_i) \\ \emptyset & \text{if} \quad \neg T(C_i) \text{ or } C_i = \emptyset \end{cases} \tag{2.8}$$

where the criterion T usually is of the form $T(C) = (\mathrm{Attr}(C) \geqslant \lambda)$ with $\mathrm{Attr}(C)$ a real quantity deriving from C and λ a threshold value; its application selects all the grains C_i of S which satisfy the increasing criterion T.

Combining a trivial opening with a connected opening leads to the attribute opening; more precisely, an *attribute opening* Γ^T of a image subset X with grains C_i for increasing binary criterion T is

$$\Gamma^T(X) = \bigcup_{p \in S} \Gamma_T(\Gamma_p(X)) \tag{2.9}$$

which is the application of the trivial opening to each image grain selected by the application of the connected opening Γ_p. This way, only the grains which satisfy the increasing criterion T are preserved. The attribute opening is increasing, idempotent and anti-extensive [Breen and Jones, 1996].

Similar to the trivial opening is the trivial thinning [Breen and Jones, 1996, Salembier et al., 1998], which relaxes the criterion T by not requiring the increasingness. The *trivial thinning* Φ_T of a grain C ($\Phi_T(C)$) of an image subset X with binary criterion T therefore is defined as

$$\Phi_T(C_i) = \begin{cases} C_i & \text{if} \quad T(C_i) \\ \emptyset & \text{if} \quad \neg T(C_i) \text{ or } C_i = \emptyset \end{cases} \tag{2.10}$$

where its application selects all the grains C_i of X which satisfy the criterion T.

The application of the trivial thinning to all the connected components of an image subset X is the *attribute thinning* Φ^T with binary criterion T and connected components C_i:

$$\Phi^T(X) = \bigcup_{p \in S} \Phi_T(\Gamma_p(X)) \tag{2.11}$$

which is the application of the trivial thinning to each image grain selected by the application of the connected opening Γ_p. This way, only the grains C_i of X which satisfy the increasing criterion T are preserved. The attribute thinning is a idempotent and anti-extensive operator.

So far binary operators are introduced; in order to treat gray images as well those definitions need to be extended by using the level sets L_h of an an image. The operators extended to gray level images are symbolised by lower case Greek letters; the gray intensity here is a positive natural number ($h \in \mathbb{N}_0$).

Similar to the binary case, a connected operator is defined as follows: a *gray-level connected operator* is an operator ψ which induces a *finer* partition of flat zones, which is $\mathcal{P}_f \sqsubseteq \mathcal{P}_{\psi(f)}$.

Theoretical aspects of gray-level morphology are analyzed in [Heijmans, 1991] while gray-scale trivial openings are formalised in [Breen and Jones, 1996]. Generalizing definition (2.5) leads to the definition of a *gray-scale connected opening* γ_p of an image subset X with level set $L_h(X)$ at point p as

$$\gamma_p(X) = \max(\{0\} \cup \{h \mid \Gamma_p(L_h(X)) \neq \emptyset\}) \tag{2.12}$$

which is the application of the binary connected opening to all the different level sets where only the grains containing the point p are selected; the result is the maximal gray intensity among the fulfilling connected components. In this way the increasingness, idempotence and anti-extensieveness of the binary connected opening are preserved.

In a similar way the binary trivial opening of definition (2.8) is generalized to gray-level images as follows: a *gray-scale trivial opening* γ_T of a image subset X with level set $L_h(X)$ for increasing binary criterion T is

$$\gamma_{p,T}(X) = \max(\{0\} \cup \{h \mid \Gamma_T(\Gamma_p(L_h(X))) \neq \emptyset\}) \tag{2.13}$$

which is the application of the binary trivial opening to all the grains selected by point p in the different level sets $L_h(X)$; the maximal gray intensity is then returned.

Analogous to the construction of the binary attribute opening (2.19) is the construction of the gray-scale attribute opening: a *gray-scale attribute opening* γ^T of an image subset X with level set $L_h(X)$ for increasing binary criterion T is

$$\gamma^T(X) = \max\left(\{0\} \cup \left\{h \mid \Gamma^T(L_h(X)) \neq \emptyset\right\}\right) \tag{2.14}$$

which is the maximal gray intensity among all the grains in the level set $L_h(X)$ which satisfy criterion T.

To illustrate the gray-scale connected opening and the gray-scale attribute opening a couple of examples are shown in figure 3 on the next page.

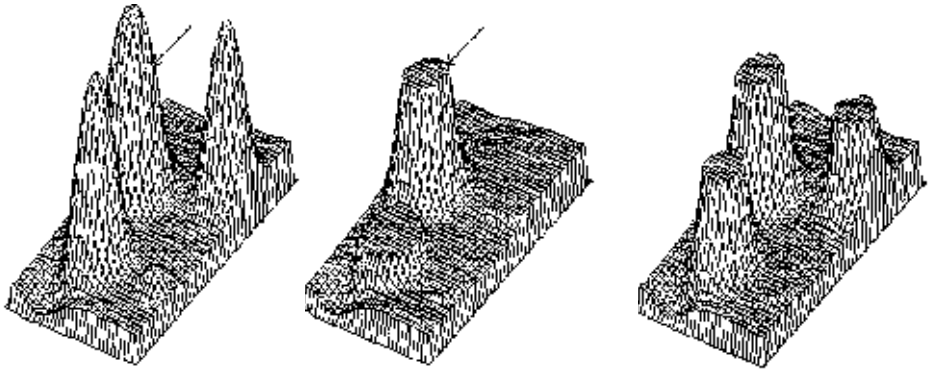
Similar to the construction of the gray-scale trivial opening of definition (2.15) is the the gray-scale trivial thinning: a *gray-scale trivial thinning* ϕ_T of a image subset X with level set $L_h(X)$ for increasing binary criterion T is

$$\phi_{p,T}(X) = \max(\{0\} \cup \{h \mid \Phi_T(\Gamma_p(L_h(X))) \neq \emptyset\}) \tag{2.15}$$

which is the application of the binary trivial thinning to all the grains selected by point p in the different level sets $L_h(X)$; the maximal gray intensity is then returned.

Also the gray-scale attribute thinning is very similar to the gray-scale attribute opening: a *gray-scale attribute thinning* ϕ^T of an image subset X with with level set $L_h(X)$ for increasing binary criterion T is

$$\phi^T(X) = \max\left(\{0\} \cup \left\{h \mid \Phi^T(L_h(X)) \neq \emptyset\right\}\right) \tag{2.16}$$



(a) A gray level image. (b) The image after performing the connected opening γ_p (c) The image after performing the attribute opening γ^T.

In the above sub-figures the point p is indicated by an arrow; the increasing criterion T here accepts grains with area larger than 50. Figures taken from Breen and Jones [1996].

Figure 3: Example of connected and attribute openings.

which is the maximal gray intensity among all the grains in the level set $L_h(X)$ which satisfy criterion T.

More general attribute opening and thinning are not limited to a single attribute, therefore the multivariate versions can be defined: a *multivariate gray-scale attribute opening* $\gamma^{\{T_i\}}$ of an image subset X for the increasing criteria set $\{T_i\}_{i=1}^n$ is

$$\gamma^{\{T_i\}}(X) = \bigcap_{i=i}^{n} \gamma^{T_i}(X) \tag{2.17}$$

which preserves the grains of the level set $L_h(X)$ that satisfy each of the increasing criteria $\{T_i\}$, while a *multivariate gray-scale attribute thinning* $\phi^{\{T_i\}}$ of an image subset X for the criteria set $\{T_i\}_{i=1}^n$ is

$$\phi^{\{T_i\}}(X) = \bigcap_{i=i}^{n} \phi^{T_i}(X) \tag{2.18}$$

which in similar way preserves the grains in the level set $L_h(X)$ that satisfy each of the criteria $\{T_i\}$.

The different thresholds of the criteria $\{T_i\}$ might be condensed into attributes vector $\vec{\tau}$ and reference vector $\vec{r}$; in this case the criteria can be substituted by a vector distance $d(\vec{\tau},\vec{r})$ and a distance threshold ϵ. This leads to the definition of vector *attribute thinning* of an image subset X with with level set $L_h(X)$ for binary criterion $T_\epsilon^{\vec{r}}$:

$$\phi_\epsilon^{\vec{r}}(X) = \max\left(\{0\} \cup \left\{h \,|\, \Phi^{T_\epsilon^{\vec{r}}}(L_h(X)) \neq \emptyset\right\}\right) \tag{2.19}$$

where $T_\epsilon^{\vec{r}}$ is the distance-based criterion imposing the maximum distance ϵ from the attributes vector $\vec{\tau}$ to the reference vector $\vec{r}$ on the selection of the grains of X.

3

MAX-TREE

The gray-level connected operators imply a structured representation of the image by means of flat zones [Salembier et al., 1998] as discussed in chapter 2; this observation leads to the definition of the Max-Tree.

A Max-Tree is a structured representation of an image in tree form, oriented to the maximum gray intensity value (i.e. the leaves refer to local gray intensity maxima, while the root refers to the minimum gray intensity) and to the application of anti-extensive operators.

The tree is computed by recursively considering all the thresholded versions of the image, from the minimum gray intensity to the maximum one. A thresholded version of the image is the set of image pixels of intensity level larger than some threshold level h, which is the threshold set T_h of definition (2.4).

At each recursion step, all the pixels belonging to a temporary node TC_h^k (representing the k-th peak component P_h^k from the threshold set T_h) undergo a *binarization* step; then the *connected components definition* step takes place.

In the binarization step all the pixels of the present gray intensity are considered as "local background" and assigned to the current node C_h^k (the $k-$th node found for gray level h): $C_h^k = \{(x,y) \in P_h^k | f(x,y) = h\}$, therefore , $C_h^k \in L_h$.

In the connected components definition step, the remaining pixels belonging to $TC_h^k \setminus C_h^k$ are assigned to new temporary nodes TC_{h+1}^k at level $h+1$, one for each peak component. This is achieved by labelling those pixels which do not belong to the "local background" following the chosen connectivity rule (4-, 8- or mixed-connectivity). This connectivity-based labelling is important as it implicitly defines the objects which will be represented by the Max-Tree nodes, even after successive processing.

Eventually, if C_h^k is empty, it is removed from the Max-Tree and recursion is started upon the set of peak components $\{TC_{h+1}^k\}_k$; an example of Max-Tree construction is shown on figure 4.

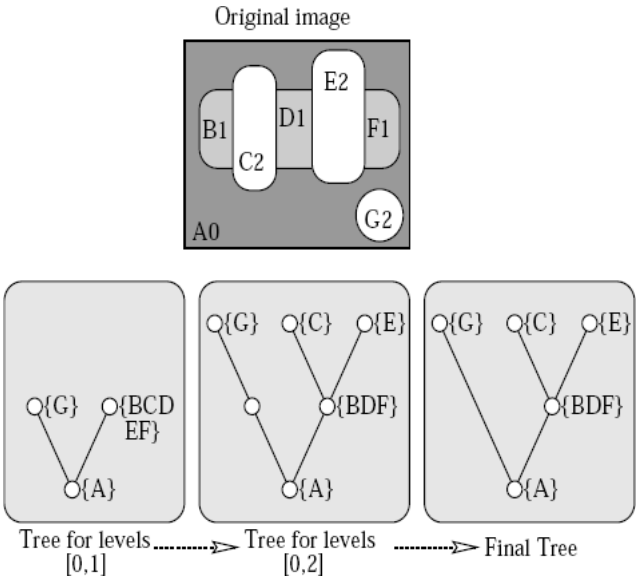


Figure 4: Max-Tree construction procedure; figure taken from Salembier et al. [1998].

Within the same research work [Salembier et al., 1998] an efficient algorithm for fast Max-Tree computation is presented; this is shown in algorithm 3.1 on the next page.

It is important to notice that the Max-Tree in the end represents a decomposition in flat zones of a single gray intensity. Any node of the Max-Tree, the Max-Nodes, derives from flooding a single connected component of a threshold set; if a gray scale image is considered as a height map, such a node corresponds with a *single band* between two successive heights along a mountain peak.

The Max-Tree construction is driven by the connected components while its interpretation is not bound by them; often its nodes are regarded as peak components, which are all the gray intensity bands from some gray level up to the mountain top. This implies that at any node the corresponding image part is found by considering the whole subtree rooted at that node.

This has important implications for the interpretation of the Max-Tree. For instance, when looking at the flat zones, the root node represents all the (possibly disjoint) connected components of minimum gray level intensity. When looking at the peak components instead, the same root node represents the pixels of the whole image.

The aim now is to look efficiently at the peak components and this is possible by modifying the procedure of algorithm 3.1; the new procedure constructs Max-Nodes directly corresponding to peak components.

This new algorithm contains just few differences. Only a small step is introduced after each recursive call returns; at that point, the content of the children Max-Nodes is merged with the "local background" for any Max-Node. The whole extended Max-Tree construction procedure is set out in algorithm 3.2 on page 16. The tree structure now represents the inclusion relationship among the peak components, which are now the Max-Tree nodes.

Once the Max-Tree is set up, it can be manipulated by the gray-intensity operators described in chapter 2; these operators simplify tree structures by removing nodes. The simplified tree can be used to reconstruct an image with the interesting parts, depending on the specific application. The used simplification criterion determines which regions are removed and may depend e.g. on size, contrast, texture, motion or shape.

The simplification strategies are subdivided in pruning and non-pruning [Salembier and Wilkinson, 2009]; pruning strategies may remove entire node sub-trees while non-pruning strategies may remove any number of nodes independent of the node location in the tree.

Pruning strategies may rely on increasing criteria; in the Max-Tree case, pruning strategies remove regional intensity maxima.

For the non-increasing pruning strategies, two filtering rules are known:

- The *min rule* removes all the nodes from the leaves up to the first preserved node;
- The *max rule* removes all the nodes from the leaves up to the last preserved node.

For the non-pruning strategies, two filtering rules are known:

- The *direct rule* removes all the nodes which do not fulfil the criterion, irrespective of their place in the tree; the corresponding regions are assigned to the first preserved ancestor;
- the *subtractive rule* performs the same actions as the direct rule and the gray intensity of the regions to merge are lowered as well.

In particular, the last two rules are gray-intensity (multivariate) thinnings as defined in (2.16) and (2.18).

Algorithm 3.1 FLOODING PROCEDURE FOR THE MAX-TREE CREATION

hierarchical FIFO queue operators:

HQUEUE-ADD(h, p) adds the pixel p of gray level h in the queue of priority (level) h;

HQUEUE-FIRST(h) extracts the first available pixel p of the queue of priority h;

HQUEUE-EMPTY(h) returns the boolean truth value for the queue of priority (level) h being empty.

Further notation:

ORI(p) denotes the original gray level of the pixel p

STATUS(p) stores the status information for pixel p:
NOT_ANALYZED, IN_THE_QUEUE or assigned to the node k of level h

NUMBER-NODES(h) is the number of nodes C_h^k at level h.

The original algorithm for fast Max-Tree computation is as follows:

```
flood(h)                                  /* Flood gray level h */
while not hqueue-empty(h)                 /* First step: propagation */
  p <- hqueue-first(h)                    /* while pixels in the pro-
  STATUS(p) <- number-                    cess queue */
nodes(h)                                  /* process pixel p */
  for every neighbor q of p               /* 4 or 8 connectivity */
  if STATUS(q)==NOT_ANALYZED              /* if a new pixel is found */
    hqueue_add(ORI(q),q)                  /* add to the process queue */
    STATUS(q) <- IN_THE_QUEUE
    node_at_level(ORI(q)) <-
TRUE                                      /* found a child flat zone at lev. q */
    if (ORI(q) > ORI(p))
      m <- ORI(q)                         /* for each gray-
      repeat                              level from the new one
                                          down to the present */
       m <- flood(m)                      /* flood the child */
      until m==h                          /* queue empty -
number-nodes(h)++                         flooding on h finished
                                          - update nr nodes*/

m <- h-1                                  /* Second step:define paren-
while m >= 0 and !node_at_level(m)        t/child rel.*/
m--                                       /* look for the father */
if m >= 0
i <- number-nodes(h)-1
j <- number-nodes(m)
father C_h^i<-node C_m^j
else                                      /* assign the father */
C_h^i has no fa-
ther (C_m^j is the root node)
node_at_level(h) <- FALSE                 /* no father - it's the root node */
return m
```

Algorithm 3.2 Adapted Flooding procedure for the Max-Tree creation

Added notation:

`ZERO_VEC` denotation for a vector with zero values ($\vec{0}$).

`COORD(p)` denotation for coordinates of pixel `p`.

Added operators:

`UPDATE-MOMENTS(ATTR,COORD(p))` adds the coordinates of pixel `p` to the geometric moments `attr`

`MERGE-MOMENTS(ATTR,NEWATTR)` „merges“ (vector sum) the geometric moments `attr` with `newattr`

The adapted algorithm for fast Max-Tree computation with geometric moments is as follows (added parts are underlined):

```
flood(h,prevarea,prevattributes)

  area = prevarea
  attr = prevattr


 while not hqueue-empty(h)
    area <- area +1
    p <- hqueue-first(h)
    STATUS(p) <- number-nodes(h)
    update(attr, coord(p))
    for every neighbor q of p
    if STATUS(q)==NOT_ANALYZED
      hqueue_add(ORI(q),q)
      STATUS(q) <- IN_THE_QUEUE
      node_at_level(ORI(q)) <- TRUE
      if (ORI(q) > ORI(p))
        m <- ORI(q)
        childarea <- 0
        childattr <- zero_vec
        repeat

         m <- flood(m)
        until m==h

        area <- area + childarea
        merge-moments(attr,childattr)
number-nodes(h)++


m <- h-1
while m >= 0 and !node_at_level(m)
  m--
if m >= 0
  i <- number-nodes(h)-1
  j <- number-nodes(m)
  father C_h^i<-node C_m^j
else
  C_h^i has no fa-
ther (C_m^j is the root node)
node_at_level(h) <- FALSE

node.Area <- area
node.Attribute <- attr
node.Vector <- NULL
prevarea <- area
prevattr <- attr
return m
```

```
/* Flood gray level h - prevarea and
   prevattribute references from caller */
/* init area with area prev. peak comp. */
/* init attr with attr prev. peak comp. */
/* only in inner loop calls are nonzero*/
/* First step: propagation */
/* while pixels in the process queue */
/* process p: update area MaxNode */
/* process p: update status MaxNode */


/* 4 or 8 connectivity */
/* if a new pixel is found */
/* add to the process queue */


/* a child flat zone is found */

/* init child's area to 0 */
/* init child's attr to zero_vec */
/* for each gray-level from the
 new one down to the present */
/* flood child from the new gray value */
/* until pres gray level reached */

/* incr the pres area by the flooded */
/* incr the pres momts by the flooded */
/* queue empty - flooding on h ends
- update nr nodes */

/* Second step: determine father */
/* look for the father */




/* assign the father */

/* no father - it's the root node */


/* Last step: Manage recursion pars */
/* assign flooded area to node */
/* assign flooded moments to node */
/* initialize node attributes */
/* update ref to prev area */
/* update ref to prev moments */
```

4

INVARIANT MOMENTS

So far a sensible method to compute an image decomposition is introduced; now (possibly different) kind(s) of valuations for the peak components are needed in order to distinguish them based on their *shape*. To this extent different *image moments* are introduced; these image moments form convenient *vectors* of *features* (the valuations) which make it possible to relate the peak components to each other by means of quantitative shape similarity. The moments play therefore the role of *shape features*. For this reason it is useful to pay attention at *invariant* features, which is features having similar values for entities differing e.g. by a simple translation or rotation.

Similar to the definition of moments in mathematics and classical mechanics, which is

$$\mu'_n = \int_{-\infty}^{+\infty} x^n \cdot f(x)\, dx \quad (n = 0, 1, 2, \ldots) \tag{4.1}$$

the 2D moments of order $(p + q)$ of a continuous image irradiance distribution $f(x, y)$ are defined by

$$m_{p,q} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p \cdot y^q \cdot f(x, y)\, dx\, dy \quad (p, q = 0, 1, 2, \ldots) \tag{4.2}$$

For digital images the above becomes the discretized variant

$$M_{p,q} = \sum_x \sum_y x^p \cdot y^q \cdot f(x, y) \quad (p, q = 0, 1, 2, \ldots) \tag{4.3}$$

where x and y vary describing the coordinates set of the image.

Equation (4.3) is often referred as the definition for the *Geometric Moments*. These moments are known to be sensitive not only to rotation and scaling but also to translation: by simply shifting the image towards higher coordinate values, the moment values increase. This is expected to limit their usefulness for the image recognition task by looking only at a derived moment set.

Analogous to the normalization for data with non-zero mean, the image centroids can be used to to remove sensitivity to translation. The coordinates of the center of mass are given by

$$\overline{x} = \frac{M_{1,0}}{M_{0,0}} \quad , \quad \overline{y} = \frac{M_{0,1}}{M_{0,0}} \tag{4.4}$$

The *Central Moments* are then defined by:

$$\mu_{p,q} = \sum_x \sum_y (x - \overline{x})^p \cdot (y - \overline{y})^q \cdot f(x, y) \quad (p, q = 0, 1, 2, \ldots) \tag{4.5}$$

Still these moments are sensitive to scale changes and rotation. For instance, homogeneous scaling enlarges or shrinks images and parts of them, effectively changing the number of pixels and their coordinates involved in the summation. A rigid rotation instead, does not change the amount of involved pixels but instead the set of pixels changes. In both cases there is no invariance.

In order to remove changes due to scaling, the factor

$$\gamma = (p + q/2) + 1 \tag{4.6}$$

is introduced and then the *Normal Central Moments* are defined as:

$$\eta_{p,q} = \mu_{p,q}/\mu_{0,0}^{\gamma} \tag{4.7}$$

Here the scaling correction proportional to the *size* of the image adds invariance to homogeneous scale changes but still value changes due to rigid rotations are not considered.

From the normalized central moments a set of seven moments invariant to rotation are derived in [Hu, 1962]:

$$\phi_1 = \eta_{2,0} + \eta_{0,2} \tag{4.8}$$

$$\phi_2 = (\eta_{2,0} + \eta_{0,2})^2 + 4\eta_{1,1}^2 \tag{4.9}$$

$$\phi_3 = (\eta_{3,0} + 3\eta_{1,2})^2 + (\eta_{0,3} + \eta_{2,1})^2 \tag{4.10}$$

$$\phi_4 = (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{0,3} + \eta_{2,1})^2 \tag{4.11}$$

$$\begin{aligned}\phi_5 = {} & (3\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2}) \cdot \\ & \cdot \left[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2\right] + \\ & + (3\eta_{2,1} - \eta_{0,3})\left[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2\right]\end{aligned} \tag{4.12}$$

$$\begin{aligned}\phi_6 = {} & (\eta_{2,0} + \eta_{0,2})\left[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2\right] + \\ & + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3})\end{aligned} \tag{4.13}$$

$$\begin{aligned}\phi_7 = {} & (3\eta_{2,1} - \eta_{0,3})(\eta_{3,0} + \eta_{1,2}) \cdot \\ & \cdot \left[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2\right] + \\ & + (3\eta_{1,2} - \eta_{3,0})(\eta_{0,3} + \eta_{2,1}) \cdot \\ & \cdot \left[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{3,0})^2\right]\end{aligned} \tag{4.14}$$

In a series of papers (among others [Flusser and Suk, 1993] (1992/3/4)) Flusser and Suk discussed how the Hu moments, which in the meanwhile were widely known and applied, are neither a complete base of moments nor fully discriminative with respect to reflection (there is a particular axial symmetry where those moments fail to discriminate two images, for details see [Flusser and Suk, 1993]). The completeness of a base (of moments) is the prerequisite to fully describe a space (of moments).

The same authors derived in [Flusser and Suk, 2003] another set of invariant moments, based on the so called *complex moments*, defined as

$$c_{p,q} = \sum_{k=0}^{p}\sum_{j=0}^{q}\binom{p}{k}\cdot\binom{q}{j}\cdot(-1)^{q-j}\cdot i^{p+q-k-j}\cdot m_{k+j,p+q-k-j} \qquad (p, q = 0, 1, 2, \ldots) \tag{4.15}$$

On top of these complex moments a base of moments are defined which fulfil the invariant properties to translation, scaling and rotation, the *Complex Moment Invariants* (CMI) (shown up to the 3^{rd} order for $p_0 = 2$ and $q_0 = 1$):

$$\Phi(1,1) = c_{1,1} \tag{4.16}$$
$$\Phi(2,1) = c_{2,1} \cdot c_{1,2} \tag{4.17}$$
$$\Phi(2,0) = c_{2,0} \cdot c_{1,2}^2 \tag{4.18}$$
$$\Phi(3,0) = c_{3,0} \cdot c_{1,2}^3 \tag{4.19}$$

As a result of further work Flusser and Suk [1993], Flusser and Suk derived also another set of invariant moments, the *Affine Moment Invariants* (AMI):

$$I_1 = \frac{1}{\mu_{0,0}^4}\left(\mu_{2,0}\mu_{0,2} - \mu_{1,1}^2\right) \tag{4.20}$$
$$\begin{aligned} I_2 &= \frac{1}{\mu_{0,0}^{10}}\left(\mu_{3,0}^2\mu_{0,3}^2 - 6\mu_{3,0}\mu_{2,1}\mu_{1,2}\mu_{0,3} + 4\mu_{3,0}\mu_{1,2}^3 + \right. \\ &+ \left. 4\mu_{0,3}\mu_{2,1}^3 - 3\mu_{2,1}^2\mu_{1,2}^2\right) \end{aligned} \tag{4.21}$$
$$\begin{aligned} I_3 &= \frac{1}{\mu_{0,0}^7}\left(\mu_{2,0}\left(\mu_{2,1}\mu_{1,3} - \mu_{1,2}^2\right) - \mu_{1,1}\left(\mu_{3,0}\mu_{0,3} - \mu_{2,1}\mu_{1,2}\right) + \right. \\ &+ \left. \mu_{0,2}\left(\mu_{3,0}\mu_{1,2} - \mu_{2,1}^2\right)\right) \end{aligned} \tag{4.22}$$
$$\begin{aligned} I_4 &= \frac{1}{\mu_{0,0}^{11}}\left(\mu_{2,0}^3\mu_{0,3}^2 - 6\mu_{2,0}^2\mu_{1,1}\mu_{0,2}\mu_{0,3} - 6\mu_{2,0}^2\mu_{0,2}\mu_{2,1}\mu_{0,3} + \right. \\ &+ -9\mu_{2,0}^2\mu_{0,2}\mu_{1,2}^2 + 12\mu_{2,0}\mu_{1,1}^2\mu_{2,1}\mu_{0,3} + 6\mu_{2,0}\mu_{1,1}\mu_{0,2}\mu_{3,0}\mu_{0,3} + \\ &+ -18\mu_{2,0}\mu_{1,1}\mu_{0,2}\mu_{2,1}\mu_{1,2} - 8\mu_{1,1}^3\mu_{3,0}\mu_{0,3} - 6\mu_{2,0}\mu_{0,2}^2\mu_{3,0}\mu_{1,2} + \\ &+ \left. 9\mu_{2,0}\mu_{0,2}^2\mu_{2,1}^2 + 12\mu_{1,1}^2\mu_{0,2}\mu_{3,0}\mu_{12} - 6\mu_{1,1}\mu_{0,2}\mu_{3,0}\mu_{2,1} + \mu_{0,2}^3\mu_{3,0}^2\right) \end{aligned} \tag{4.23}$$

For these moments mathematical proof of invariance to rotation and affine transformations is given in [Flusser and Suk, 1993].

4.1 DISTANCE MEASURES

The invariant moment which describe the peak components can be used to evaluate how *similar* different peak components are. In our case we look at the similarity to reference moments.

Distance definitions are required in order to have quantitative measures of the similarity between moments. In general a distance might be tied to a particular domain, but without any a priori knowledge the very generic *Minkowsky distance* is chosen to measure the similarity between moment vectors $\vec{x}$ and $\vec{y}$:

$$d_{L_p}(\vec{x},\vec{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{4.24}$$

Of course other choices for the used metrics are possible. In particular three measures are chosen, all derived from the Minkowski metric:

$$d_{L_1}(\vec{x},\vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.25}$$

$$d_{L_2}(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{4.26}$$

$$d_{L_\infty}(\vec{x},\vec{y}) = \max|x_i - y_i| \tag{4.27}$$

Equation (4.25) is known as the L_1 (Manhattan) distance, equation (4.26) as the L_2 (Euclidean) distance and equation (4.27) as L_∞ (Infinity norm) distance.

In Urbach's traffic sign detection application another measure was introduced:

$$d^{\vec{r}}_{L_2}(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{n} \delta^2_{\vec{r}}(x_i - y_i)} \tag{4.28}$$

where $\vec{r}$ indicates that $\vec{y}$ is interpreted as the reference moments vector and

$$\delta_{\vec{r}}(x_i - y_i) = \begin{cases} \frac{|x_i - y_i|}{y_i}, & y_i > \epsilon \\ \frac{|x_i - y_i|}{\epsilon}, & y_i \leqslant \epsilon \end{cases} \tag{4.29}$$

This can be interpreted as a normalization against the reference vector $\vec{r} = \vec{y}$ where also a small quantity ϵ is introduced to prevent numerical problems.

Note that this is not a distance as it does not fulfil the triangular inequality; this because of the asymmetry of the distance ($d^{\vec{r}}_{L_2}(\vec{x},\vec{y}) \neq d^{\vec{r}}_{L_2}(\vec{y},\vec{x})$) and because of the abrupt change around ϵ. Nevertheless it turns out to be a useful similarity measure.

5 APPLYING MACHINE LEARNING

Up to now the necessary theory to treat and filter images by means of Max-Trees, which is, hierarchical trees of connected components (chapter 3), has been presented. Applying the (image) moments theory (chapter 4) to the peak components let arise a powerful tool to evaluate this image decomposition into peak components on a quantitative basis.

The prototype filtering application coming forth from previous work [Urbach and Wilkinson, 2001, Urbach et al., 2004, 2005, 2007] applies the following idea: a morphological filter (or operator) is defined based on so-called *attributes* of the elements of the Max-Tree, which are vectors of values of (some kind of) image moment; this vector is then considered interchangeably as an attributes vector and a features vector. The filter accepts or rejects the peak component by analyzing only those attributes; this idea corresponds to the direct filtering rule described in chapter 3.

This filter takes a *reference attribute vector*, a *minimal area*, and a (*dis-*)*similarity threshold* as parameters; see chapter 1 for the details. The applied accepting criterion T is then

$$\| \mathrm{MN} \| > a_{t_h} \quad \wedge \quad \left\| \vec{\mathrm{MN}}_{m_t,o} - \vec{\mathrm{P}}_{m_t,o} \right\|_{L_i} < d_{t_h} \tag{5.1}$$

where

$\| \mathrm{MN} \|$ is the *size* (*area*) of the selected peak component,

$\vec{\mathrm{MN}}_{m_t,o}$ is the *feature vector* describing the peak component,

$\vec{\mathrm{P}}_{m_t,o}$ is the *reference feature vector* associated with the selected *reference* traffic sign,

$\| \|_{L_i}$ is the norm induced by a Minkowsky *metric*,

a_{th}, d_{th} are the *threshold parameters* for the minimal *area* and maximal (*dis*)-*similarity*,

m_t, o are the image moments *type* and the relative *order*.

In our problem the reference vectors are computed upon binary images of the traffic sign; for any selected traffic sign there is one such prepared image. The attributes vectors are computed for the components of the Max-Tree while constructing it.

This criterion compares the components of the Max-Tree one by one with a reference image by means of the attribute vector; in this way the binary reference images can be compared with gray scale connected components. Here we explicitly talk about (dis-)similarity instead of distance because of the theoretical objections on $\| \|^{\vec{r}}_{L_2}$ (see section 4.1); a metric is not required as a semi-metric (where the triangular inequality is not satisfied) is sufficient.

As explained in chapter 3, the idea behind the Max-Tree is a decomposition in flat zones of increasing gray level. This would be a heavy restriction as it is highly unlikely that any traffic sign can be captured by a single flat zone. Shadings caused by non-homogeneous light sources, irregularities and bumps on the traffic sign are quite normal and these flat zones are likely to capture just thin lines on the boundary of the object.

To deal with this the decomposition is „relaxed“ (or actually extended) by considering the peak components associated to the nodes instead of the flat zones. Those peak components are hills or mountains in the three-dimensional intensity image which are more likely to resemble the targeted traffic signs.

It is also likely that several peak components will contain extraneous pixels or regions of pixels. This phenomenon is expected to be compensated by the natural nesting of the peak components (to be more precise, the expectation is that among the nested peak components, there is at least one which resembles *closely enough* the traffic sign as humans recognize it).

For the traffic sign recognition, the problem now is shifted to find the appropriate (dis-)similarity measure among the ones of section 4.1, the right (dis-)similarity threshold and area threshold. Those are simple user tasks but such interventions prevents the morphological attribute filtering from being applied in an automated way and, even worse, is image dependent. For any new image the very same user-driven analysis is required.

Here the research within machine learning offers us a solution, which is precisely to let a computer program *learn* the solution to our problem. At this point it is useful to recall how the learning process by a machine (program) is defined:

> **Definition**: A computer program is said to **learn** from *experience* E with respect to some class of *tasks* T and *performance* measure P, if its performance at tasks in T, as measured by P, improves with experience E [Mitchell, 1997].

This is a very general definition, but it is rather useful in order to put the problem of the traffic sign recognition within a reasonable framework. In particular, the task T is just the recognition of a traffic sign as initially stated, but related to the morphological attribute filter, it is the *classification* of Max-Tree elements as a (known) traffic sign or just anything else (which are considered background elements).

The performance measure P is then related to the *recognition performance* of such a classifier and will depend also on its kind. Nevertheless the initial assumption is to consider the *percentage* correctly classified Max-Tree elements, the Max-Nodes, as a reasonable performance measure. Within the context of the traffic sign recognition, this performance measure implies that the Max-Tree method develops Max-Nodes corresponding with the aimed traffic signs. In the positive case, the recognition percentage is a first rough measure; in the negative one, the classifier should reject all the Max-Nodes.

As for the experience E, this is the *classification* of Max-Tree elements like it is performed by humans. This classification is simply the labelling of Max-Tree elements as one of the traffic signs or background. This is particularly convenient, as the learner can follow a *direct* learning scheme, where each learning step can be evaluated and criticised (and such critics are then the *feedback* for the learner) instead of waiting for a sequence of learner's output (sequence which then is to be graded in order to give feedback, a often difficult problem).

Having defined the context of the learning task, we have to set up a *learning system* for the traffic sign recognition problem.

Therefore, the *training experience* needs to be defined. Because this classification relies on the image moments, a natural way to set up the training experience is to put together the image moments feature vectors representing the Max-Nodes and the corresponding label. This will form $(\vec{a}, \mathsf{C_{ts}})$ pairs, where $\vec{a}$ is a feature vector of image moments and $\mathsf{C_{ts}}$ is the traffic sign class the Max-Node belongs to. Such pairs will form the *feature vectors data set*, which is the first main goal of this research. Sets of labeled patterns are very usual in machine learning.

6 THE LEARNING ALGORITHMS

Now the learning task and its context are clear; the learners are now to be introduced. For this project the application of the following learning algorithms is studied: kNN (k Nearest Neighbours), CCM (Class Conditional Means), LVQ1 (Learning Vector Quantization). See sections 6.1,6.2 and 6.3 for a short introduction.

These techniques derive from Statistical Pattern Recognition which tries to pose the problem in probabilistic terms by quantifying the trade-offs between various classification decisions using probability and decision costs.

In the following notions and the notation from [Duda et al., 2000] are used. The true *state of nature* is denoted by ω, with ω equal to some category ω_i denoted by the index i. As the state of nature is unpredictable, ω can be seen as a probabilistic variable. Then the *a priori* probability (or *prior*) of some pattern to belong to a category ω_i is denoted by $P(\omega_i)$; of course all the $P(\omega_i)$ sum up to one.

If x is a random variable representing an unseen item, $p(x)$ is the probability density function for x and $p(x|\omega_i)$ is the probability density function for x given that its state of nature ω equals ω_i: the *class-conditional* probability. This is also called the *likelihood* of ω_i with respect to x.

The *a posteriori* probability (or posterior) $P(\omega_i|x)$ is then the probability of ω being ω_i given x: $P(\omega_i|x) = P(x|\omega_i) \cdot P(x) / \sum_i P(x|\omega_i) \cdot P(x)$.

In general this probabilistic information is used to classify a point x in a meaningful way to a class ω_i. A very general pattern is to look at estimated posteriors like in the following *decision rule*: decide ω_1 if $P(\omega_1) > P(\omega_2)$, otherwise decide ω_2. Note that the posterior corresponds to the performance measure P mentioned in (5).

All of the kNN, CCM and LVQ1 are *data-driven* procedures, where no assumptions are made about the underlying (often parametrized) probability density function. This way these methods can deal with arbitrary probability function, including multi-modal ones. Moreover, all of these methods are multiclass classifiers.

6.1 K NEAREST NEIGHBOURS

For the k Nearest Neighbours rule (kNN) a complete theoretical foundation is stated in [Duda et al., 2000].

The kNN procedure does not estimate the underlying probability distribution $p(x)$ but bypasses this estimation and goes directly to a decision function.

If $\mathcal{D}^n = \{x_1, \ldots, x_n\}$ is a set of n labeled prototypes (examples), and x a test point, this rule directly assigns x to the most frequently represented class among the nearest k prototypes; this is like taking a vote among the k nearest neighbours. See algorithm 6.1 on the next page for the kNN procedure in pseudo-code.

This can be seen as a tentative of estimating the a posteriori probability $P(\omega_i|x)$ from known samples. To one side one would have as many votes as possible, to the other side one would have as many samples as possible. This situation leads to compromises where k is small with respect to n. Note that for certain k, a growing number of samples n makes all the k nearest neighbors of x converge to x itself. At the same time, the larger the k, the higher is the probability for $\omega = \omega_i$ to be selected.

Algorithm 6.1 K NEAREST NEIGHBORS RULE

Learning procedure:

```
1 Store all the sampled examples with the class label
```

Classification rule:

$$c(x) := c(w_i) \,|\, w_i = \text{argmax}_{\#} \left(\text{argmin}^{k}_{w_j} d(x, w_j) \right)$$

Classification procedure:

```
1 Compute Euclidean distances from target pattern to
      sampled examples
2 Order examples by distance
3 Choose k nearest examples
4 Assign target pattern to the class with most 'votes'
```

Algorithm 6.2 CLASS-CONDITIONAL MEANS RULE

Learning procedure:

```
1 For each pattern class
2   Compute the mean example pattern
3 end
4 Store the k mean example patterns (class-conditional
      means)
```

Classification rule:

$$c(x) := c(w_i) \,|\, w_i = \text{argmin}_{w_j} d(x, w_j)$$

Classification procedure:

```
1 Compute Euclidean distances from target pattern to the
      class-conditional means
2 Assign target pattern to the class with the smallest
      distance
```

Theoretically for any k a *Voronoi tessellation* is induced where each polygon represents the classification boundaries for a certain class.

The computational complexity is relevant just for the classification because the explicit training is bypassed. This complexity is $\mathcal{O}(knm)$ for k nearest neighbours, n example patterns and m patterns to classify. Note that the examples storage is $\mathcal{O}(md)$ for n example patterns and d dimensions.

6.2 CLASS-CONDITIONAL MEANS

Instead of keeping all the examples like kNN does, the Class-Conditional Means (CCM) procedure estimates the underlying class-conditional probability distributions $p(x|\omega_i)$ by assuming that each distribution can be represented by a single item, the *mean* of the labeled examples.

The voting of kNN is then substituted by finding the nearest prototype to the test point x; the decision rule is then simply to assign x to the same class ω_i of the nearest prototype.

See algorithm 6.2.

Algorithm 6.3 LEARNING VECTOR QUANTIZATION RULE

Learning procedure:

```
Initialize kn prototypes
For each epoch
  For each example pattern
    Compute Euclidean distances from example pattern
        to prototypes
    Update the nearest prototype with the LVQ1 rule
  end
end
Store the kn prototypes
```

Update rule:

$$w_i(t+1) \quad := \quad w_i(t) + \eta(t) \cdot [2 \cdot \delta(c(\xi), c(w_i)) - 1] \cdot (\xi - w_i)$$

Classification rule:

$$c(x) := c(w_i) \,|\, w_i = \text{argmin}_{w_j} d(x, w_j)$$

Classification procedure:

```
Compute Euclidean distances from target pattern to the
     kn prototypes
Assign target pattern to the class with the smallest
    distance
```

Theoretically a *Voronoi tessellation* is induced (analogous to kNN but with a simpler structure) where each polygon represents the classification boundaries for a certain class, one for each class mean.

The computational complexity for the training is $\mathcal{O}(m)$ for m patterns to classify. The same $\mathcal{O}(m)$ complexity for m patterns to classify applies also to the classification time.

Note that the idea of representing a class by a single point is quite simplistic and is known to perform poorly in some cases, e.g. concave class distributions like ring-shaped or banana-shaped classes.

6.3 LEARNING VECTOR QUANTIZATION

Learning Vector Quantization (LVQ) has been introduced by Kohonen [Kohonen, 1997]; it has been the subject of quite some recent research and the following is also based on [Biehl et al., 2009]. Similar to kNN and CCM, LVQ is a prototype-based classifier based on metric comparisons (mostly the Euclidean distance (4.26)).

Different from kNN, the LVQ procedure estimates the underlying class-conditional probability distributions. To this end, LVQ learns by means of codebook vectors, which is a set of prototypes. These vectors are used to infer the class of unseen data by looking at the closest prototype. CCM uses a single prototype per class to perform the same inference.

Assume here $\mathcal{D}^n = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^n \times \{1, \ldots, C\}$ a set of labelled data points, a LVQ classifier is defined by the set of k prototypes $W^k = \{w_1, \ldots, w_k\} \in \mathbb{R}^n \times \{1, \ldots, C\}$. Similarly to CCM, for a given test point x first is the nearest prototype found; the classification rule is then again simply to assign x to the same class ω_i of the nearest prototype (the winner takes all):

$$c(x) := c(w_i) \quad | \quad w_i = \text{argmin}_{w_j} d(x, w_j) \tag{6.1}$$

where $d(x, w_j)$ is the Euclidean distance (4.26). The initialization is not defined within LVQ and several possibilities are present. Here we consider the following initializations: random, class-conditional means and k-means initializations.

The *random* initialization consists in randomly taking prototype vectors from the data points of a given class of the training set. The CCM initializations consist in taking first the class-conditional means and using them as unique prototype for each class. Finally, the k-*means* initialization consists in taking the output vectors of the k-means clustering for each class and use them as initial codebook vectors.

The training consists in presenting the training set of example points in a randomly permuted order for a number of times; a presentation of the examples set is called a *learning epoch*. Each time a random example is presented to the LVQ learner the following learning rule (the so called *winner takes all* rule) is applied:

$$w_i(t+1) \quad := \quad w_i(t) + \eta(t) \cdot [2 \cdot \delta(c(\xi), c(w_i)) - 1] \cdot (\xi - w_i(t)) \tag{6.2}$$

where ξ is the random example in the LVQ notation, $\eta(t)$ is the learning rate at time t, $\delta(c_i, c_j)$ is the Kronecker delta function which assumes the value 1 if the two labels c_i and c_j are equal and 0 otherwise, $c(\xi)$ and $c(w_i)$ are the class labels of the example ξ and the prototype w_i; finally $w_i(t)$ is the closest prototype to the data point ξ at time t for distance measure $d()$:

$$w_i(t) \quad | \quad w_i(t) = \text{argmin}_{w_j} d(\xi, w_j(t)), \quad \xi \in \mathcal{D}^n, w_j(t) \in \mathcal{W}^k(t) \tag{6.3}$$

The learning rate η might be *fixed*, or an *annealing schedule* (a sequence of values decreasing to 0) might be applied. The choice usually depends on the learning problem and/or eventual overfitting, as well as the choice for the number of learning epochs.

The appealing features of LVQ are it's conceptual simplicity which makes the implementation easy and intuitive. Moreover using the same space of the input data points for the codebook vectors makes the interpretation of these prototypes intuitive and useful for the domain experts.

See algorithm 6.3 on the previous page for the LVQ training and classification procedure.

Analogous to kNN and CCM, a *Voronoi tessellation* is induced by the LVQ prototypes where each polygon represents the classification boundaries for a certain class, one for each prototype.

The computational complexity for the training is $\mathcal{O}(cne)$ for c classes, n training patterns and e training epochs. The computational complexity for the classifier is $\mathcal{O}(cm)$ for c classes and m patterns.

6.4 THE ROC CURVE

Having introduced different data classifiers after the (dis-)similarity filters, a way is needed to visualize, organize and compare the performances. To this end the *Receiver Operator Characteristics* (ROC) graph is a simple and intuitive tool which is introduced here; for an extended introduction see Fawcett [2004, 2006]

By offering a data instance to a classifier, there are four outcomes (as shown in the *confusion matrix* of table 1):

		True Class	
		Traffic Sign	Background
Hypothesized Class	Traffic Sign	**T**rue **P**ositive	**F**alse **P**ositive
	Background	**F**alse **N**egative	**T**rue **N**egative

The decision entries on the major diagonal are the correct decisions, while the entries off the major diagonal are the errors.

Table 1: CLASSIFICATION CONFUSION MATRIX

True Positive: the instance is a traffic sign and is recognized as traffic sign;

True Negative: the instance is a background and is recognized as background;

False Positive: the instance is a background and is recognized as traffic sign;

False Negative: the instance is a traffic sign and is recognized as background.

Based on these quantities, the following are defined:

True Positive Rate: (TP rate, also *hit rate*) the instances fraction of correctly classified traffic signs divided by the total number of traffic signs instances;

False Positive Rate: (FP rate, also *false alarm rate*) the instances fraction of incorrectly classified background divided by the total number of background instances;

Now it is possible to visualize the performance of different classifiers in the so called ROC space. The ROC space is drawn in the Cartesian space where on the abscissa axis the FP rate is plotted, while on the ordinate axis the TP rate is plotted. Both the TP rate and the FP rate belong to the $[0 \dots 1]$ domain, therefore the ROC space is defined by the unit square $[0 \dots 1] \times [0 \dots 1]$. The point $(0, 1)$ represents the perfect classification, the diagonal line from the origin to $(1, 1)$ represent the random guessing classification; any classifier in the upper left triangle performs better than random guessing, and the closer to $(0, 1)$ the better is the classifier (while barely considering TP and FP ratios).

For classifiers which give a probability or core for an instance to belong to a class it is possible to draw a curve in the ROC space, starting from $(0, 0)$ where no data instances is accepted, and ending in $(1, 1)$ where all the data instances are accepted (and so all the true positives as well as the false positives). This curve is traced by shifting a *score threshold* ideally from $-\infty$ to ∞. In the context of traffic sign recognition there is a discrete amount of data, therefore instead of a continuous ROC plot a step function is traced. This step function approaches the continuous curve as the number of data instances grows to infinity. An example of such curve is shown in figure 5 on the following page.

If several classifiers are compared it is convenient to reduce the ROC curve to a single scalar representing the classifier's performance. A common reduction way is to look at the *area under* the ROC curve (AUC). The ROC square is the unity box and therefore the AUC is always in the interval $[0 \dots 1]$; for comparison it is useful to notice that the AUC for the random guessing is equal to the area of the lower right triangle of the ROC space, which has value 0.5.

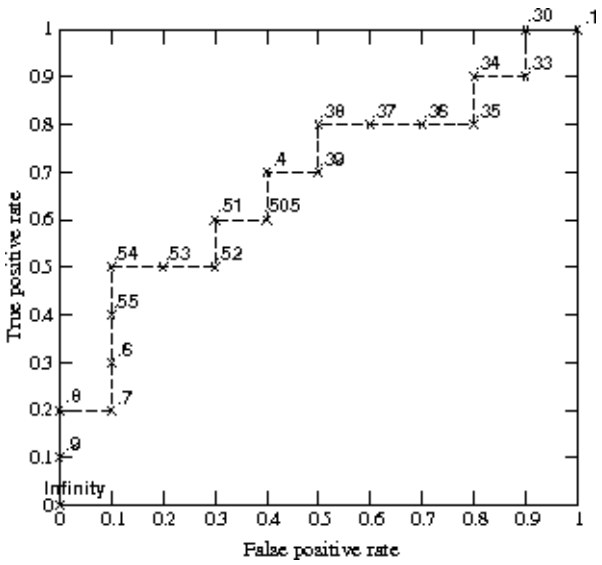


Figure 5: Discrete ROC curve example. Figure taken from [Fawcett, 2004].

The AUC is statistically important too:

> "the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly negative instance." [Fawcett, 2006]

This makes the AUC a statistically founded measure of the discriminative power of the classifier, and therefore the AUC is a very good performance and generalization power indicator.

Part II

FEATURE EXTRACTION FROM IMAGES

Now that the involved theory is set out, the first goal of the automatic traffic sign recognition project is to be tackled: starting from a set of images a data set of labeled shape features is constructed. To this end the application developed in [Bracci, 2008] is used; during the labeling of the images also modifications and extensions were added to the application. As this project naturally continues previous work, chapters 7 and 8 are based on parts of [Bracci, 2008].

In this part first the initial image data set is described in chapter 7, the abstract structure of the labeling system and the used labeling application are explained in chapter 8, the data labeling process is dealt with in chapter 9 and eventually the produced features data set is set out in chapter 10.

7 STARTING DATA

7.1 IMAGE SET

The starting point for this project is a collection of road pictures taken with a (COTS[1]) digital camera. These picture were collected by E.Urbach for the purpose of his research Urbach et al. [2007, 2005].

The picture collection consists of 166 gray scale images of 3Mpixels resolution (2160×1440 pixels with intensity depth 255) spread across 34 directories. The pictures are stored in files of the raw Portable Graymap Pictures (PGM[2]) format. Such image depth requires one byte of memory per pixel and therefore its size in bytes equals the number of pixels of the image. The PGM storage format is a choice made during previous research by Erik Urbach in Urbach et al. [2007, 2005], but any (preferably loss-less) image format could be used too.

An alternative images database is available at the Intelligent Systems group; this is made by 48 color images in Portable Network Graphics (PNG[3]) file format of resolution 360×270 pixels[4].

A careful look reveals that some of the data set images are copies of each other; the amount of unique images is 93. See figure 6 for examples of the used pictures of urban roads.

The aim of this project is, as introduced in chapter 1, to achieve traffic sign detection within urban road pictures for a selected set of four particular target traffic signs, using real world images. For this reason every image file may contain any traffic sign with any multiplicity. Some of them contain one or more target traffic signs, others do not contain any target traffic sign, a number of them contains non-target traffic signs. See table 2 for an overview of the number of urban road images available for each traffic sign.

This image set contains pictures taken at different day times with varying light conditions, varying quality and context. The traffic signs do not follow any particular placing (e.g. the traffic signs occur always at the left or right side of the image) so no spatial information can be easily neither meaningfully collected nor a priori stated in order to ease the recognition task.

Moreover, the traffic signs themselves do not have a particular spatial disposition either (e.g. they are all photographed while staying in front of it) and therefore they may show a broad range of rotations and perspective deformations. This is consistent with the aim of putting together images representing real life situations where ideally the traffic signs are to be targeted.

See figure 7 for a number of examples of urban road images with problematic traffic signs.

Ideally, a larger and more complete set of picture could be considered as the data is relatively easy to gather. Enlarging the image data set might be worth considering in follow-up projects and research.

1 Common off the shelf
2 For the PGM file format specification see http://netpbm.sourceforge.net/doc/pgm.html
3 For the PNG file format specification see http://www.libpng.org/pub/png/
4 This color images database is available at http://www.cs.rug.nl/~imaging/databases/traffic_sign.html

(a) Image with the *drive straight ahead* traffic sign

(b) Image with the *pedestrian path* traffic sign

(c) Image with the *bicycle path* traffic sign

(d) Image with the *pedestrian crossing* traffic sign

(e) Image without any of the *target* traffic signs

Black and white circles highlight the traffic signs.

Figure 6: Typical input images.

(a) Image with an atypical *bicycle path* traffic sign

(b) Image with an heavily bent *pedestrian path* traffic sign

(c) Image with small atypical *drive ahead* traffic sign in the background

(d) Image with a blurred *bicycle path* traffic sign

(e) Several tiny images on the background. The foreground traffic sign is not considered a *drive ahead* traffic sign

(f) Image with an heavily bend *bicycle path* traffic sign. The other two traffic signs are not considered as *drive ahead* traffic signs

Black and white circles highlight the traffic signs.

Figure 7: Problematic input images.

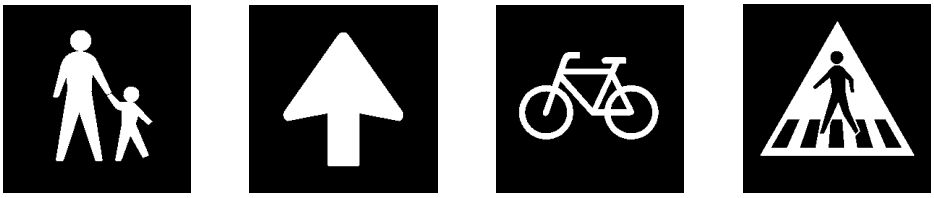
Traffic sign type	Pedestrian path	Pedestrian crossing	Drive straight ahead	Bicycle path	None	Total
nr. pictures	5	3	32	31	34	93

Note that some images contain several and/or different traffic signs.

Table 2: Distribution of images with traffic signs.

7.2 TARGET IMAGES

In this project only a small selection of different traffic signs are chosen as recognition targets. This is satisfactory for the sake of this project, which is exploratory; within further research and/or future applications it would be possible to set up a broader targets set which covers all the useful traffic signs.



(a) Obligatory pedestrian path.

(b) Drive straight ahead.

(c) Obligatory bicycle path.

(d) Pedestrian crossing.

Notice that all these reference images are *binary* and consist of exactly a *single* connected component. This holds also for sub-figure 8a.

Figure 8: The target traffic signs.

The system is provided with the following four different traffic signs shown in figure 8:

- (Obligatory) Pedestrian path
- Drive straight ahead
- (Obligatory) Bicycle path
- Pedestrian crossing

Those images are needed for the purpose of reference pictures, as is explained in the next chapter (8).

The idea is that each of these images will serve as the *ideal representation* of a particular traffic sign. To this end these images are *binary* images (such that the contrast is maximal) and are made of just a *single* connected component (to simplify the information extraction).

This is the case also for the pedestrian path sign, where the adult and the child seem separated. Instead the two stylized human figures are connected by a few pixels in order to have a single connected component here too; this way it is possible to treat all the target signs the same. Of course this approach is not very general and is not adequate for more complex signs which cannot be approximated by a single connected component.

All the four reference images are roughly of the same size (the pedestrian path picture has size 279×280 pixels; the other three pictures are of size 260×260 pixels) although the relevant part is the singleton foreground connected component; the relative connected components are of comparable areas as well (respectively 9592, 13761, 7065 and 16077 pixels). This way the set of reference images is reasonably assumed to provide pictures of comparable quality.

8 LABELING SYSTEM

The set of digital images and targets described in chapter 7 is just the starting point of this project. For the purpose of automatic traffic sign detection such data is unsuitable to be used directly.

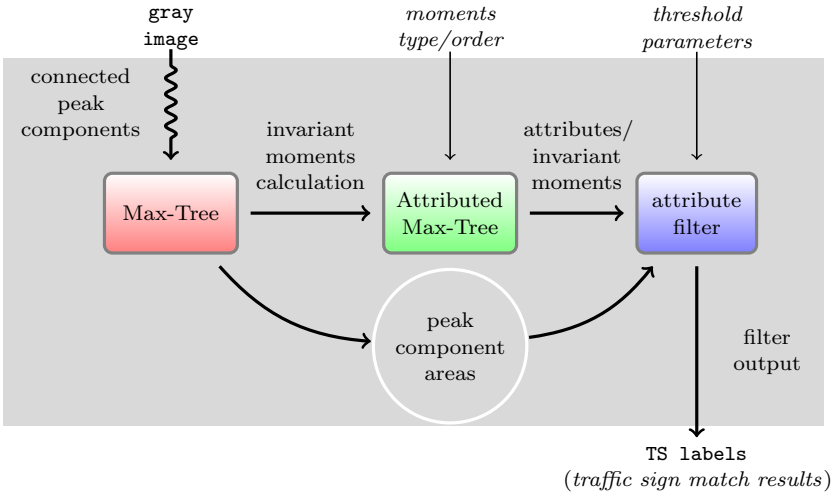
For instance, in [Biehl et al., 2008] a "direct" approach is used; this was achieved by an image pre-processing step where regions of interest (bounding boxes) containing the sperm cells (aligned to the bounding boxes) are found, and by considering these regions as a vectors of pixels. This approach is unfit for this task because the pre-processing is highly non-trivial and the pixels vector length would project the problem to such high dimensionality that the learnability becomes at stake.

In this project the choice is instead to decompose the image in small meaningful parts and to perform the recognition on these image parts afterwards; the aim is to realize this by using ML classifiers. The chosen learning techniques are of the supervised type (see chapter 5) and are based on a training phase where labeled examples (feature vectors) are offered to the learner.

For this reason a fundmental step in this project is the generation of suitable training data sets; because of the number of images and the number of image parts present in the image decomposition, a tool is needed to ease and partially automate the image parts labeling. First the abstract information process is set up to label the data in the next section; the implemented application and its usage is treated after in section 8.2.

8.1 LABELING SYSTEM STRUCTURE

The problem of generating labeled data sets based on image moments is dealt with in Bracci [2008]; there a system is built for the visualization and labelling purpose. See the research internship report [Bracci, 2008] for an extensive discussion from



The shaded box contains the processing stages of the labeling system; on top of it the inputs are shown, below the outputs are shown. Thin arrows represent user driven parameter input, the snake arrow represents the recursive flood-fill procedure to decompose the image into peak components.

Figure 9: THE IDEAL LABELLING SYSTEM.

the involved theory (the Max-Tree, the Invariant Moments and the geometric, central and normalized central moments) to the software engineering problems.

The ideas behind the labelling system are explained in the following, where the concepts explained in chapter 2, 3 and 4 are put together in a seamless process.

The main system input is a gray scale image as described in chapter 7, which is decomposed in connected peak components organized in a tree by using Salembier's recursive flood-fill procedure[Salembier et al., 1998] (see chapter 3 for the relative theory).

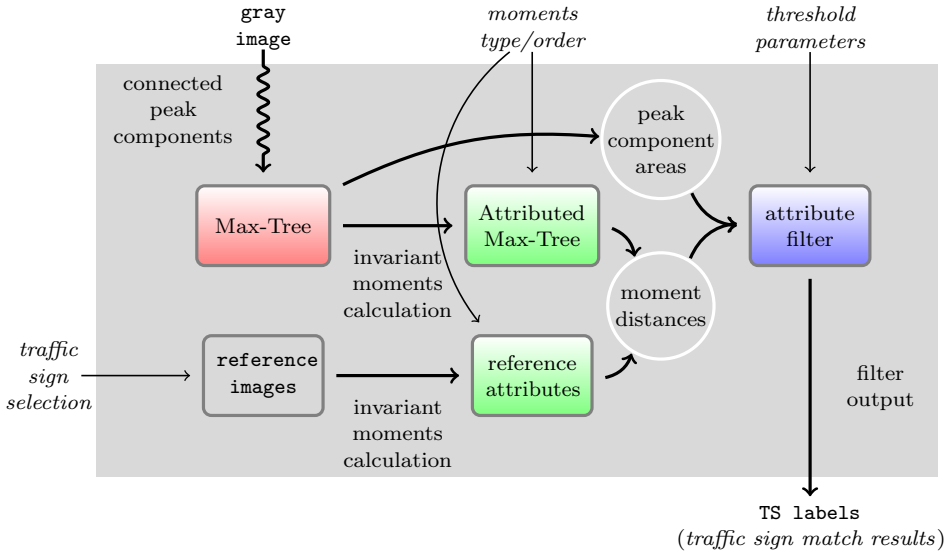
Such a tree now describes the image, the *Max-Tree* (MT); here the invariant moments (see chapter 4) are computed for all the connected peak components held in the Max-Tree according the user's selected type and order and then the invariant moments are added to the Max-Tree itself.

The invariant moments play the role of *attributes* for the Max-Tree, which now becomes an *Attributed Max-Tree* (AMT) of the image. Such attributes are the feature vectors which are later constituting, together with the class labels, the training and testing data sets for the learning algorithms, treated in chapter 5.

The ideal last stage is a *filter* for the attributed Max-Tree; this filter should select connected components based on the attributes. Also the size of the connected components matters and for this reason this filter is also provided with area information (which is already available at the previous stage, the plain Max-Tree).

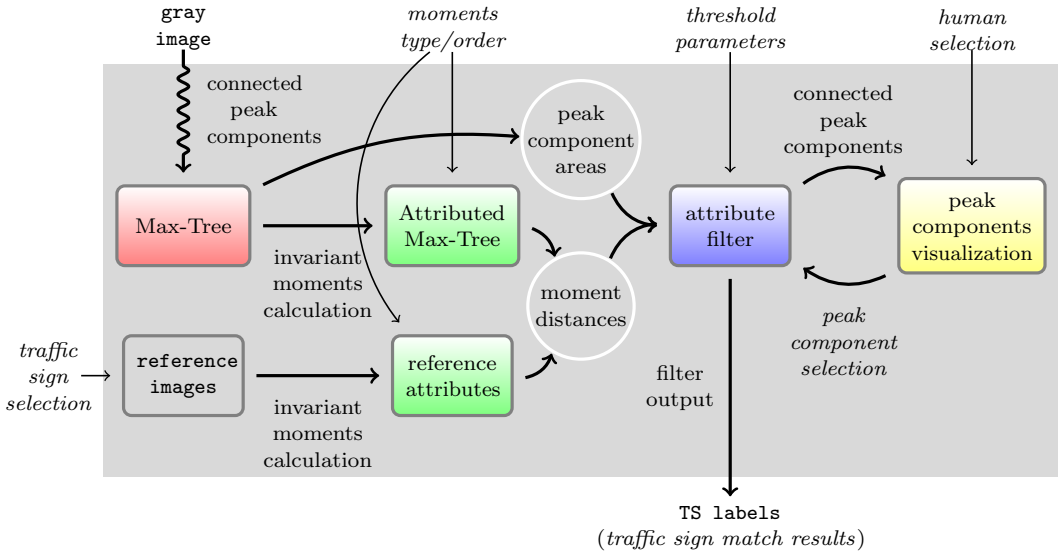
A filter ideally removes all the connected components which do not belong to a selected target traffic sign. By putting together several filters, one for each selected target, it is possible to set up a filter bank providing the appropriate label to each connected component. Such a label indicates either which traffic sign the connected component belongs to or if it is just a background entity.

The block diagram in figure 9 on the preceding page shows the way the image information flow is structured into different stages in this system; this system is indicated as the *ideal* labeling system.



The shaded box contains the processing stages of the labeling system; on top and to the left of it the inputs are shown, below the outputs are shown. Thin arrows represent user driven parameter input, the snake arrow represents the recursive flood-fill procedure to decompose the image into peak components.
Note the addition with respect to the previous system in figure 9 of the lower branch in the system, from the traffic sign selection up to the moment distances.

Figure 10: The reference-based labelling system



The shaded box contains the processing stages of the labeling system; on top and to the left of it the inputs are shown, below the outputs are shown. Thin arrows represent user driven parameter input, the snake arrow represents the recursive flood-fill procedure to decompose the image into peak components.
Note the addition with respect to the previous system in figure 10 of the peak component visualization and selection to the right.

Figure 11: THE SUPERVISED LABELLING SYSTEM

The attribute filter at the core of this system is ideal in the sense that it actually entirely *solves* the whole recognition problem; this is not the case as shown in [Urbach et al., 2007].

In that work a certain level of recognition is achieved by using ideal *reference images* for the target traffic signs together with the invariant moments and similarity computation. The reference images here are the ones described in chapter 7.

Here the invariant moments are used as reference attributes and provided to the filter. The filter considers how (*dis-*)*similar* the Attributed Max-Tree elements are with respect to these reference attributes according to a (user selected) *metric* (L_1, L_2, L_∞ or $L_2^{\vec{r}}$). The user provides a (dis-)similarity *threshold* as well, which reduces the filtering to a simple distance criterion check. Also for this solution the area information is provided to the filter. This system is indicated as *reference-based* labeling system and is represented in figure 10 on the preceding page.

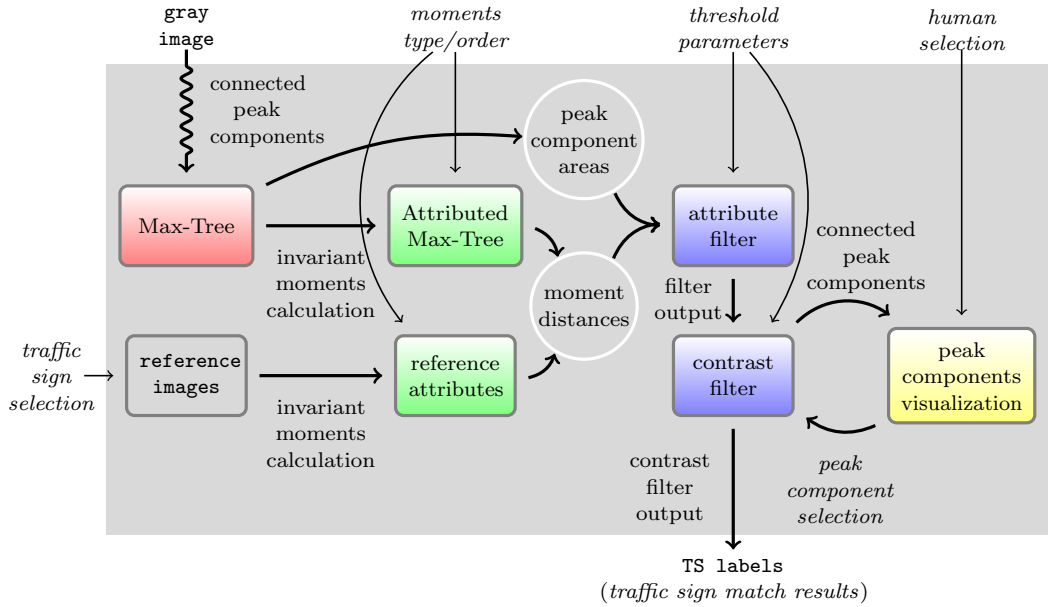
This references-based filter acts basically as a pre-processing step. The peak components corresponding to traffic signs are expected to be recognized with appropriate user parameters, but not all of them. Most important, nothing is said yet about discarding all the background peak components.

Therefore a last stage is added to the system, where the filter's output is visualized and eventually corrected by the user; this stage is walked through as long as the output is not satisfactory.

Eventually, when all the peak components are correctly tagged according to the human supervisor, the system will store the traffic sign labels in a new file[1]. This system is indicated as *supervised* labeling system and its structure is shown in figure 11.

As it is discussed in the next section (8.2), another pre-processing filter is needed in order to avoid pollution of the visualization by hundreds or thousands of

1 This labels file has the same name as the image file, followed by the extension `.tag`.



The shaded box contains the processing stages of the labeling system; on top and to the left of it the inputs are shown, below the outputs are shown. Thin arrows represent user driven parameter input, the snake arrow represents the recursive flood-fill procedure to decompose the image into peak components.
Note the addition with respect to the previous system in figure 11 of the contrast filter right below the distance and area filter.

Figure 12: The contrast-based supervised labeling system.

background elements. In such a situation too many entities need to be discarded by the user and the burden of this task makes the whole labeling unfeasible.

In order to cope with this, a contrast filter is introduced; this filter is applied in the processing pipeline right after the reference filter and it's output still is offered for visualization/supervision. This system is indicated as *contrast-based supervised* labeling system and its structure is shown in figure 12.

An overview of all the system's parameters which are eventually driven by the user is to see on table 3.

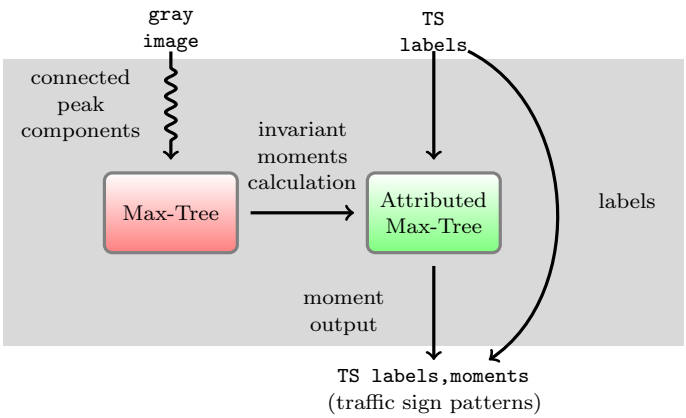
Parameter	*Values*
minimum area	$1 \ldots \texttt{size}(\texttt{image})$
maximum area	$1 \ldots \texttt{size}(\texttt{image})$
moment type	geometric, central, normalized central, Hu, Flusser and Suk Affine, Flusser and Suk Complex
moment order	$1 \ldots 10$
maximum distance	$0 \ldots 10$
distance type	L_1(Manhattan) distance, L_2(Euclidean) distance, L_∞ distance
minimal contrast	$1 \ldots \texttt{max\_gray}(\texttt{image})$

Note that for the Hu moments just the *third* and *fourth* orders are defined, while for the Flusser and Suk Affine moments only the *first*, *fourth* and the *sixth* order are defined.

Table 3: The system's parameters

Ideally one would write a data set of labeled moments after the user supervision is completed. In case of labelling problems and/or mistakes the whole user-driven labelling task has to be repeated, without reusing (parts of) a previous trial. Decoupling the labels from the moment data simplifies reloading and correcting the labels and so the moments data.

The cost for this decoupling is to set up a second small system to generate the final data sets from the images and the labels; this system is indicated as *data set production* system and is set out in figure 13. This task involves loading and processing the images which previously undewent the labeling procedure again, decompose them again in peak components, setting up the attributed Max-Tree again, add the precedently stored labels and finally store the labeled Max-Nodes. While this involves some computing time, it is easy to automate[2] and therefore this two-stage data set production process (first the labeling step, then the data set storing step) looks attractive for this project.



The shaded box contains the processing stages of the labeling system; on top of it the inputs are shown, below the outputs are shown. The snake arrow represents the recursive flood-fill procedure to decompose the image into peak components. Note the absence of any filter and user input: the label files totally determine the traffic sign class and the output is generated straight away.

Figure 13: THE DATA SET PRODUCTION SYSTEM

2 In UNIX environments shell scripts are very practical for this task.

8.2 LABELING APPLICATION

In this section the labeling application is discussed. As mentioned, for the Attributes Max-Tree filtering a prototype application was realized in [Bracci, 2008]. This program performs the Max-Tree computation for an image file offered as a command line parameter and is implemented in the c++ programming language using the fltk (Fast Light Tool-Kit[3]) GUI framework.

Its structure is rather simple: there is a *control* window, an *image* window, a *Max-Tree* window and an *about* window. The program is named `shapedemo-dynamic` (see figure 18 on page 45 for an overall impression of the application).

Note the superposed traffic signs; this is a typical visualization of the connected components accepted by the filters. Here the real traffic sign is recognized together with at elast one background component.

Figure 14: The image window

IMAGE WINDOW: this application window visualizes the loaded image file; whenever filtering is performed, icons of the reference images are applied to the loaded image, centered on the matching peak component and scaled horizontally and vertically proportionally to the matching component's area. See figure 14.

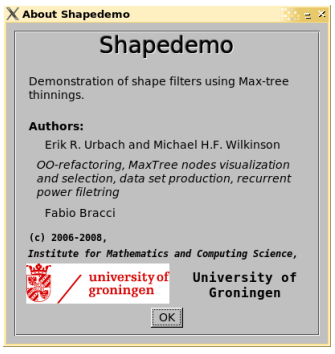
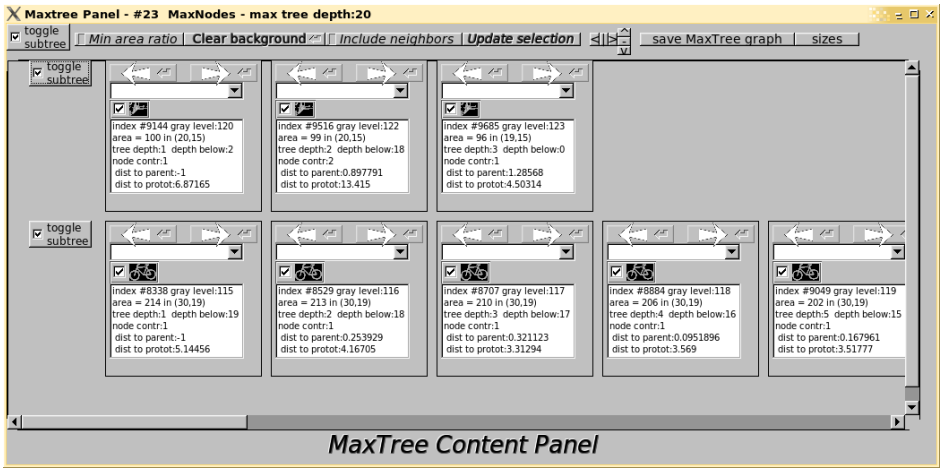


Figure 15: The about window

ABOUT WINDOW: this window displays some usage, version and copyright information. See figure 15.

3 see http://www.fltk.org/



Note the link with the previous figure (16): here al the accepted connected components are shown in a tree-fashion which corresponds with the Max-Tree component nesting.

Figure 16: The Max-Tree window

MAX-TREE WINDOW: this window visualizes the peak components stored the Max-Tree. Moreover, the labels assigned by the filters can be inspected and corrected through drop-down menus placed next to each visualized peak component.
Also several buttons are provided for quick scrolling across large Max-Tree sections, buttons to toggle the label for a whole sub-tree, buttons to refresh/update the visualization, a button to save a screenshot. See figure 16.

CONTROL WINDOW: this window controls the system and collects the parameters input. It allows to select the target traffic sign, the moment type and order, the (dis-)similarity measure and threshold, the area threshold and the minimum contrast for the connected components (see the parameters overview in table 3 on page 40).
There are also three buttons: the `Save tags` button which saves the tags so far present in the system; the `Show Max-Nodes` button which toggles the Max-Tree window visible and invisible; the `About` buttons which toggles the about window and the quit button which terminates the application. See figure 17 on the following page.

While labeling the set of images, a number of modifications has been made to the application developed in [Bracci, 2008] to overcome problems arisen on the way.

In order to perform a fast flood-fill, the application uses a memory allocation optimization. At the very beginning a huge array of Max-Nodes of the same size in pixels of the image is allocated, instead of allocating memory when a new component is flooded; this minimized the memory allocation overhead. Eventually just few Max-Nodes are really computed and the array is highly sparse; this turns out to leave too little memory to compute the moments.

In order to overcome this problem an array *compaction* step is introduced where a new array of the right size is allocated for the Max-Nodes; the Max-Nodes are copied to this new array and the previous sparse array is then freed. At this point the moment calculations can be performed without memory shortage problems. This step is linear in the number of flooded components.

While labeling, the number of background nodes accepted by the reference filter turns out to be enormous. Also it turns out that the traffic sign Max-Nodes pile

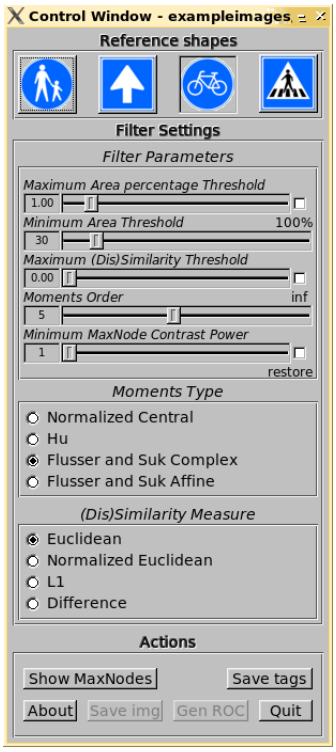


Figure 17: The control window

up stretching across a considerable interval of gray levels; this gives the inspiration for contrast-based Max-Tree filtering.

A sight at the filter output structure quickly reveals that a large part of the sub-tree is shallow; only the traffic signs and a few background elements show up in the filter visualization in the form of very deep sub-trees with few branching. Considering that any of these sub-trees stretches across a minimum gray intensity at the root Max-Node and a maximum gray intensity at the deepest leaves, the gray levels difference can be seen as the sub-tree *contrast*.

It is possible to define contrast filtering as a post-processing step of the reference filter by discarding all the elements of a sub-tree which span a low contrast from

Algorithm 8.1 CONTRAST FILTER

```
1  contrastFilter( MaxNode mn, int minContrast )
2  if (mn is accepted)
3    positiveContrast = gray contrast to parent(mn);
4  else
5    positiveContrast = 0;
6  end
7  for child in children(mn)
8    childContrast[child] = contrastFilter(child);
9  end
10 positiveContrast = positiveContrast + max(
       childContrast[]);
11 if (positiveContrast<minContrast)
12   reject(mn);
13 end
14 return positiveContrast;
```

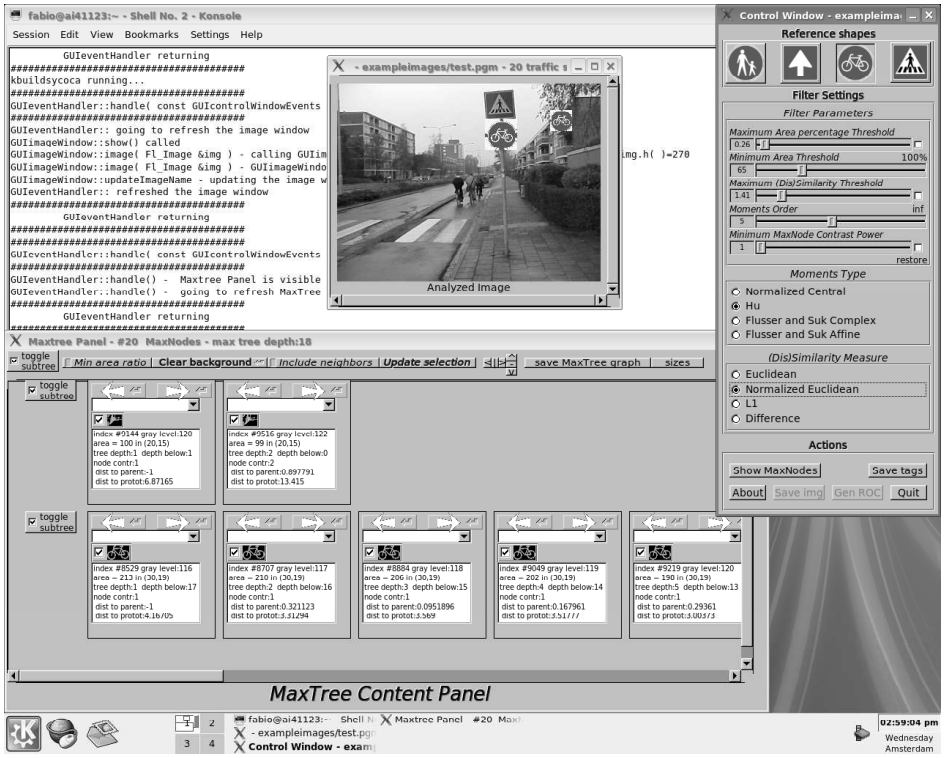


Figure 18: An impression of the labeling application

the sub-tree root; the rejection criterion is a contrast smaller than a minimal required contrast, say c_{min}. The contrast filtering is linear in the sub-tree size and it is set out in algorithm 8.1 on the preceding page.

Even if the contrast filtering suppresses quite some extraneous entities, background nodes still appear in the visualized filtered Max-Tree. This leads to a highly branched Max-Tree, where at any branching point the child node clearly resembles the parent node and the other node (or possibly several other nodes) most often is a small region detaching from the father node. such branches are in most cases obviously irrelevant for the recognition but clutter the visualized Max-Tree and make the 2D size of the visualization grow beyond reasonable usefulness for human inspection and beyond the implementation limits of the GUI framework FLTK. This is circumvented by introducing a minimal parent/child area ratio: Max-Nodes whose area is relatively small are ignored without further analysis.

In order to document graphically the Max-Tree, a procedure based on offscreen rendering was added as well; this allowed to set up a screen-shot facility for the the Max-Tree visualization.

Using the application to label the data involves a particular operating procedure, which can be described as follows:

1. Start the application with the image name to process as a command line parameter.
   The application then performs the peak components decomposition, sets up the Max-Tree and computes all the moments.

2. Choose the moment type and order.
   The application now considers only the selected moment type at the selected order.

3. Choose area and (dis-)similarity values by moving the sliders.
   In order to catch a reasonable number of true traffic signs hits while keeping

the number of false hits reasonable (i.e. lower order than 10^3), appropriate values are to be found.

4. Cycle through steps 2 and 3 until the filter output is as optimal as possible.

5. Open the Max-Tree panel and deselect all the sub-trees which are not resembling the considered traffic sign, the false hits.
   While doing this the clear background button can be used. It removes the deselected nodes from the visualization and sets their tag to the background class.

6. Relax the visualization criterion by pressing the include neighbors mode button.
   Now also children and parent Max-Nodes of those accepted so far are visualized as well.

7. Select the wrongly tagged background Max-Nodes.
   Those miss-labeled Max-nodes are in most cases the sub-tree root or leaf but also Max-Nodes inside the sub-tree might be miss-labeled.
   Again while doing this the remove background button might be useful to clean the visualization and update the tags.

8. When the visualization looks entirely correct, press the write Max-Nodes button to store the tags for the Max-nodes.
   The tags in the system are the traffic sign class labels produced from the analyzed image for all the peak components.
   The file name `<image_name>.tag` is automatically chosen.

This procedure is to be repeated for each traffic sign which is present in the input image.

Note that this procedure accomplishes the most extended labeling system among the three explained in section 8.1, the contrast-based supervised labeling system of figure 12 on page 40. By shortening this procedure it is possible to use the other systems as well, the reference-based labeling system of figure 10 on page 38 and the supervised labeling system of figure 11 on page 39. Depending on which labeling scheme the user wants to follow, appropriate steps are to be skipped.

At this point the peak component labels are set and stored; based on this the feature vectors data set has to be built by the data set production system (as shown in figure 13). The application implementing this is `shapedemo-datasetwriter`. This application takes the image name and the tags file name as command line inputs and produces seven data set files: one file for each of the four kinds of invariant moments, two files for the geometric and central moments (needed to compute the invariant moments) and an index file containing shared information. See table 4 on the next page for an overview of the data set file types.

Notice that the index file contains index information to find a peak component in the image's Max-Tree, the traffic sign class label, the area and the gray intensity. This information is grouped into a single row and all the seven files have the same number of rows. All the i-th rows ideally are coupled together and describe all the information gathered about a particular peak component, the i-th peak component of a Max-Tree. See tables 5 and 6 for details about the organization of the moment data.

The labeling process described here has to be applied to each picture of the images data set; afterward, all the tagged feature vector data sets have to be joined to form the total data set.

file name	*data set type*
`<image_name>.gem.dat`	geometric moments
`<image_name>.cem.dat`	central moments
`<image_name>.ncm.dat`	normalized central moments
`<image_name>.hu.dat`	Hu moments
`<image_name>.fsc.dat`	Flusser and Suk complex moments
`<image_name>.fsa.dat`	Flusser and Suk complex moments
`<image_name>.idx.dat`	index moments

`<image_name>` stands for the file name of the input image without the `.pgm` extension.

Table 4: Data set file extensions

```
0.2222222222, 0.04938271605, 0, 0, 0, 0, 0
0.274657888, 6.861176549e-05, 0.01187327467, 6.275003175e-05,
    -5.176897284e-08, -9.479117793e-08, 1.592642183e-08
0.3249384122, 0.02380414012, 0.006418318241, 0.0006093685397,
    -8.418582512e-07, -7.799146542e-05, 8.623174547e-07
0.2222222222, 0.04938271605, 0, 0, 0, 0, 0
0.2002906952, 0.01155292551, 0.0001573291497, 1.482481867e-05,
    -2.774690384e-10, 2.539291303e-08, -6.600071772e-10
```

Some rows taken from a data file, in this case a Hu moments data file. Note that each row represents a vector of seven values, all the values of the Hu moments at the highest order. The data files for the other moment types are structured exactly in the same way, excepted the length of the rows which directly depend of the highest order of the moment considered.

Table 5: Data set file snippet

```
4903    92    41    41    0    0    0 [Background]
4909    92    47    370   0    0    4 [Pedestrian crossing]
4927    92    65    265   0    0    3 [Obligatory bicycle
    path]
5012    93    17    6426  0    0    0 [Background]
5026    93    31    49    0    0    0 [Background]
```

Some rows taken from a index file. Note that each of these rows corresponds to a row in table 5; together these two rows completely link the moments vector to a connected peak component of a given image.

Table 6: Index file snippet

9 LABELING ANALYSIS

The first goal of this project is the production of a data set of labeled feature vectors which is needed by the selected pattern recognition techniques. As mentioned, this involves the labelling of data items (for the ML side just patterns, for the IA side the peak components of an image and their moment representation) by experts of the problem domain.

Luckily the domain experts are not too scarce: anybody having a driving license is in practice a traffic sign recognition expert, even if this is not obvious to the eye. Now the conceptual framework to label the data and an application implementing these ideas are ready; in this chapter the labeling tentatives are handled.

In the following sections the labeling results are explained. Mainly the Hu moments (4.14) up to the fourth order are chosen together with the normalized euclidean distance (4.28); this choice is made as these moments and distance qualitatively seem to show the best performance for discriminating the traffic sign peak components. In practice during the labeling other moments and distances are considered as well to see if they might perform better, but often it is not the case.

For a qualitative explanation of the experiments the image shown in figure 1 on page 3 is used as a test case.

9.1 PURE REFERENCE-BASED LABELING

The purpose here is to analyze the behavior of the reference-based labeling system described in figure 10. In particular the focus is on how far it is possible to go with the output of this labeling system towards a pure data set where all the components are correctly labeled.

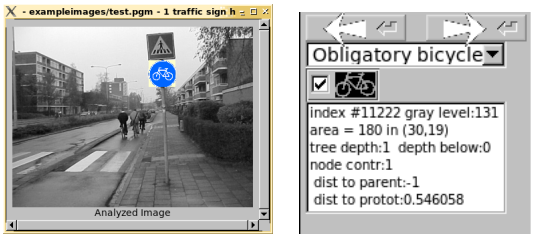
This filtering system is driven by the minimum and maximum area of the peak components and the distance of the peak component's invariant moments to the invariant moments of the reference images (also referred as the *reference moments*). This is roughly the criterion of equation (5.1), with the addition of a maximum area value $a_{t_h}^{max}$, in order to avoid large peak components matching the criterion by chance. The large false traffic sign hits clutter the visualization and hinder the labeling process as a whole, which is why it is necessary to deal with them.

The filtering criterion now is

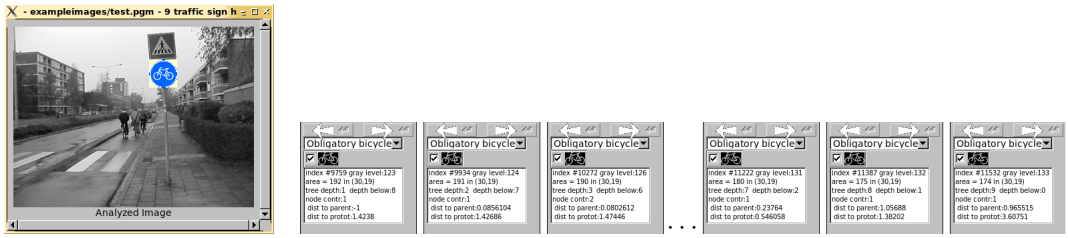
$$\| MN \| > a_{t_h}^{min} \wedge \| MN \| < a_{t_h}^{max} \quad \wedge \quad \left\| \vec{MN}_{m_t,o} - \vec{P}_{m_t,o} \right\|_2^{\vec{r}} < d_{t_h} \tag{9.1}$$

where

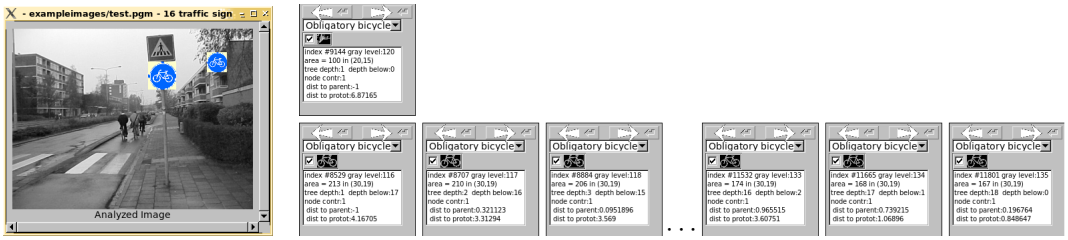
- $\| MN \|$ is the size (area) of a Max-Node (peak component),
- $a_{t_h}^{min}$, $a_{t_h}^{max}$ are the minimum and maximum area value,
- m_t, o are the moment type and its order,
- $\vec{MN}_{m_t,o}$ is the attributes vector (moments of a given type and order) describing the (shape of the) peak component
- $\vec{P}_{m_t,o}$ is the reference attributes vector (moments of a given type and order) associated with the selected reference traffic sign,
- $\|\|_2^{\vec{r}}$ is the $L_2^{\vec{r}}$ metric of equation (4.28) and



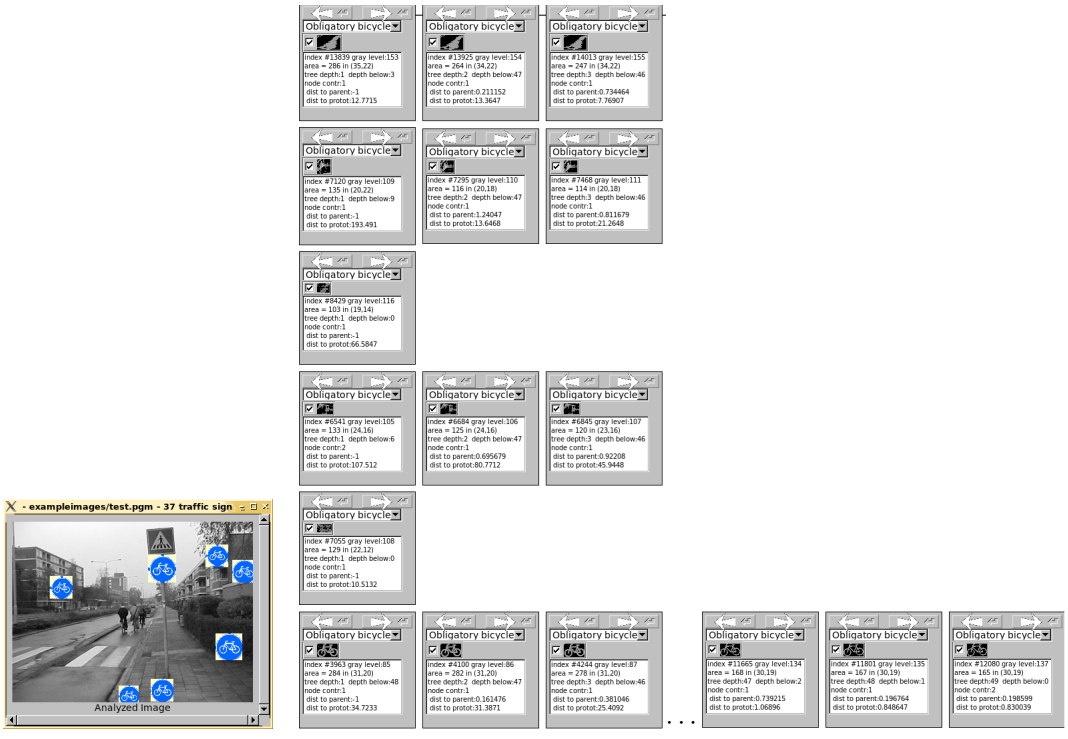
(a) threshold 0.5 - 1 hit



(b) threshold 1.0 - 9 hits



(c) threshold 1.4- 16 hits



(d) threshold 2.1- 37 hits and many background elements

The results for four different (dis-)similarity thresholds. In all the four cases the minimal peak component area is 100 pixels and the maximal is 0.21% of the image size.
The first bicycle path traffic sign hit is found at distance threshold 0.5. The bicycle path hits raise to 9 for distance 1.0. By relaxing further the distance threshold to 1.4, 25 traffic sign hits and a false traffic sign hit are reached. Eventually for threshold 2.1 37 hits are found of which just 16 are bicycle path traffic sign hits.
The furthest bicycle peak component has distance 34.7 with respect to the reference moments, relaxing the distance threshold up to this value means to catch thousands of false hits.
Note that in all the sub-figures the longest chain is constituted by true positive bicycle path hits.

Figure 19: Reference-based labeling output.

d_{t_h} is the maximum distance value between the component invariant moments and the reference moments.

The results of this labeling system for several distance thresholds are shown in figure 19 on the preceding page.

There is to see that for a distance threshold raising from 0 there is a first match for threshold 0.5. The number of matches grows up to 9 for threshold 1.0 and up to 16 for threshold 1.4; here a first background peak component appears among the traffic sign hits, a false positive. By raising further the similarity threshold many false traffic sign hits show up, i.e. at threshold 2.1 there are 37 hits but still 16 of them are real traffic sign hits. Going further, relaxing the threshold to 10 delivers 40 true traffic sign hits but this comes at the price of a total of 341 presumed traffic sign hits. The total number of bicycle path traffic sign hits for this image turns out to be 52 at the distance threshold value 10, but the number of presumed hits for this threshold is unclear.

It is clearly observed that after the first few traffic sign hits, the *precision* of the reference-based filtering degrades very quickly while relaxing the threshold value. This behavior makes it impossible to set up a pure data set using this method.

9.2 SUPERVISED REFERENCE-BASED LABELING

Here is considered the behavior of the supervised labeling system as represented by the block diagram in figure 11 on page 39. Analogous to previous section (9.1), the focus is here on the system behavior for the production of a correctly labeled data set, constituted by only true positive traffic signs and true negative traffic signs, the background elements.

The starting point here is the output of the reference-based filter; this output is then corrected until only the true traffic sign hits are left. At that point the tags (or class labels) can be stored.

The idea is to use the added heuristics (the discarded root and leaves visualization and the minimal area ratio between parent and child nodes) in order to set up a pure data set.

These heuristics are meant to one side to suppress false traffic sign hits which are too different in size from the parent, and to the other side to show the true traffic sign hits which are rejected by too agressive distance parameter of the reference-based filter.

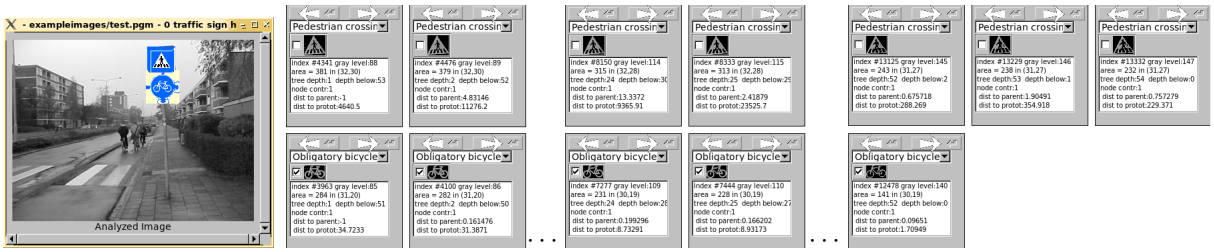
The results confirm the expectations: low similarity values can capture some traffic sign Max-Node, but either some positive nodes are lost or a lot of false positive nodes are accepted; high similarity values instead lead to a huge number of false positive traffic signs.

The parent-child area ratio heuristic simplifies a little the resulting Max-Tree from obvious false positives, while the root-leaves discovery heuristics is very practical in finding all the true positive traffic signs.

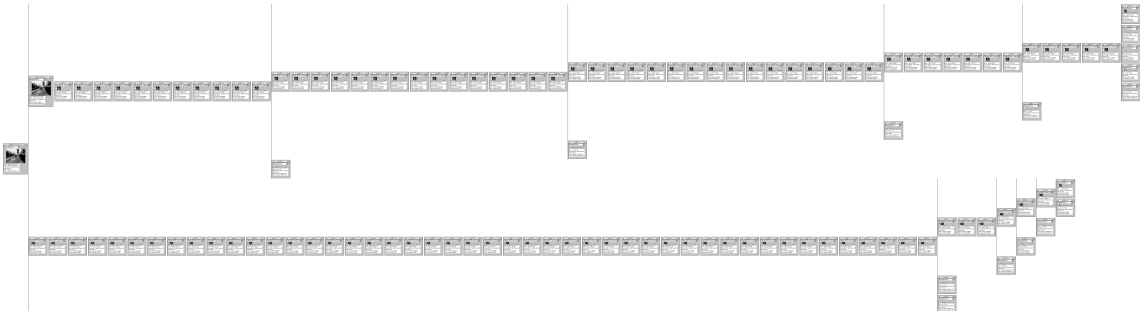
In the end the aimed pure data set can be constructed with this method; the huge number of false positive nodes requires too many inspections and corrections to make this method really practical as a labeling system.

9.3 SUPERVISED CONTRAST-BASED LABELING

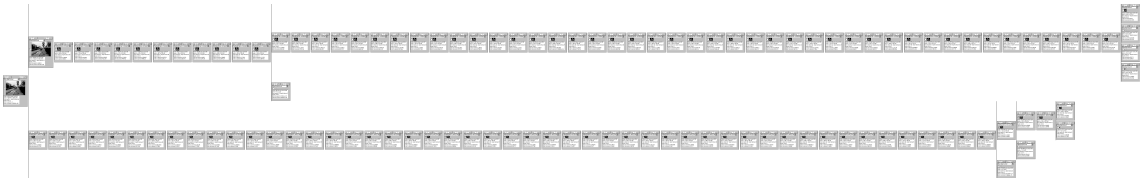
In the previous sections (9.3 and 9.1) the burden of the false hits is evident; the focus now is on the system behavior for the production of a correctly labeled data set and on the simplification power of the contrast filter.



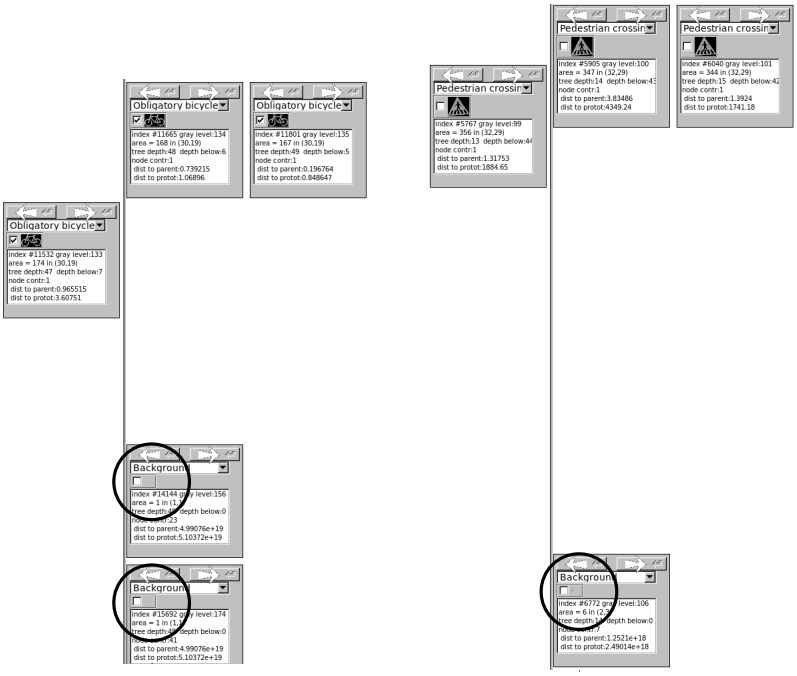
(a) Final output Max-Tree with 52 pedestrian crossing hits and 54 bicycle path hits.



(b) Unpruned output Max-Tree. Tree view with discarded root and leaves.



(c) Pruned output Max-Tree. Tree view with discarded root and leaves.



(d) Detail of the Max-Tree with pruned singleton nodes highlighted.

(e) Detail of the Max-Tree with pruned pixel region nodes highlighted.

The set of all labeled peak components, for both the pedestrian crossing and the bicycle path classes, is shown in sub-figure 20a. Note the typical chain-like distribution of the (nested) peak components.

The corresponding Max-Tree with the neighboring discarded parent and child nodes is visualized, before pruning (sub-figure 20c) and after (sub-figure 20b). Note that the pruning based on the parent-child area ratio already suppresses several irrelevant nodes.

Sub-figures 20d and 20e show examples of nodes removed from the same Max-Tree by the pruning. In the first case singleton pixels are removed, in the second case a small region of six pixels is removed. The pruned nodes are highlighted by circles; note the difference in size with the parents.

Figure 20: Supervised labeling output

The starting point is again the output of the reference-based labeling system; this output is filtered by means of contrast and then the same inspection and correction step of previous section (9.3) is applied.

In the labeling output of the reference-based labeling system, the peak components for a traffic sign show a particular conformation: the peak components are aligned into a chain where the right one is a child nested into the left one.

Recalling the recursive Max-Tree image decomposition (see chapter 3), this chain of peak components can be interpreted as the footprint of a hill where each time a slice is cut off from the bottom of the hill. In this way the footprint slowly but surely shrinks while generally keeping for most of the time its characteristic shape.

In the case of traffic signs, the high hills are due to the characteristic high contrast. After all, traffic signs are meant to be salient and to be easily recognized by humans and this is often achieved through high contrast in images.

This is not the case for most background peak components; for them the contrast across the chain of false hits is in most cases very low. See for instance the false hits in figure 20 on the preceding page.

This is an interesting characteristic which can be exploited by a contrast filter as in the supervised contrast-based labeling system (shown in figure 12 on page 40); this is the main rationale of this labeling system.

The most salient results of this supervised contrast-based labeling system (based on the algorithm 8.1 on page 44) are evaluated and shown in figure 21 on the following page.

Also for this labeling system the resuts confirm the expectations: applying a contrast filter to the reference-based system's output leads to a significantly simplified Max-Tree; with an adequate contrast threshold value most false positive nodes are effectively removed and the number of corrections diminishes to a reasonable level.

9.4 LABELING CONSIDERATIONS

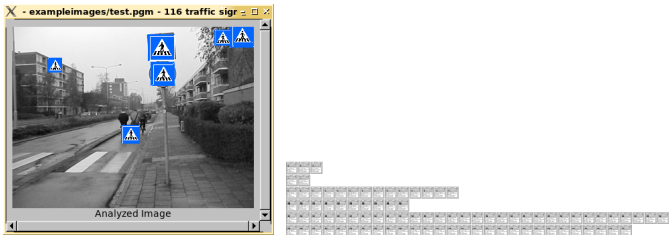
While labeling the data, a number of combinations different from the leading case (the Hu moments with normalized euclidean distance) were considered for the above experiments. Different moment and distance combinations result in different performance for traffic sign discrimination. This translates to different starting situations for the reference based filtering where the filter output can already be heavily "polluted" by background hits (false positives), when reaching the first peak component. The final impression is indeed that the leading case in general is the best performing.

In any case the human supervision cannot be limited to the approval of the filter's output. Correcting the output by "expanding" the results is necessary to find all the true positive nodes too and to this end visualizing the Max-Nodes in a tree-way turned out to be insightful and very practical; this is also true for the "visualize the first rejected leaf or root Max-Node too" heuristics. This visualization option greatly helps in finding and tagging all the Max-Nodes humans can recognize as being traffic signs. This is very useful to maximize the number of traffic sign examples which can be found in an image.

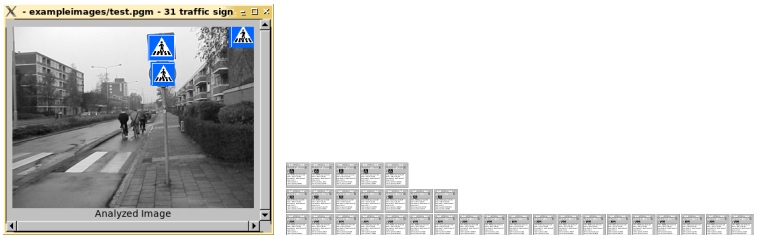
Note that the first peak component reached by relaxing the distance threshold is not necessarily the root of the peak component chain for a traffic sign. Most often it is a node somewhere in the middle of the chain as the top nodes and the bottom nodes are in general degraded, the top ones by *leaking* and the bottom ones by *noise* in the image, due to the camera being *out-of-focus* and/or *moving*.

(a) Max-Tree for threshold ∞ and 0 contrast - 582 hits

(b) Max-Tree for threshold ∞ and 10 contrast - 253 hits



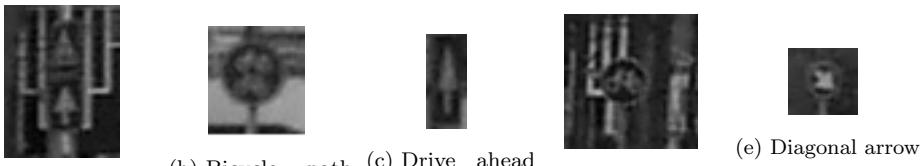
(c) Max-Tree for threshold ∞ and 25 contrast - 116 hits



(d) Max-Tree for threshold ∞ and 50 contrast - 31 hits

The set of all labeled peak components, for the bicycle path class is shown for minimal contrast values 0, 10, 25 and 50. In all the cases a minimal area of 100 pixels, a maximal area of 0.36% and infinite similarity threshold are used.
Note the progressive suppression of the false positive traffic sign hits; some background elements might have high contrast as observed here.
Because no attribute similarity is used, the bicycle traffic sign is accepted too due to his high contrast.
Note that in all the sub-figures the two longest chains are constituted by true positive bicycle path and pedestrian path hits.

Figure 21: Contrast-based Supervised labeling output.



(a) Ped. path and drive ahead 43×57 pixels

(b) Bicycle path and motor path 36×41 pixels

(c) Drive ahead 15×35 pixels

(d) Bicycle path 50×50 pixels

(e) Diagonal arrow 30×30 pixels

These image snippets are provided of size in pixels.

Figure 22: Examples of too small traffic signs

The tree-like visualization of the Max-Tree also is the fundamental step to realize the high contrast across the nested traffic sign peak components.

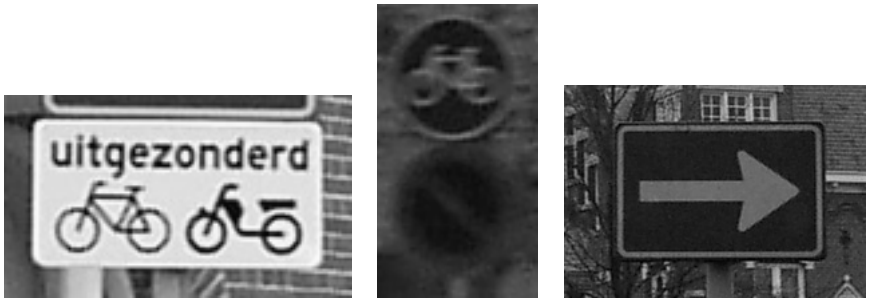
In this chapter mainly the testing picture of 360×270 pixels of figure 1 on page 3 is considered; this picture has 20217 peak components (the number of nodes in the Max-Tree). The false positives problem is staggering in particular on the pictures of the used images data set. These pictures have a resolution of 2160×1440 pixels (3Mpixels) and have some 200.000~800.000 peak components. In the case of some of these images more than 10.000 false positive nodes show up after filtering; in principle for each of these nodes visual inspection is needed, which is time-consuming.

So far in the labeling trials distance thresholds of minimal 100 pixels and maximal 0.21% or 0.31% of the image size were used. In general the labeling of the 93 images happened by using a threshold λ_{min} of minimal 40 or 50 pixels and a threshold λ_{max} of maximal 0.1% up to 0.6% of the image size; if the choice to ignore very small traffic signs is made, a minimal threshold $\lambda_{\text{min}} = 100$ pixels is advisable.

The maximal area is due to the absence of very large (relative to the image size) traffic signs. The reason is that in this image data set relatively very large traffic signs do not occur. All the images simply contain a traffic sign photographed from a not too close distance. Pictures with a very large traffic sign are not likely in a real world situation, simply because they are generally observed from a certain distance.

The minimal area is due to blurring and quality degradation which affects small connected components in general, including traffic signs. The problem is that very small components with size $\lambda < 50$ are not likely to be recognized by humans due to their shape but more likely by *color* and/or by *context*.

Considering that the recognition by (invariant) moments is shape-sensitive, it is relatively unfair to expect the components decomposition to reveal these very small traffic signs of size $\lambda < 50$ and therefore those small traffic sign are neglected. In the context of dirver's assistance, this is not necessarily a problem: drivers are likely not willing to be warned about too far away traffic signs. Moreover, this issue vanishes as the observer approaches the traffic sign. Besides this, components with size $\lambda < 20$ pixels within 3Mpixels pictures generally become too blurred to be recognized. There is just an exception: the drive straight ahead signs could be tagged down to the size of 26 pixels, probably due to the relatively simple shape.



(a) Inverted bicycle sign

(b) Highly blurred bicycle path

(c) Drive-ahead with different proportions

Figure 23: Examples of ignored traffic signs

10 IMAGE MOMENTS DATA SET

Once for each image the Max-Node tags are stored, the image moments data set is generated by using the data set production system (see section 8.1) for each pair of image and Max-Node tags;

The tagging strategy aims to maximize the amount of traffic sign examples; this turns out to be highly desirable as the number of components of an image decomposition is very large and by far the most are background components. The background components in the end are roughly 10^4 times more than the traffic sign, another fact which has to be taken into account.

The large amount of components puts another burden on this project. Hundred of thousands components for 93 images need to be translated into moments data of 6 different types which have dimensions (and therefore of vector components) varying from 7 to 60. All these vectors in all these components need to be stored and the total storage capacity used is 445GB, another fact which needs to be taken care of. For an overview of the various data set sizes see table 8.

Traffic sign type	Pedestrian path	Pedestrian crossing	Drive straight ahead	Bicycle path	Background
nr. pictures	5	3	32	31	93
nr. patterns	155	284	2305	2315	45,193,735

Table 7: Distribution of traffic sign patterns

Moment type	Geo-metric	Central	Norma-lized Central	Hu	Flusser and Suk Affine	Flusser and Suk Complex	Total
*Data set size(*GB*)*	142.9	87.5	98.5	11,6	13.7	91.7	445.9

Table 8: Data set sizes

Part III

DATA SET PREPROCESSING

features data set for the automatic traffic sign detection is constructed; at this point it is clear that direct application of the selected learning techniques (and of many other techniques as well) is infeasible; this holds also for a single image. This data set size problem makes it necessary to pre-process the data. In particular, the core problem seems to be the number of image moments deriving from the background components.

In this part the approach to data set reduction is treated. A data set reduction is necessary for the collected data: the data set is too large (see table 8) to be treated as a whole. The size of the data set is such that it cannot be processed and/or learned directly, as it is often the case for data sets of smaller size. The data simply does not fit into the physical memory of modern computers, so any processing is doomed to be slowed down by disk I/O.

Even for single data files deriving from a single image any form of direct learning is not feasible: in general the images from the given data set are decomposed into $\mathcal{O}\left(10^8\right)$ components; the training set is then of the same magnitude which makes kNN too memory demanding and computationally too slow and LVQ even not feasible. Consider the case where LVQ is trained in 1000 epochs with the whole training set; a single training implies analyzing $\mathcal{O}\left(10^8\right)$ patterns and a staggering $\mathcal{O}\left(10^{11}\right)$ patterns have to be processed.

Furthermore, statistical properties of the data set need to be considered as well. While for kNN it is quite natural to use raw data (as long as the class densities do not differ too much), LVQ might be sensitive to differences within the class sizes and require further data reshaping. This aspect is analyzed in part IV.

Data *reduction* and *selection* is therefore necessary to make the traffic sign recognition problem treatable in the first place; at the same time which data can be safely removed is unclear. In the following chapters two methods are discussed, namely area-based pre-processing in chapter 11 and distance-based pre-processing in chapter 12. The environment chosen to implement the computations from now is Matlab©[1] as it has very practical I/O and plotting facilities.

1 MATrix LABoratory, see http://www.mathworks.com/products/matlab/

The first idea to reduce the size of the data set is to look at the distribution of the components areas. The idea is to remove from the data set all the components with an area outside of the interval $\left[\lambda^{\min} \dots \lambda^{\max}\right]$ described in the previous chapter (10).

A reason to apply $\lambda_{\min}$ is that the number of represented shapes diminishes when the number of pixels of a region decreases to one, making different entities indistinguishable. As entities which cannot be discriminated are unlikely to be told apart by any classification, they are of no use. This phenomenon is amplified by noise, blurring, fog and/or image focus problems.

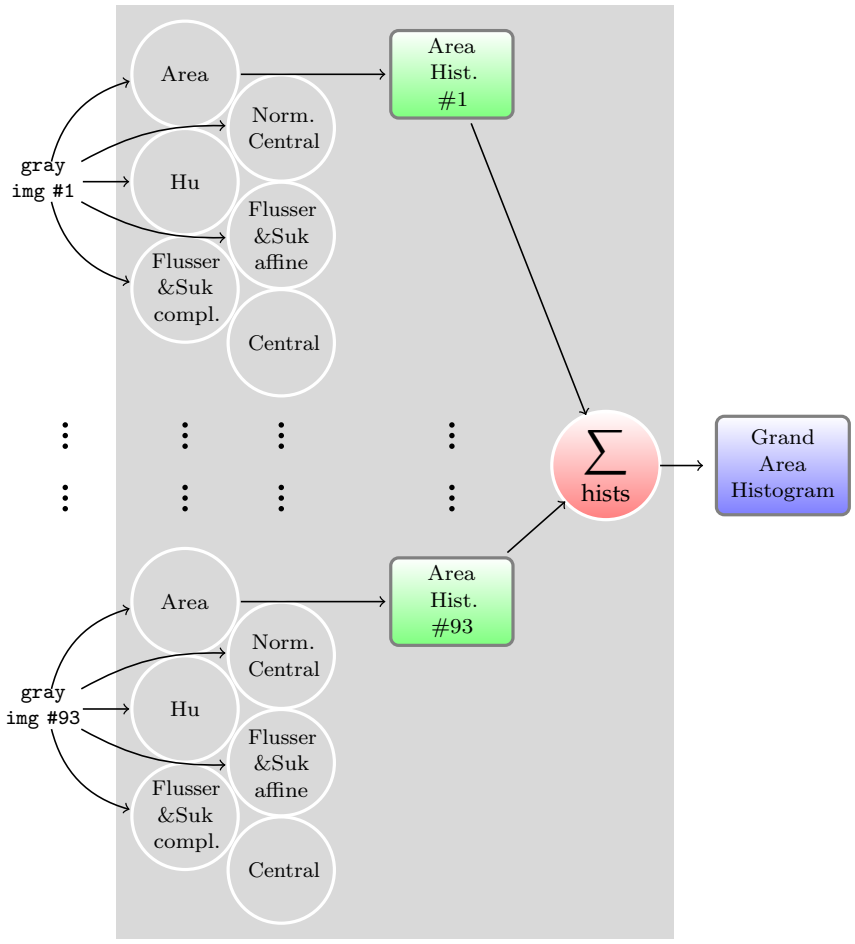


Diagram representation for the construction process of a total area histogram for all the Max Nodes of the data set.
The shaded box encompasses all the phases of the scatter and gather process.
The summation symbol represents the sum of the image histograms.

Figure 24: Area histogram computation scheme

In order to take sensible component size thresholds, the histogram of the area distribution of all the components of the whole data set is to be computed. Because of the size of the whole data set, it is not possible to naively merge all the data sets and count the components falling in each area-bin. This can be overcome

by considering first the components relative to a single image and then merge together (by means of a summation) all the partial histograms. This is a classic *scatter/gather* strategy. Figure 24 on the previous page shows the whole process.

The histograms resulting from this computation process are shown in figure 25 on the facing page. There the histogram area intervals are $\left[1 \dots 5 \cdot 10^4\right]$ for the traffic sign components and $[1 \dots \mathsf{size}(\mathtt{img})]$ for the background.

A couple of consideration are to be made about this data. First of all, the two small traffic sign classes (pedestrian path and pedestrian crossing) show a rather compact distribution with respect to the area, while the two large ones (drive ahead and bicycle path) are much more smeared out. This is likely to be a direct effect of the class size asymmetry within our data set: for the two small classes there are not enough patterns to have smeared distibutions. Moreover, especially the distributions of the two large classes can be seen as a series of bumps of different sizes; as the size of the traffic signs depends on the photographer's distance, it seems that the pictures were taken at a small number of distances (discrete distance steps), maybe an inconvenience due to the ad hoc image data set construction.

The strongly asymmetric distribution of the background components in figure 25 on the next page is another remarkable point. A plain histogram visualization shows just the first bin, therefore a log-histogram is necessary. The log-histogram reveals that the background components roughly follow a inversely exponential distribution across the first bins: the smaller the component's size becomes, the higher their frequency is.

This is likely to be inherent to the components tree decomposition. The tree is rooted on the peak component with the minimal gray intensity of the image; the children flow by looking at higher gray intensities.

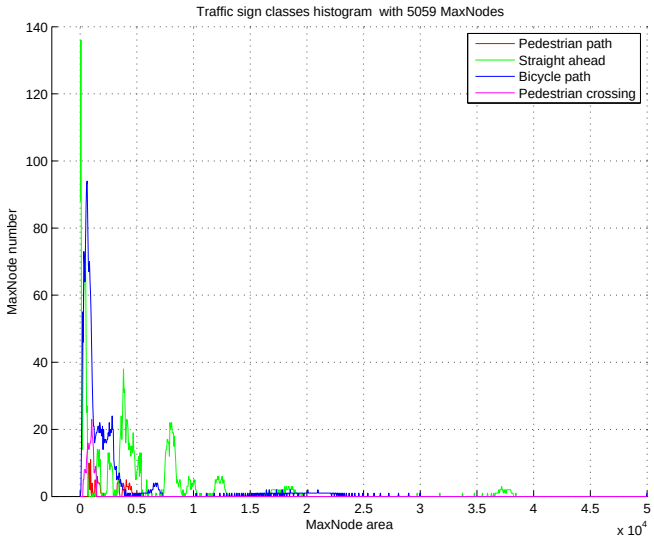
Each time the transition to a higher gray intensity is made, fewer pixels have that minimal higher gray intensity and components may break up into (possibly several) smaller peak components. Those new child components are *nested* in the parent one.

This describes a nonlinear behavior with respect to the gray intensity: the number of components does not grow with the gray level ($\#\mathsf{c} \sim \mathsf{h}$) but more likely with a branching generally taking place at each level ($\#\mathsf{c} \sim \mathsf{k}^{\mathsf{h}}, \mathsf{k} \geqslant 2$). This corresponds with an exponential behavior. Indeed on average images contain a vast number of singleton peak components (the leafs in the Max-Tree). This behavior is inherent to the component branching across the gray levels and is also consistent with the fact that images generally show far more small scale details than large scale ones.

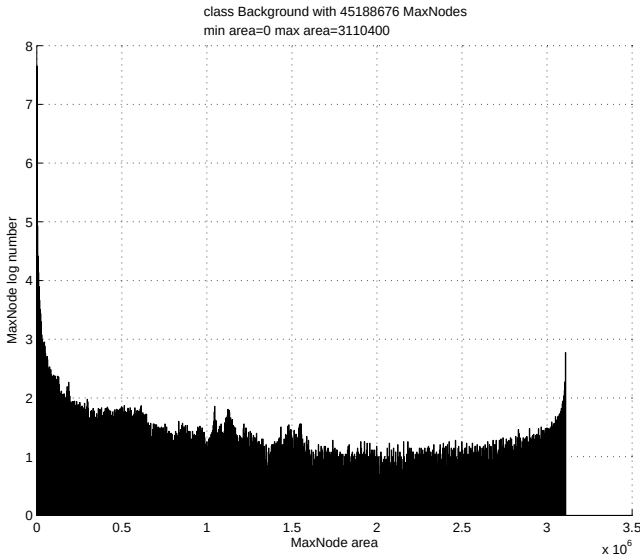
The components of the traffic sign class have a different behavior; while labeling (see chapter 9) it turns out that those components are quite homogeneous for the same traffic sign. The background nodes are nevertheless present and it is wise to consider their "explosion" when performing further analysis. Right now this explosion implies a negligible prior for the traffic sign classes and a bloated prior for the background; the background then saturates naïve classifiers.

Moment invariance might cause further consequences: invariance to translation, scaling and rotation together with a smaller number of described shapes (because of small sizes) might have awkward consequences in terms of discriminability of the shapes.

Nevertheless this analysis is useful; by defining simple minimal and maximal size thresholds (e.g. defining the sizes interval of interest $\left[\lambda^{\min} = 50 \dots \lambda^{\max} = 10^5\right]$ for our 3Mpixel images) the data set can roughly be reduced to half its size without losing any (likely) sensible information.



(a) Traffic sign classes



(b) Background class

Area histograms of the Max-Tree components subdivided by class. Note that the background class histogram has a logarithmic ordinate axis for visualization purposes; this class is likely to have an inversely exponential distribution.

Figure 25: Class-wise area histograms

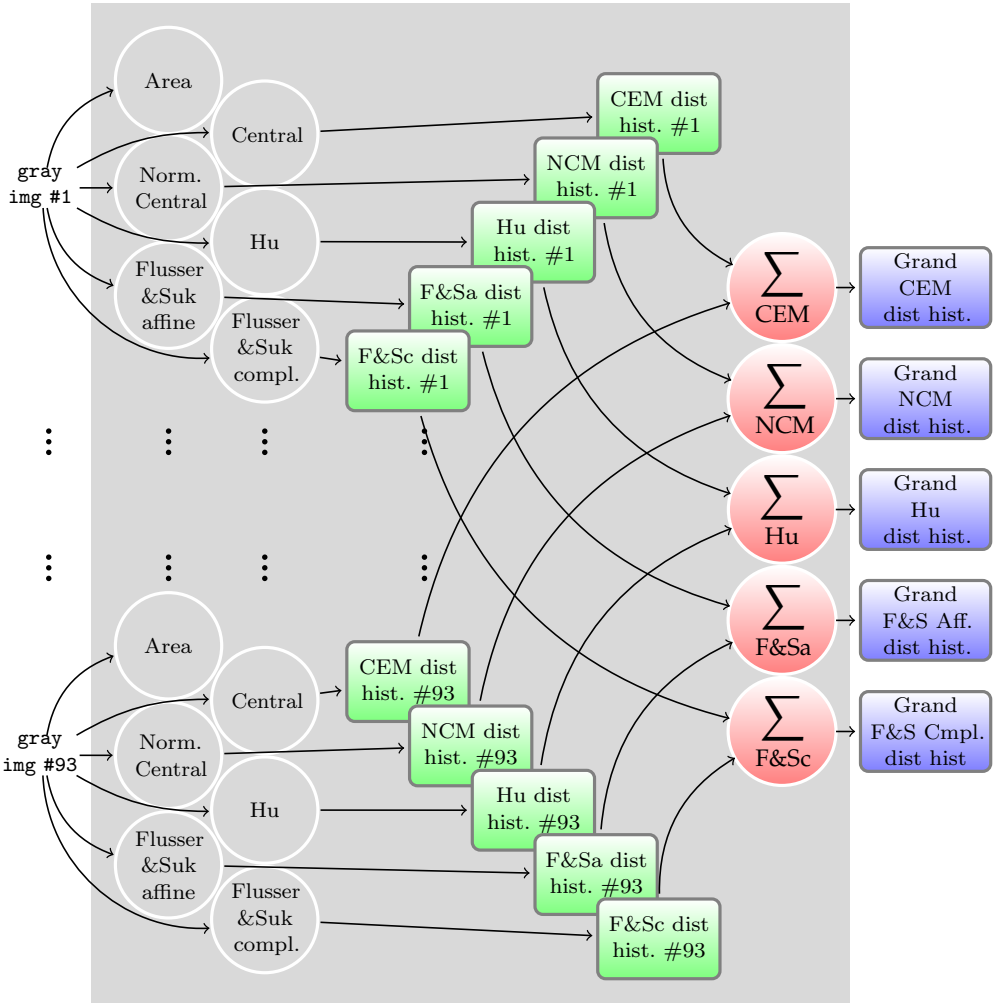


Diagram representation for the construction process of the total distance histogram for all the components of the data set for the six different moments and the four traffic sign classes. The shaded box encompasses all the phases of the scatter and gather process.
For simplicity the four partial histograms of the distances for a single moment type and the considered moment orders are represented together in a box; in the implementation all the partial histograms are stored together in a file, one for each image.

Figure 26: Distance histogram computation scheme

Looking at the area of the components can help to reduce the data roughly by half; this is interesting but it is not enough for the traffic sign data set where the background components are roughly 10^4 times the number of traffic sign components.

The purpose of this data set is to train, in an appropriate way, a number of classifiers (see chapter 6); to this end not all the background components are useful. In particular, the chosen classifiers rely on *metric comparisons* and therefore exploring the data distances might give more insights. For a data set size n there are $\mathcal{O}\left(n^2\right)$ distances to consider where n is large. Therefore evaluating all the distances in the data is infeasible and selecting a small portion of the data is necessary.

While labeling the data the basic assumption is that all the traffic sign components are likely to *resemble* one of the *ideal* traffic signs (see figure 8 on page 35) and the moments of the components are similar to the reference ones. Because of this, a natural way of choosing such reference points is to look at the similarity distances (set out in section 4.1) in the data set and to consider again the moments of the ideal traffic signs as reference points.

Considering that for different moment types and moment orders the distances may change, this distance analysis is to be performed for any combination of moment type, moment order and reference point. For the Hu moments just two orders are defined (3 and 4), for the Flusser and Suk affine only two orders are useful (4 and 6; the first order consists of just of scalar, which is unlikely to satisfy by itself the recognition problem), for the Central moments, the Normalized Central moments and the Flusser and Suk complex moments the orders 4 up to 10 are considered. This leads to total of 25 different combinations to analyze.

The Geometric moments are not considered as they do not present any invariant property and learning by examples is impossible (as the examples are translated the moments become different, making any classification inference unfeasible).

For any of the 25 combinations, the four reference moments define four distances; considering four target classes also implies 16 histograms for each combination and a total of 400 histograms to be computed. This relates to all the image moment sets and for the grand total histograms.

Similar to the area histograms of chapter 11 a scatter/gather scheme is needed to compute all the histograms, as illustrated in figure 26. Moreover, to make the histogram computation feasible the disk I/O has to be minimized, therefore for each moment data set all the analysis is computed in one go. Unfortunately Matlab©[1] lacks of strong typing and complex variable types, which makes the glue code needed to cycle through all the computations and graphs rather cumbersome. A plotting bug caused identical graphs for different orders, obviously not a likely outcome.

A workaround seemed a reasonable solution: instead of applying the scatter/-gather to the histogram computation, the scatter/gather is applied to the *sampling* of data sets themselves. This is illustrated in the diagram of in figure 27. Here an asymmetric sampling is performed, where all the moments from traffic sign components are put into a samples set together with a certain number of background moments (e.g. 10.000); this is necessary in order to preserve the relatively scarce traffic sign information.

This way a *Grand Sample* of each moment type data set is produced which is likely to contain all the data necessary for future learning while statistically behaving similar to the entire data set. This grand sample contains all the 5059 traffic sign moments together with 930.000 randomly sampled background moments (for each moment type).

After performing the different analysis on the sampled data a control is performed on the histogram shapes against the non-sampled histograms deriving from the scatter/gather procedure; the result is that indeed the sample data histograms share the same shape with the total moments data set histograms.

12.1 PLAIN DATA

The distance histograms for each of the moment type and order, reference moments, distance type and class are now available and it is possible to look for differences among the distance distributions of target data and background. In case of large distances for a significant number of background elements it is worth to consider their removal from the data set. To evaluate this for each of the considered

1 MATrix LABoratory, see http://www.mathworks.com/products/matlab/

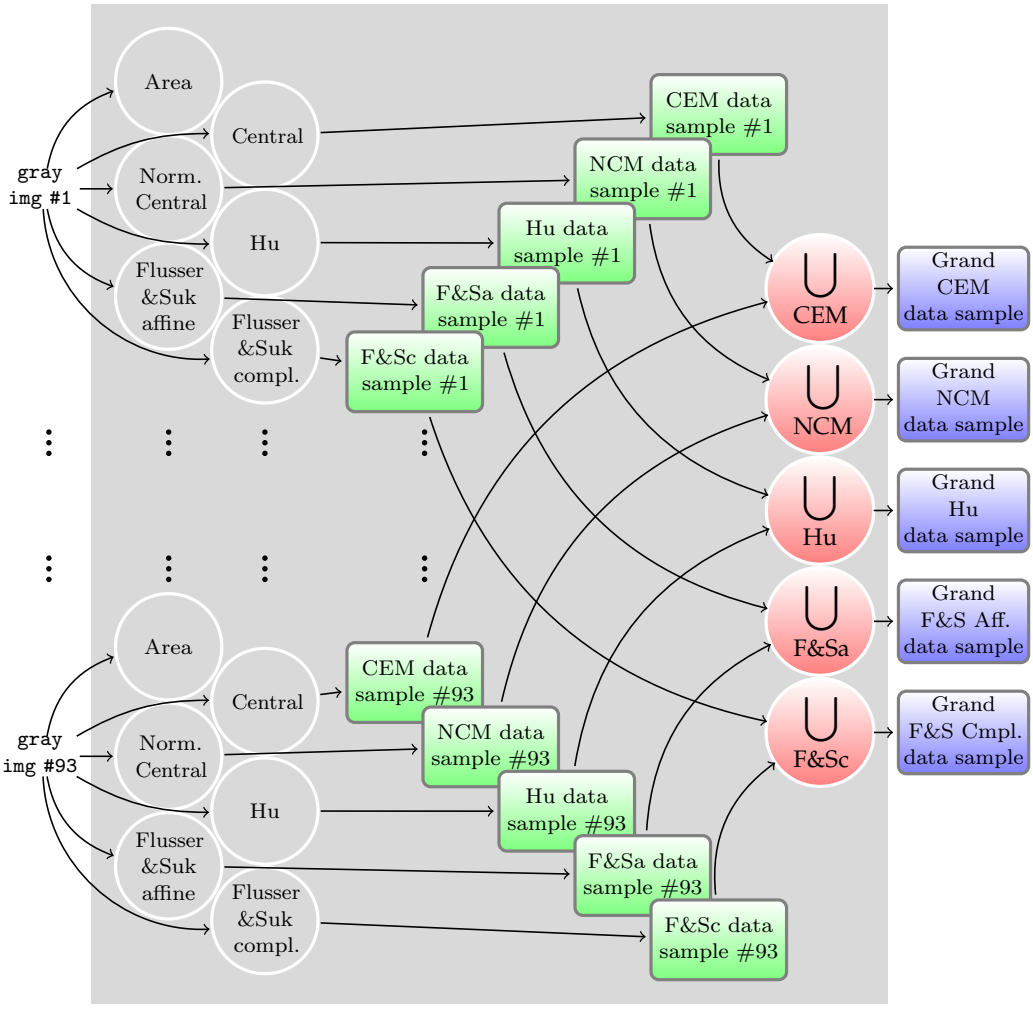


Diagram representation for the sampling process of the total moments data set for the five considered moments. The shaded box encompasses all the phases of the scatter and gather process.

Figure 27: Data set sampling scheme

combinations two histograms are necessary: one relative to the distances of target data to the reference moment, the other relative to the distances of background data to the *same* reference moment. A data perspective is searched where a large part of the background data is to tell easily apart from the traffic sign data.

If the reference moment is thought of as a *receiver*, a *Receiver Operator Curve* (ROC, see section 6.4) is a useful analysis tool which is computed based on these two histograms. The target data histogram describes the distribution of *true positives* distances (to the reference moments) distribution while the background data histogram describes the distribution of *true negatives* distances (to the reference moments).

Based on these histograms it is possible to draw a curve by considering a distance threshold rising from 0 (no acceptance; distances are positive quantities) to ∞ (total acceptance). As the data is discrete, a step curve is to be drawn; here the technique is to draw a new step each time a new true positive is reached; the curve stays flat up to the next true positive data point because any false positive point can only contribute by pulling the curve away from the true positive rate axis.

In the context of traffic sign recognition, the data elements have one of five possible labels: background or one of the four target traffic signs. For simplicity

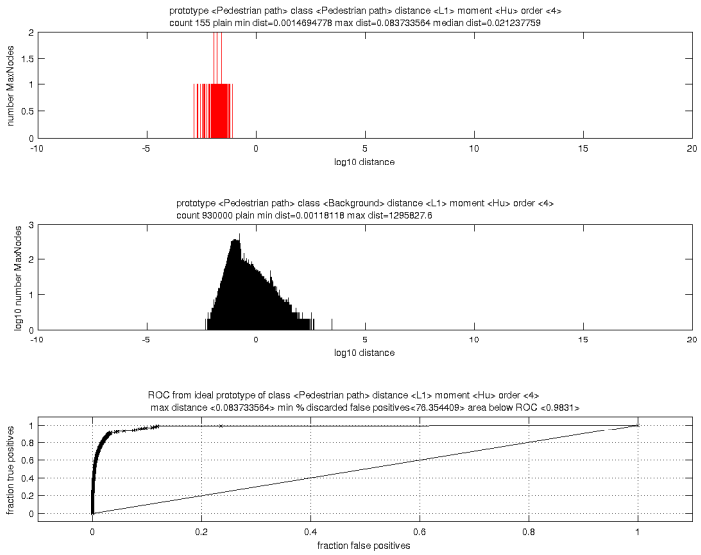


Figure 28: An example of target class and background histograms and relative ROC curve

the dichotomy traffic sign target (any of the four) versus background is considered; after all, the background class is expected to be the troublemaker.

The ROC curves are a useful derivate of the sampled data sets. The scatter/-gather procedure cannot be applied for these curves because a grand ROC curve is not equal to the naïve average of a series of ROC curves, one for each data set part. Computing a grand ROC curve turns out to be non-trivial and is left out of the project scope.

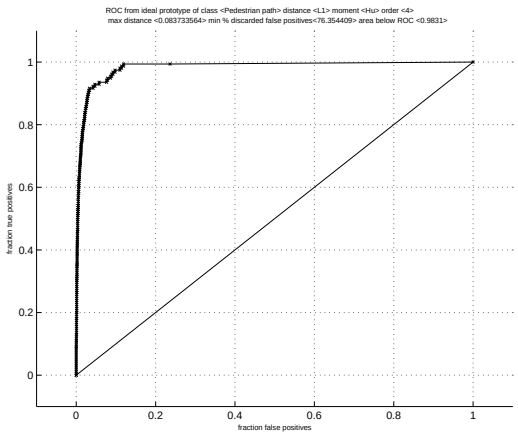
An example of target class histogram, background class histogram and its derived ROC curve is shown in figure 28. The two histograms and the derived ROC are computed for each of the considered moments and target classes; a few examples of resulting ROC curves are compared in figure 29 while a larger overview is in figures 52 and 51 in the appendix.

For comparison purposes it is unfeasible to consider all the single ROC plots, therefore the AUC is uset to estimate the classification performance; for this reason a synthetic plot considering just the area under ROC (AUC) for each of the analyzed combinations is used. The resulting AUC graphs are shown in figure 30.

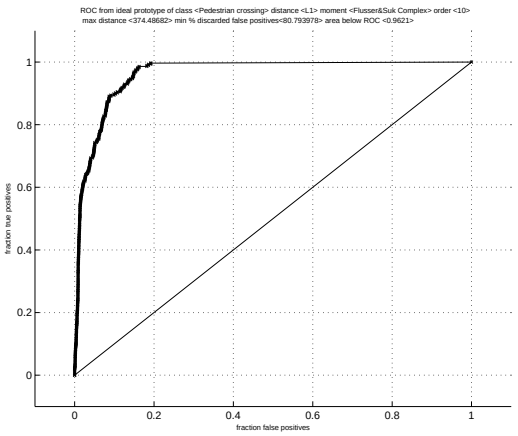
A number of conclusions flow from this synthetic overview. First of all, the performance differences due to the different metrics are rather marginal. Second, for all the moments a higher moment order leads to a higher AUC, which indicates a overall better recognition performance. Third, some classes are easier to recognize by this reference moment method than others; this is actually moment type dependent for the Central moments, for all the other moments the easiest class is the pedestrian path, the hardest class is pedestrian crossing. Except for the bad performance of the Central moments, all the AUC plots show area under ROC's of 0.8 and above which is already a quite good value.

Moreover, for the Hu moments and the Flusser and Suk moments the performance difference across the two orders is rather marginal. In general, the two path signs seem to be more difficult to recognize; at this point the reason for this behavior is unclear. Probably the small size of the pedestrian path class and the complex shape of the pedestrian crossing traffic sign might play a big influence.

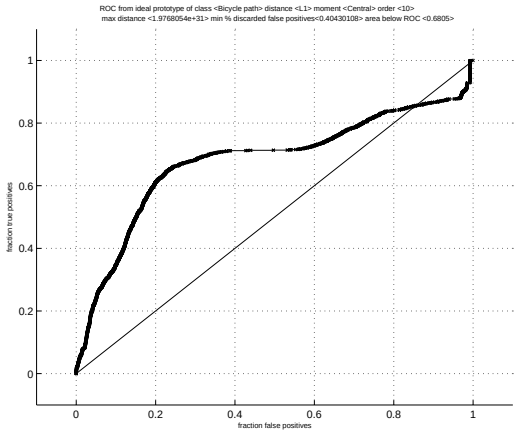
An interesting fact is that any point in the ROC step curve corresponds to a certain distance value. Considering that once a certain acceptance level for the



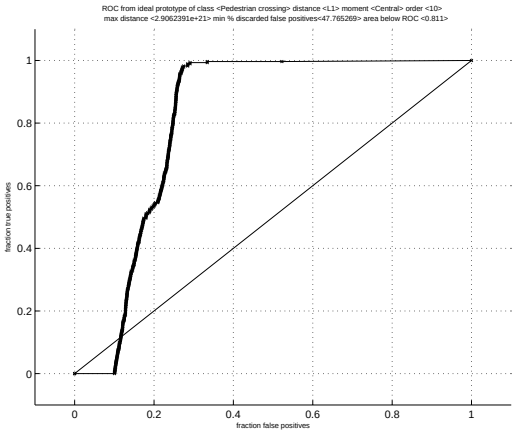
(a) High-performance ROC



(b) Medium-performance ROC



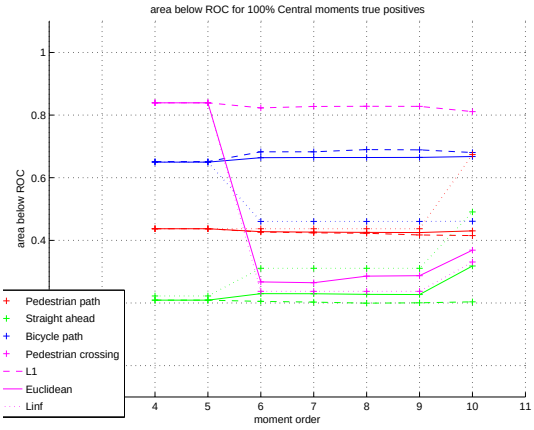
(c) Low-performance ROC



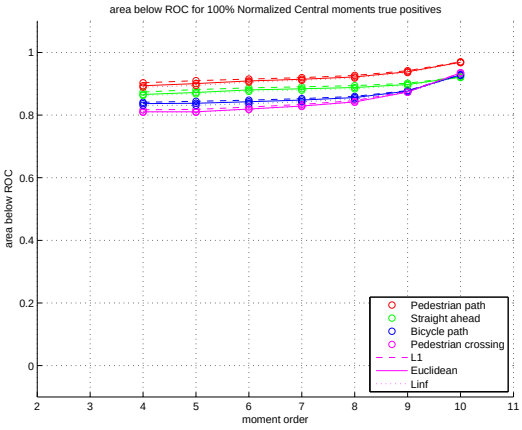
(d) Awkward ROC

Some examples of ROC curves obtained while analyzing the moments data set. For a larger overview it is possible to look at figures 52 and 51.

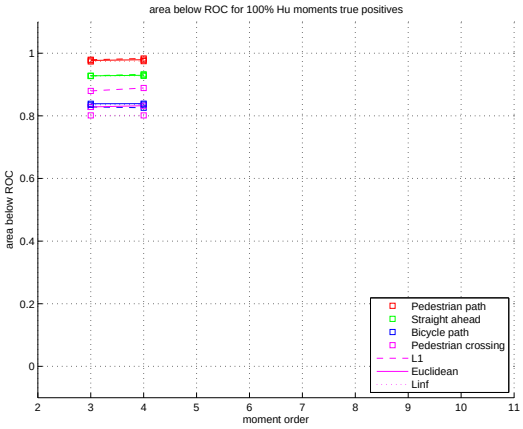
Figure 29: Examples of ROC curves



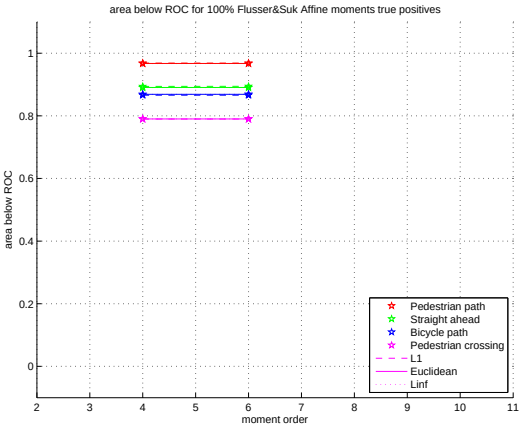
(a) Area below Central moment ROC's



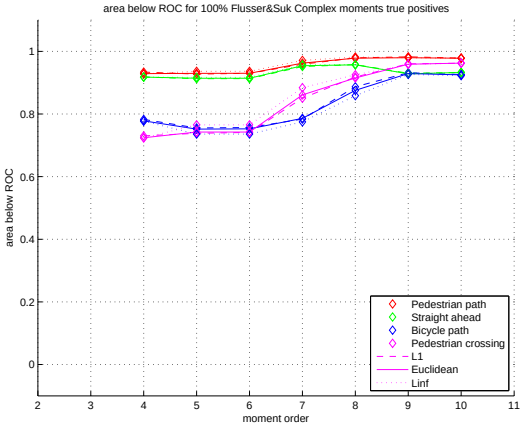
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



(d) Area below Flusser and Suk affine moment ROC's



(e) Area below Flusser and Suk complex moment ROC's

Area below ROC for 100% traffic sign acceptance with plain data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

Figure 30: Synthetic overview AUC

true positives is reached (i.e. 100%, 99% and 90%), there is no need for looking at further background data. Here implicitly true positives may be discarded as well, which means a data loss of 1% and 10% for 99% and 90% true positives acceptance as a trade-off. Those 1% or 10% data are the furthest away from the reference moments and can be considered as outlier data or simply too dissimilar from the other class members.

The AUC plots for 99% and 90% acceptance are shown in figure 46 on page 119 and in figure 47 on page 120 in the appendix. There is to see that the 99% and the 90% trade-offs lead in general to slightly higher AUC values; the general trends are unchanged.

Consistently with this, a similar behavior is to see in the graphs for the minimal discarded background components. There we see percentages of minimal discarded background data starting from 40% and generally growing up to 90% excepted for the Central Moments (which confirm their poor performance). Again it can be seen the different distances do not add information, the pedestrian crossing class is the hard one to detect while the pedestrian path is the easy one to detect, for higher orders there is a higher discarding ratio, the Normalized Central moments and the Flusser and Suk complex moments perform the best out of the bunch and their different curves seem to converge (for higher orders) around $80\% - 90\%$.

12.2 NORMALIZED DATA

The reference-based pre-processing by using plain data deliver rather good performance for discriminating the class data from the background and therefore on discarding excessively dissimilar data. The background removal ratios are insufficient when considering the 10^4 order difference in number of patterns. Discarding 90% of all the background data brings back this distributions asymmetry to 10^3, which is still too much for the learning techniques. Considering the area reduction as well still does not help to achieve a manageable situation, therefore a more effective pre-processing is necessary.

The use of the ad-hoc metric (4.28) turns out to be effective at the labeling task. Implicit in this metric is the idea of considering the reference moment as the focal point; the reference moment components function as estimates of the class data *spread* in the different dimensions.

A well-known idea also applied in Machine Learning is *data normalization*: to translate and rescale data to the zero mean and unit variance ($\sim \mathsf{N}(0,1)$). This usually happens assuming that the data is gaussian distributed around some mean μ with variance σ^2; data is normalized by performing

$$x' = \frac{x - \mu}{\sigma} \tag{12.1}$$

where x is a data point and x' is a normalized data point.

This procedure has two problems within this project. First, assuming that the data is gaussian distributed (for any data dimension) is not reasonable; nor for the traffic sign data neither for the background data it is likely that the moment component values scatter around some particular value; none of the data histograms produced so far induce such an expectation. Second, this normalization is known to be sensitive for outliers (which may bloat the variance σ).

Valid alternatives to the Gaussian μ and σ are the median and the inter quartile range which are known to robust against noise and are not explicitly tied to the normal distribution.

Given the *order statistics* (of data points X_i) $Y_1 = \min_j X_j \leqslant Y_2 \leqslant \ldots \leqslant Y_{N-1} \leqslant Y_N = \max_j X_j$, the statistical *median* of the random sample is defined by

$$\tilde{x} \ (= Q_2) \quad = \quad \begin{cases} Y_{(N+1)/2} & N \text{ odd} \\ 1/2 \left(Y_{N/2} + Y_{1+N/2} \right) & N \text{ even} \end{cases} \tag{12.2}$$

and the *Interquartile Range* (IQR) is defined by

$$IQR \quad = \quad Q_3 - Q_1 \tag{12.3}$$

There are several methods to compute the first quartile Q_1 and third quartile Q_3 values[2]. According the Minitab method, Q_1 and Q_3 are defined by

$$Q_1 = \frac{n+1}{4} \quad \text{and} \quad Q_3 = \frac{3n+3}{4} \tag{12.4}$$

for $N = 4n + 5$.

The interpretation of the first (lower), second (the median) and third (upper) quartiles is about cuts in the (ordered) data: the first quartile cuts off the lowest 25% of data, the third quartile cuts off the highest 25% of data and the median cuts the data set in half.

Based on the median $\tilde{x}$ and the IQR the adapted normalization is

$$x' \quad = \quad \frac{x - \tilde{x}}{IQR} \tag{12.5}$$

for all the components of the (different) moments.

Because the background data shows spreads in the values of the moments larger by several magnitude orders, it is necessary to calculate the median and the IQR on each (separated) moments class instead of considering a global median and IQR. The total IQR squeezes the traffic sign data points together, a counterproductive outcome.

The class-wise IQR instead rescales the class moment data roughly to the unity and possibly a large part of the background data to much larger values, effectively amplifying the values spread in the background moments and making them easier to detect.

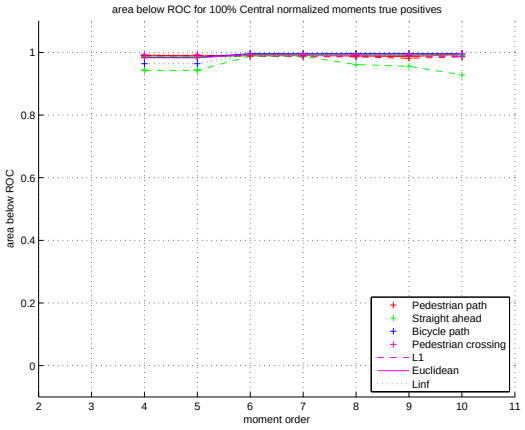
A small adaptation turns out to be necessary as for some moment components the IQR turns out to be smaller than 10^{-3}; for those components it is assumed that the information content is too little to be used and the data is just translated against the median:

$$x' \quad = \quad x - \tilde{x} \tag{12.6}$$

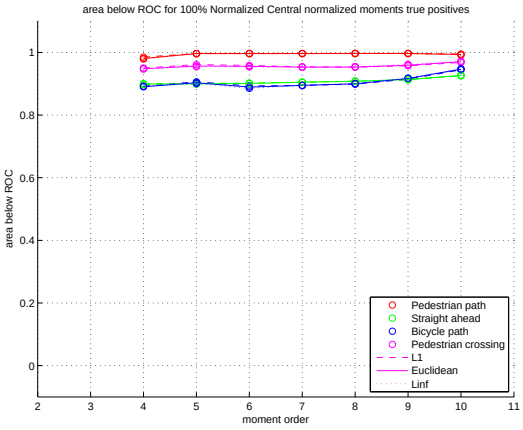
Now that an effective normalization procedure is available, the very same analysis performed in the previous section (12.1) on the sampled data set is repeated after normalizing the data with this novel IQR-normalization. The AUC graphs flowing from this analysis are in figure 31.

These results are to be compared with the analysis on plain data shown in figure 30 on page 70. The emerging picture is slightly contradictory as some trends are confirmed and others are not. Analogous to the plain data analysis, the use of different measures do not make sensible differences; the use of higher order measures generally improves the AUC values, with the exception of the Hu moments for which the 4th order degrades the performance of the 3rd. For the Flusser and Suk affine moments, the differences between different classes

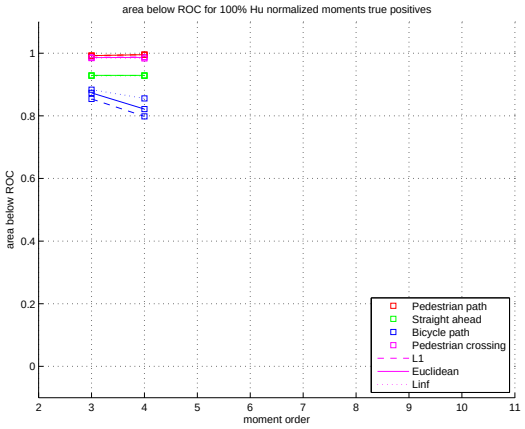
2 For a compact overview see http://mathworld.wolfram.com/Quartile.html



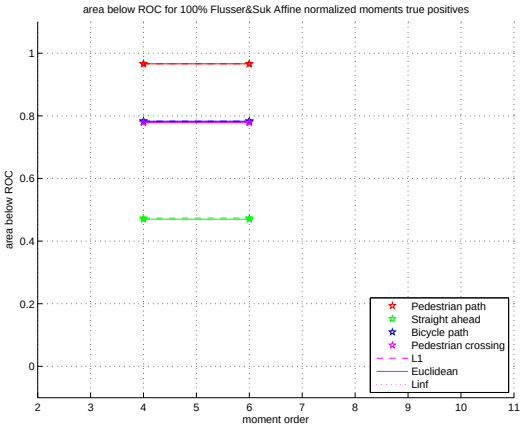
(a) Area below Central moment ROCs



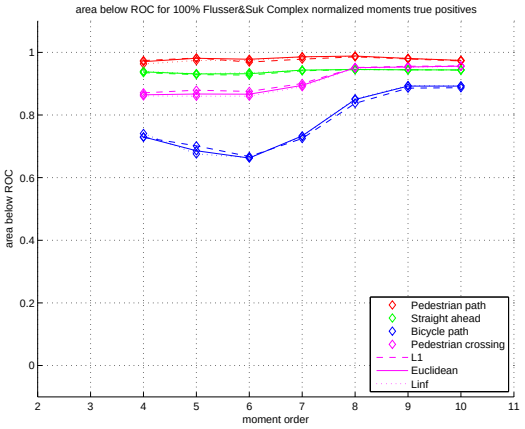
(b) Area below Normalized Central moment ROCs



(c) Area below Hu moment ROCs



(d) Area below Flusser and Suk affine moment ROCs



(e) Area below Flusser and Suk complex moment ROCs

Area below ROC for 100% traffic sign acceptance with *normalized* data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

Figure 31: Synthetic overview area below ROC normalized

get amplified while for the normalized central and the Flusser and Suk complex moments the overall performance is slightly improved by this IQR-normalization.

For the completeness of the analysis even the IQR-normalized central moments are shown; these moments are expected to perform poorly but the contrary is true, an impressive performance is achieved. At the moment no sensible reason for this is known and further analysis should be carried out to verify and explain this behavior.

Also the analysis for the minimal discarded background data is repeated on IQR-normalized data.

Part IV

APPLYING THE LEARNING ALGORITHMS

The automatic traffic sign recognition problem now is analyzed, the moment data is tagged, a data set suitable for training and testing is constructed. The data set turns out to be too large to be used directly and for this reason in the previous part (III) pre-processing is investigated as well. The pre-processing provides a certain level of data reduction but still the number of examples is too large by several orders of magnitude. For this reason the following research on traffic sign data classification will be based on the data set grand sample instead of the reduced data sets.

For evaluation purposes the data is from now on splitted in two parts: a *training* data set used to train the classifiers and a *testing* data set used to evaluate their generalization ability (classification performance) over unseen data. This may lead to erratic results and therefore any learning step and testing step pair should be repeated several times and consequently the results should be averaged; this way more reliable results are obtained. The data set is then randomly splitted each time in a training set containing 90% of the sampled data and a testing data set formed by the remaining 10% of the sampled data. This scheme is referred from now on as the 90%/10% splitting scheme.

A classic alternative could be the well-known n-fold *cross-validation*; this scheme has the limitation that it cannot be repeated more than n times, where n usually is 10. Due to time constraints it is chosen to average the performance result of a ten-fold 90%/10% splitting scheme; with the availability of more time and computing power this process can be easily repeated with a e.g. hundred-fold 90%/10% splitting scheme.

The resulting moments data sets from the pre-processing (see part III) are used directly: the analysis now is limited to the Normalized Central moments, the Hu moments, the Flusser and Suk affine and complex moments, all of the highest available order and considering only the Euclidean distance. Moreover, multiclass problems are considered instead of one of the two-class problems, namely a traffic sign versus the background; this is a more realistic setup as well.

In part III data normalization was considered; similarly, in this part the IQR based data normalization is considered as well. The normalization procedure now is not applied class-wise as all the considered classifiers are naturally multi-class; applying data normalization with respect to the median and IQR of a single traffic sign class might bias the classifier towards this class. The normalization is now performed by taking the median and IQR of all the traffic sign data together; the expectation is that the differences between the traffic sign classes and the background class are still amplified in advantage of discriminability.

In the following the application of the selected Machine Learning techniques is investigated as well as the performance: in chapter 13 the kNN classification is analyzed, in chapter 14 the CCM classification is analyzed and finally in chapter 15 the LVQ classification is analyzed.

13 KNN CLASSIFICATION

13.1 PLAIN DATA

The k Nearest Neighbors (kNN) technique is explained in section 6.1; in this chapter the kNN classification technique is applied to the sampled data set. In theory if the kNN classifier turns out to be able to classify the traffic signs, it can be deployed for the final detection in a real world application. In practice, its simplicity in skipping a learning phase and using the training data to infer a classification of new data makes it memory and computing time consuming; nevertheless its performance is a useful reference for any learning algorithm and in this chapter it will be analyzed against the sampled traffic sign data set.

The kNN learner is a natural multiclass classifier: as long as examples of a class are present in the stored examples set, unseen data can be classified as belonging to that class. For this reason it is decided to use the training and testing sets without any modification; these data sets are just the result of the 90%/10% splitting where as many traffic sign examples as background examples are used. This background class data reshaping is necessary because of the too large number of patterns to process. At the same time some degree of background class data excess is useful as for any training and evaluation of kNN new background examples can be drawn at any time; the same does not hold for the traffic sign data.

For a better insight of the kNN performance also a classification on the traffic sign without background data is performed; the four class problem is expected to be easier to learn and the introduction of background data is expected to deteriorate the performance. The classification of the mere traffic sign data will be referred from now on as the *four class problem* while the classification with the presence of background data will be referred as the *five class problem*.

The kNN classifier has the fundamental parameter k, the number of nearest neighbors which are involved in the voting scheme. The appropriate value for this parameter is not known a priori and it cannot be easily inferred; the usual way to deal with this is to select the appropriate value for k a posteriori, based on the classification performance. The usual range is $k = [1, \dots, 21]$, k even (for k odd voting draws might occur often); for the sake of safety the ranges $k = [1, \dots, 51]$, k even and $k = [61, \dots, 201]$, $k = 61 + n \cdot 10$ are chosen.

The classification performance of kNN for the four class problem with the Hu moments is shown in figure 32 on the next page; the performance for the five class problem with the Hu moments is shown in figure 33 on the following page. The results deriving from the other considered moments for the four and five class problems are shown in the appendix (figures 58 and 59).

A glance on these results confirms most of the behaviors which arose in chapter 12 while dealing with the reference-based pre-processing.

The pedestrian path and the pedestrian crossing classes appear hard to detect and induce a classification error generally several times higher than the error on the other classes, for all the different moments and also when the background is explicitly considered.

The general trend is that a higher k value leads to higher classification errors; only with the Flusser and Suk moments the average error stabilizes but this happens when the patterns of the two pedestrian classes are totally miss-classified.

The performance degradation is very evident for the two difficult classes; those classes are also the smallest and for large k the classifier asks a consensus of

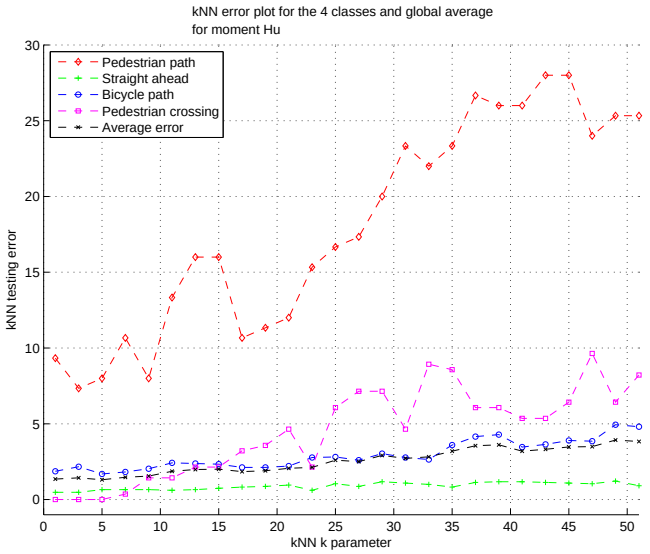


Figure 32: kNN plain Hu moments data 4 classes classification results

the same magnitude of the size of the entire class. This obviously increases false negative classifications, mostly to the advantage of the background class.

In the four classe problem for all the moments the average classification errors start (at $k = 1$) around $2\% - 3\%$; the impression is that the best classification performance is found with the Hu moments. Switching to the five class problem degrades the overall performance as expected: average classification errors now start between $5\% - 15\%$ and grows with k.

In particular the performance behavior of all the classifications for increasing k suggests the presence of overlap between the four traffic sign classes, overlap

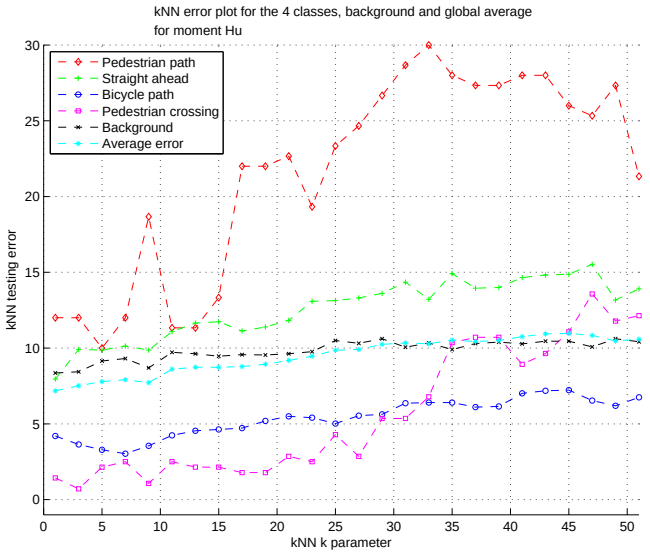


Figure 33: kNN plain Hu moments data 4 classes and background classification results

which grows with the introduction of the background patterns. The interpretation is that high k values involve a large number of considered examples which in turn are likely to be spread on a broader region; a growing region is more likely to intersect the classes overlap region which causes, so to say, voting confusion.

All in all these results are encouraging as kNN is a reference classifier for more sophisticated classifiers; more sophisticated classifiers are expected to outperform kNN.

13.2 NORMALIZED DATA

The data normalization does not improve the reference-based background sifting in a useful way as discussed in part III; to the contrary, often the sifting performance is degraded. It is not obvious that this tendency keeps on for more sophisticated classifiers and therefore the same experiments are to be repeated with the normalized data, which in this case is the IQR-normalized data.

The classification performance of kNN for the four class problem with the normalized Hu moments is shown in figure 34 on the next page; the performance for the five class problem with the normalized Hu moments is shown in figure 35 on the following page. The results derived from the other considered moments in the normalized version for the four and five class problems are shown in the appendix (figures 60 and 61).

The results for the four class problem are generally impressive: for all the moments the classification performance improves considerably and stays stable up to quite large k with respect to the plain data case, making the classification more consistent against the voting consensus, so to say.

The classification error after the normalization of the data seems to dive below 1%; only for the Flusser and Suk complex moments the classification error gets slightly worse.

These results indicate that normalization can improve the traffic sign discriminability; looking back at the pre-processing (part III), the impression is that improvements through data normalization could be possible despite the difficulties encoutered in the used approach; more knowledge about the data behavior is needed.

The results for the five class problem show a different picture: the classification curves still are smoother and stabler against a growing k, while the performance is slightly improved with respect to the plain data case. The only case where the normalization delivers slightly worse performance is with the Flusser and Suk complex moments; this is consistent with the plain data case.

All in all these first classification results are quite encouraging also with the normalized data: the traffic sign recognition problem shows to be learnable with quite good performances despite the simplicity of the kNN classifier, an encouraging starting point.

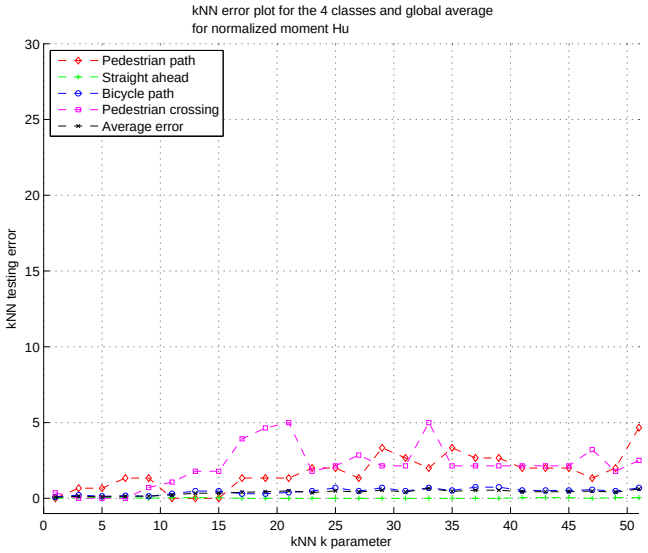


Figure 34: kNN normalized Hu moments data 4 classes classification results

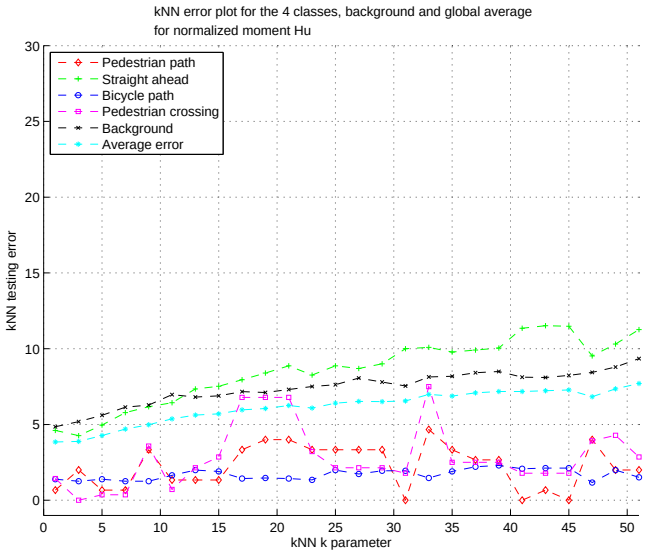


Figure 35: kNN normalized Hu moments data 4 classes and background classification results

14 CCM CLASSIFICATION

The Class-Conditional Means (CCM) are the class-wise means for the data points, the (invariant) moment vectors. The class-conditional mean can be interpreted as the mean vector of a set points belonging to a class; the idea is that such a mean point is able to represent the whole class.

Analogous to the kNN classification (see chapter 13), the classification performance is analyzed in the four and five class problems with plain data and normalized data. The analysis follow exactly the same mold: the performance is validated by averaging ten runs of the 10%/90% data set splitting scheme.

14.1 PLAIN DATA

The best classification results for the CCM classification on plain data are shown in figure 36 on the next page for the four class problem and in figure 37 on the following page for the five class problem. The complete results are not shown here; the careful reader might look in the appendix (chapter C) for the full overview.

In general it is clear that the CCM classifier generally performs poorly on the traffic sign data set. For any moment only a class is recognized correctly; for a different moment data set the correctly recognized class changes. Only on the Hu moments case the CCM classifier achieves an acceptable performance for two traffic sign classes.

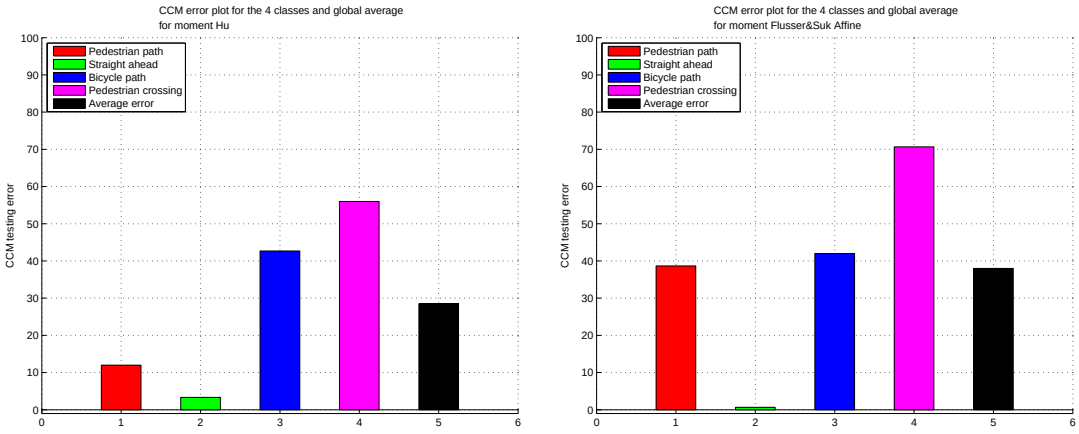
Also it is to notice that the introduction of the background data deteriorates the classification performance in all cases, analogous to observations done while pre-processing the data and while applying the kNN classifier. The reason for the dangling well-recognized class is unclear at this stage. The good news here is that for some moments it could be possible to find a simple representation for a class, the one correctly classified by the CCM method.

14.2 NORMALIZED DATA

Following the analysis pattern of previous section, in this section the classification results for the CCM classification on IQR-normalized data for the four class problem and for the five class problem are discussed.

Comparing these performances with the plain data performances, it is rather clear that the IQR-normalization does not improve the overall performances of the CCM classification: in some cases a class achieves a slight improvement with normalized data, in some other a slight impairment. The overall performance remains poor and in general it is likely to indicate a low information content for the class-conditional means: the class distributions probably are not hyperspherical (neither for the plain data nor for the normalized data) and therefore the mean is not a good class representative; a classification against this data-driven prototype does not work well, and this is consistent with the behavior shown by the use of the ideal prototypes (see chapter 12 on page 65).

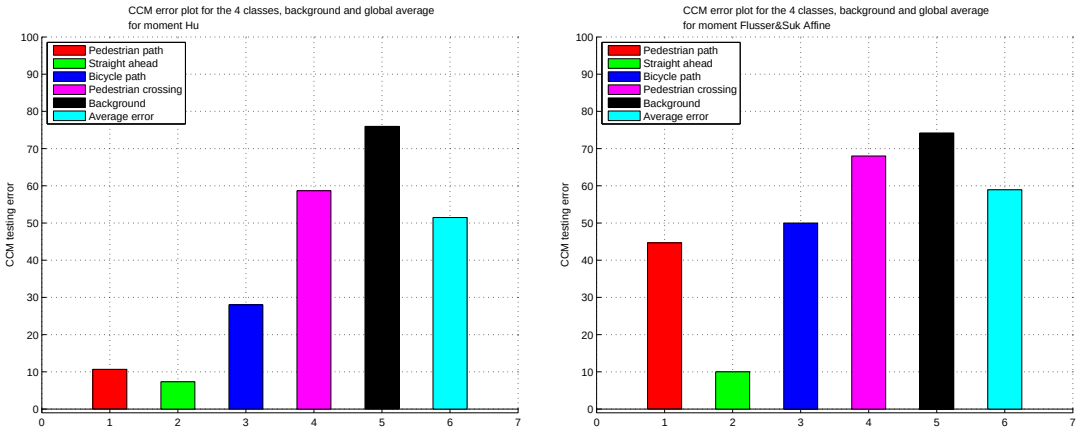
These results are not shown here; the careful reader can find an overview in the appendix in figure 64 on page 137 for the four class problem and in figure 65 on page 138 for the five class problem.



(a) Hu moments classification (b) Flusser and Suk Affine moments classification

Bar plots of the CCM classification error for the different moments; the average traffic sign classification error is plotted as well.
The complete results overview can be found in the appendix in figure 62.

Figure 36: CCM plain data 4 classes best classification results



(a) Hu moments classification (b) Flusser and Suk Affine moments classification

Bar plots of the CCM classification error with background data for the different moments; an average traffic sign and background classification error is plotted as well.
The complete results overview can be found in the appendix in figure 63.

Figure 37: CCM plain data 5 classes best classification results

15 LVQ1 CLASSIFICATION

In this chapter the Learning Vector Quantization (LVQ, introduced in section 6.3) is applied to the sampled traffic signs data set in order to obtain a classifier which performs automatic traffic sign detection. The aim here is to have the LVQ algorithm to *learn* the representation of the traffic sign classes from the training data; the derived representation is then expected to be general enough to allow the recogniion of the traffic signs among new unseen patterns (e.g. new images taken while driving). The classification performance and the generalization power are evaluated in the same way as for kNN in chapter 13 by applying ten-fold 10%/90% validation.

Different from kNN and CCM, LVQ knows an explicit training phase where example patterns from a training data set are offered in a random order for a certain *number of epochs* (#epochs). Moreover, the training on LVQ depends of the *number of prototypes* (kn) chosen for each data class and the *learning schedule* followed by the learning rate (η); see section 6.3 for more details about LVQ.

On beforehand, there is no obvious way to know which are sensible values for these training parameters. The usual way is to explore the parameter space empirically by trying a parameter setup, evaluating the performance and trying a new parameter setup with a slight change to one of the parameters; in the case of a better performance another change in the same "direction" can be evaluated as well, otherwise another parameter change might be tried. This can be seen as a search heuristic in the LVQ parameters space.

In order to get grip on the LVQ learning parameters it is useful to look at the evolution of the classification error during the learning; a plot of the classification error versus the learning epochs may give some insights about the quantity and quality of (the initialization of) the prototypes, the aggressiveness of the learning schedule.

A common problem in ML is *overfitting*, which is the situation where the learning algorithm focuses on casual features of the data set (while improving the learning error) instead of approximating the target function (evaluated by the testing error); this behavior for LVQ translates to an increasing performance during the learning while the testing performance at some point start to decrease (the symptom of a decreasing generalization power). A way to avoid overfitting is to apply *early stopping* by evaluating where overfitting (generally) occurs and limiting the learning epochs to that point; this way the learning process stops before the appearance of overfitting.

In order to see if the learner is in overfitting the evolution of the classification error on unseen data (the testing/validation error) is plotted together with the evolution of the classification error during the learning.

With LVQ different prototypes initializations can be applied, like the class-conditional means initialization (CCM), the random initialization (RND) and k-means initialization (kMEANS) (see section 6.3 for more details). In order to start from a simple scenario, initially the CCM initialization is used.

Some learning experiments are initially carried out to get acquainted with the LVQ learner; these experiments delivered rather bad results at a first sight (classification error percentages of 40% up to 80% for the traffic sign data). This seems to happen irrespective of the choice of the other parameters (kn/η/#epochs). A careful look at the learning setup reveals that the data reshaping previously

applied to kNN (chapter 13) is not sufficient for LVQ: the different class sizes are not taken into account as kNN can naturally cope with it.

For LVQ we have an explicit learning phase where examples are drawn and presented to the "winning" prototype (recall equation (6.2), the LVQ learning rule); classes with large size differences imply differences in the probability that an example of a certain class is drawn (the class priors). Very different priors have a negative effect on the training of the relative prototypes as the examples for a small class have a much smaller chance to be presented to a prototype of their class than the other examples. Such a lack of positive examples hinders a proper LVQ training.

For this reason a more thorough data reshaping is introduced for the LVQ classifier: now all the traffic sign classes are randomly sampled with the same number of examples (equal to the size of the smallest class, which has 155 examples), while the background class size still is reduced to the sum of all the traffic sign class sizes. The eventual reshaping now always results in 155 examples for each traffic sign class for the four class problem and an additional 620 background examples for the five class problem. This reshaping immediately leads to satisfying classification performance with LVQ.

A little trial and error reveals that values for the learning parameter

$$\eta \quad \text{in} \quad \left\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\right\} \tag{15.1}$$

seem to be effective with the Hu moments data; in particular 10^{-3} and 10^{-4} often give good results. The number of epochs is scaled with η as for a lower value of η a higher number of (similar) examples is needed by LVQ to adapt a prototype in a comparable way (recall the LVQ learning rule (6.2)). This interdependence in practice does not seem to be constant and the η values enumerated in (15.1) are best coupled with a inversely increasing number of epochs, e.g. like for the number of epochs parameter

$$\#\text{epochs} \quad \text{in} \quad \{100, 500, 1000, 5000\}; \tag{15.2}$$

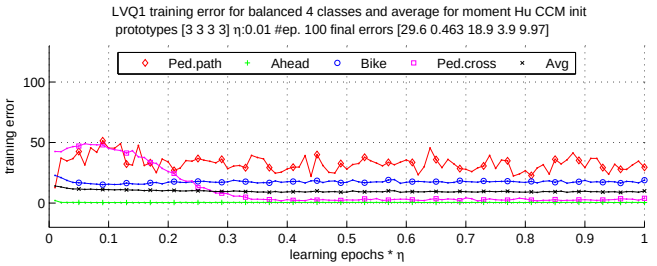
larger #epochs values seem not to lead to further learning. Some training plots for constant $k\eta$ and η are shown in figure 38 on the next page. Because of the relationship between η and #epochs, during this project normalized #epochs are used for the abscissa axis; the normalization happens by weighting each epoch by its corresponding η ($\text{epoch}(t) \cdot \eta(t)$ for time index t). The ordinate axis measures the classification error percentages.

Very often among the LVQ error curves a learning behavior is to observe where in the first few epochs the performance of all the classes improves relatively quickly, followed by a phase where the overall performance still improves but some classes experience a sudden increase in the error; finally very often these two phases are followed by a very long third phase where slow but steady classification improvement takes place (especially for the class which performance degraded during the second phase).

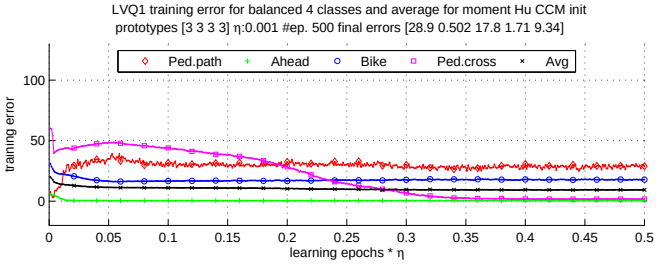
Similar to $k\eta$ and #epochs, adequate numbers of prototypes for the parameter $k\eta$ are not known in advance; in general this is problem dependent as it varies with the problem dimensionality, shape of the classes and the degree of overlap between the different classes. Keeping in mind that a real world application of a LVQ classifier for the traffic sign problem has also real-time computation constraints, it is interesting to look at relatively low numbers of prototypes per class, e.g. for

$$k\eta \quad \text{in} \quad \{1, 3, 5, 10, 15, 20\} \tag{15.3}$$

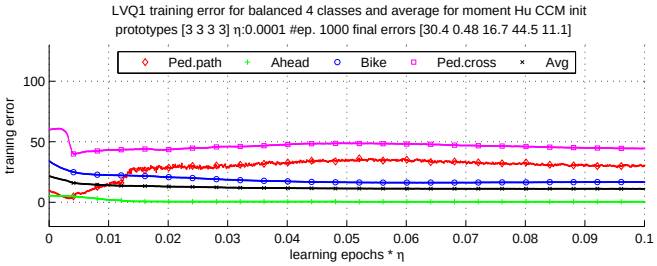
for each class. This is reasonable considering that in the case of two traffic sign classes the total number of examples is relatively small. For these two classes a



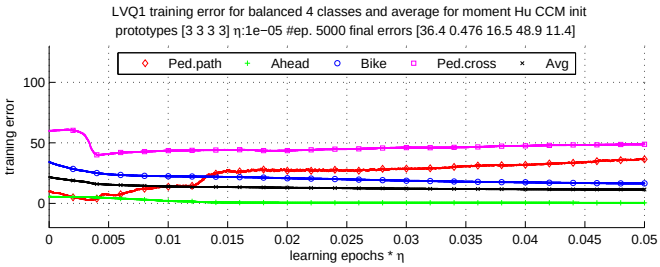
(a) Coarsest η



(b) Medium-coarse η



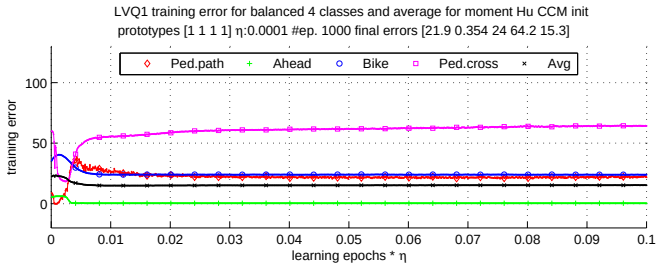
(c) Medium-fine η



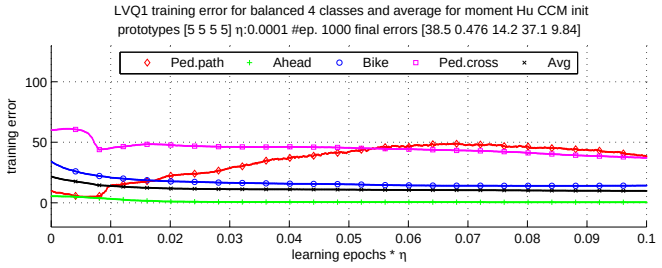
(d) Finest η

Training error plots for the LVQ1 learner. Four class problems, one prototype per class, CCM initialization, varying η and number of epochs. For lower η the number of epochs is not increased proportionally as no learning takes place.
Note how lower η leads to better performance; for the lowest η the initial error evolution of the four prototypes becomes very evident.
The relative testing error of these LVQ trainings is included in figure 66 on page 139

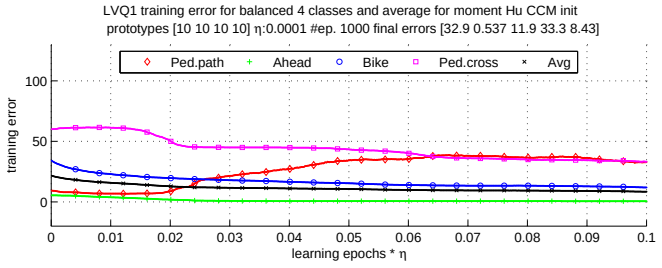
Figure 38: Some LVQ1 error evolution plots for varying η



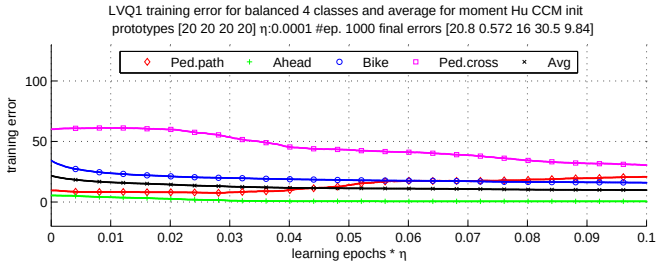
(a) Smallest $\mathtt{kn}$



(b) Medium-small $\mathtt{kn}$



(c) Medium-large $\mathtt{kn}$



(d) Largest $\mathtt{kn}$

Training error plots for the LVQ1 learner. Four class problems, varying number of prototypes per class, CCM initialization, $\eta = 0.0001$ and 1000 epochs.
Note how a larger number of prototypes leads to better performance; at the same time the initial error evolution of the four prototypes becomes increasingly stretched as well.
The relative testing error of these LVQ trainings is included in figure 67 on page 140

Figure 39: Some LVQ1 error evolution plots for varying number of prototypes

$k\pi$ of the same magnitude of the examples number would lead to (another form of) overfitting. Some examples of training error curves for constant #epochs and η are shown in figure 39 on the preceding page.

From these few training examples a new behavior is to infer: not only a link between the number of epochs and the learning rate is present, but the number of prototypes plays a role as well. For an increasing $k\pi$ the training and testing error curves get increasingly stretched, up to the point where the learning does not seem to complete within the given #epochs. The interpretation of this again requires to consider the LVQ learning rule of equation (6.2): the probability that an example is presented to a certain prototype decreases when the number of prototypes increases. Again the consequence is a poor or incomplete training, which can be compensated by tuning the parameters pair (η, #epochs). This is particularly evident when the number of prototypes increases by a factor ten or more.

Having sensible ($k\pi$, η, #epochs) triples, the different prototype initializations can be compared. Three initializations are mentioned in section (6.3):

$$k\pi_{\mathrm{init}} \quad \text{in} \quad \{\mathrm{CCM}, \mathrm{RND}, \mathrm{kMEANS}\} \tag{15.4}$$

where the CCM initialization consists of taking the mean of all the class examples and assign it to all the initial class' prototypes, the RND initialization consists of picking random class examples and assigning them to the the initial class' prototypes, and finally the kMEANS initialization consists of taking the kMEANS cluster centroids for k equal to the number of initial prototypes and use them as initial class' prototypes.

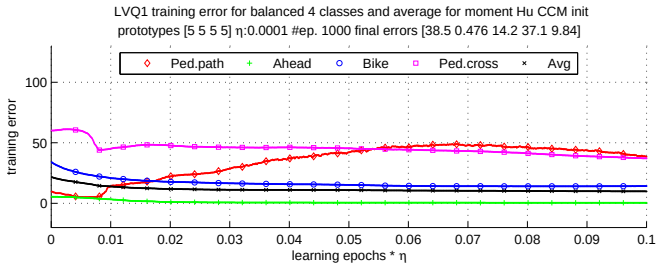
Some training plots for constant triples ($k\pi$, η, #epochs) and varying prototype initializations are shown in figure 40 on the following page. Here is to observe how a different initialization can add more information to the LVQ training and eventually boost the performance; in particular the CCM initialization performs relatively poor (which is consistent with the results of the previous chapter (14)), the random prototypes initialization performs better and the kMEANS initialization generally performs the best on the traffic sign data.

Analogous to the case of reference-based pre-processing (chapter 12), the parameter space grows explosively together with the parameters number; this makes an extensive study quickly infeasible. In order to reduce the parameters space, an *annealing* schedule is set up to "resume" the best capabilities of the explored (η, #epochs) pairs. The annealing (learning) schedule which closely reproduces the best results obtained so far is

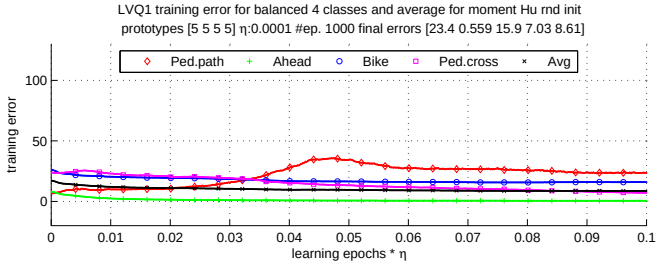
$$\left[10^{-2} \times 100_{\mathrm{ep}}, 10^{-3} \times 200_{\mathrm{ep}}, 10^{-4} \times 400_{\mathrm{ep}}, 10^{-5} \times 800_{\mathrm{ep}}\right] \tag{15.5}$$

which corresponds to a descending staircase of η's, where the quantities with a subscript ep indicate the #epochs to apply that particular η. Of course this does not solve the question of an optimal learning schedule as the problem is shifted to the choice of the annealing steps; the quest for the optimal (annealing) learning schedule is left out of the scope of this project. An example of the contribution of this annealing schedule is shown in figure 40 on the next page.

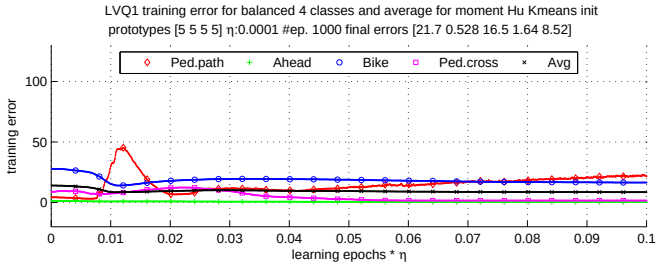
In general it seems that this annealing schedule is quite able in fitting the best (η, #epochs) parameters for most $k\pi$ configurations; the annealing schedule also seems able to flatten out the odd performance regressing phase. For the sake of parameter space reduction this annealing schedule seems adequate within this project.



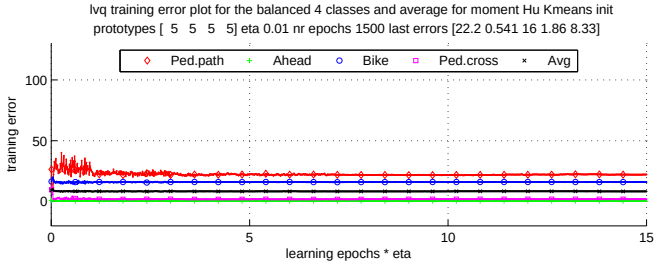
(a) CCM initialization



(b) Random initialization



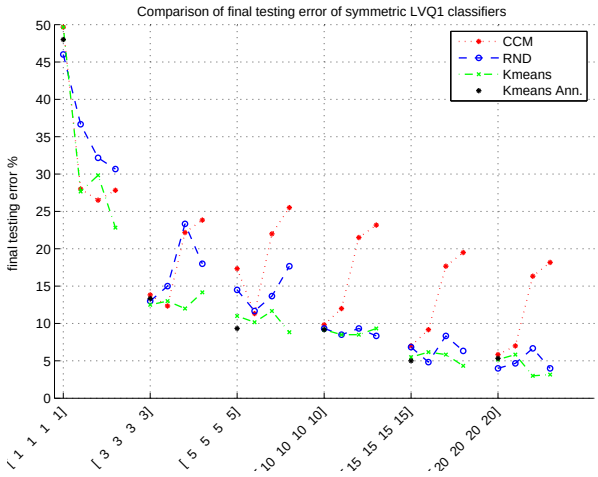
(c) k − Means initialization



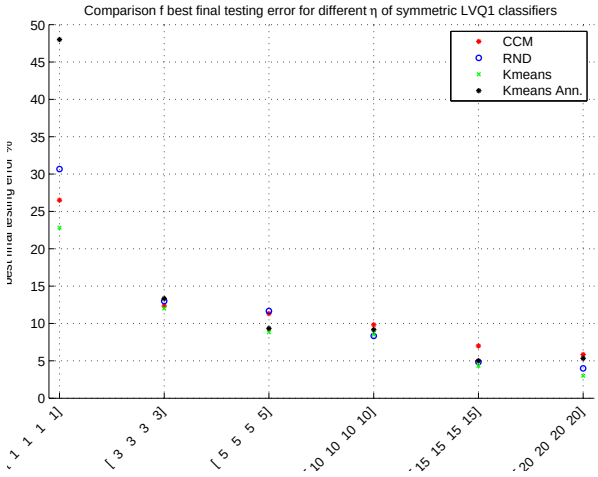
(d) k − Means initialization and annealing

Training error plots for the LVQ1 learner. Four class problems, five prototypes per class, $\eta = 0.0001$, 1000 epochs and varying initialization.
Note how better initialization leads to better performance and a steady learning curve.
The plot to the left on the bottom shows the behavior of the annealing schedule.

Figure 40: Some LVQ1 error evolution plots for varying initialization



(a) Symmetric prototypes configurations



(b) Best results symmetric prototypes configurations

Comparative LVQ classification performance plot by final average classification performance for the explored LVQ parameters for symmetric k_n configurations.
In the plot on the left different η values for the same (k_n,#epochs) tuple are grouped into curves. The plot on the right shows the best performance for each curve.
In the case of training with annealing schedule no curve is plotted; instead a single performance value is shown with an asterisk.

Figure 41: Comparative symmetric LVQ performance plots

15.1 PLAIN DATA

After this qualitative sampling of the parameter space, a close look to the classification performances on plain data is taken. The number of parameter triples (k_n, η, #epochs) to explore is now considerable and in order to compare the classification performances of all the learning experiments a synthetic representation for all the error plots is defined. To this end, the final average classification error, either the testing one or the training one, can be used as a resuming value. A new graph is now set up, where the classification error is plotted against the chosen k_n and where the points corresponding to performances of (k_n, η, #epochs) triples with a common k_n are linked into a curve. If some of the classification performance points do not look reliable, the corresponding training and testing error plots can be consulted to control the result, e.g. in the case of overfitting suspicions. The

performance graphs obtained in this way for the four class problem are shown in figure 41 on the preceding page.

A sight to these synthetic plots of classification performances let arise a number of considerations. First of all, the initialization effects are generally confirmed, where the initializations can be ordered by increasing performance as CCM $\prec$ RND $\prec$ kMEANS; in particulars, decreasing η (and consequently increasing #epochs) is useful only with RND and kMEANS initialization; with the CCM initialization the performance is generally worsened by a decrease of η. Second, indeed increasing the number of prototypes generally leads to better performances. Third, the prototypes also increase shifts the maximum performance point among (η, #epochs) parameter pairs to coarser learning rates. Last but not least, the annealing schedule confirms to closely match the best performance of the different ($k n_{init}, \eta$, #epochs).

A closer look to the training/testing error plots reveals that the kMEANS initialization not only improves the LVQ performance but also the learning; in fact the initialization alone already leads to a high detection performance. The LVQ training process improves the performance further but the improvement is often marginal. This confirms the good quality of the kMEANS initialization and therefore its high information degree.

Moreover, some classes show relatively poor recognition rate; in order to cope with this, a possibility is to set up asymmetric $k n$ configurations. The idea is to assign more prototypes to boost the classes which show weak performance; this can be effective in the case of complex class boundary shapes which are better approximated by a higher number of prototypes. This leads to the following base configuration: $k n = [4, 1, 2, 2]$ where the first class is boosted because it performs the worst on average, the second class is left untouched as it's performance is good on average, the third and fourth perform on average in between the other two and are boosted proportionally; also multiples of this configuration (i.e. $3 \times [4, 1, 2, 2]$ and $5 \times [4, 1, 2, 2]$) are considered.

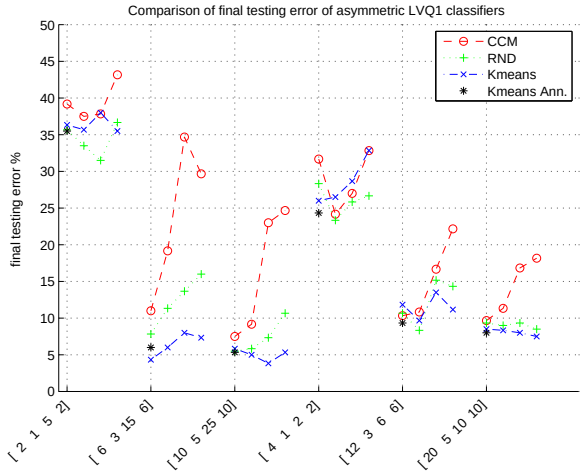
Another heuristic is to boost the classes with the largest "spread". Recalling the LVQ learning rule 6.2, it is clear the prototypes are driven by $\vec{\xi}$, the random drawn example in the LVQ notation. Depending on the spread of its size, denoted by $\langle \xi \rangle^2$, there may be more prototypes necessary to have a better cover of the topological region associated with the class' data[1]. By looking at the class-wise $\langle \xi \rangle^2$ values, it is to notice that the bicycle path has the highest spread, the drive ahead the lowest spread and the two pedestrian signs classes have a medium spread. Analogous to the previous heuristics this leads to $k n = [2, 1, 5, 2]$ and multiple configurations.

The performances obtained by applying these two heuristics are shown in figure 42 on the next page. Here is to see that both heuristics are quite able to reach good performance especially with the multiples of the base configurations, but the class data spread sensitive heuristics performs better than the poor-performance sensitive heuristics. Moreover, these asymmetric configurations seem able to mimic the performance achieved by the symmetric $k n$ configurations while using roughly the half of the total number of prototypes.

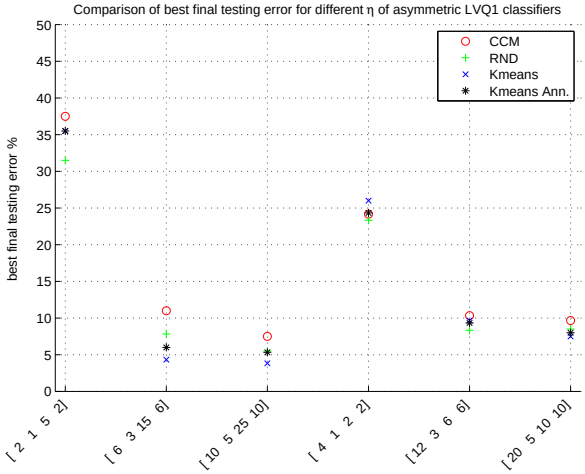
So far only the four class problem is analyzed; the real-world problem is troubled by the huge mass of background elements and therefore the five class problem is considered as well. The knowledge gathered about the ($k n, \eta$, #epochs) parameters is likely to be valid here too; from now on the only kMEANS initialization and the annealing schedule are applied.

The only added degree of freedom which is to be considered here is the number of prototypes for the background class. Analogous to the four class problem, there

1 The (scalar) value $\langle \xi \rangle^2$ is a shorthand for the inner product $\left\langle \vec{\xi} \right\rangle \cdot \left\langle \vec{\xi} \right\rangle$ where $\left\langle \vec{\xi} \right\rangle$ is the average example.



(a) Asymmetric prototypes configurations



(b) Best results asymmetric prototypes configurations

Comparative LVQ classification performance plot by final average classification performance for the explored LVQ parameters for asymmetric $k\eta$ configurations.
In the plot on the left different η values for the same ($k\eta$,#epochs) tuple are grouped into curves. The plot on the right show the best performance for each curve.
In the case of training with annealing schedule no curve is plotted; instead a single performance value is shown with an asterisk.

Figure 42: Comparative asymmetric LVQ performance plots

is no well-founded way to have an a priori estimate for this number of prototypes, therefore a little exploration is needed for this parameter.

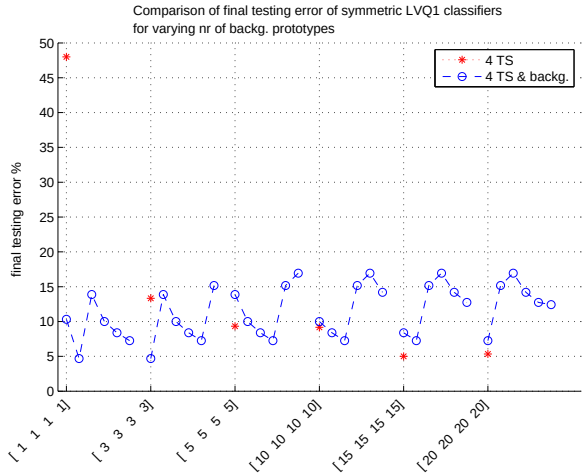
It is likely that the background data is spread everywhere around the four traffic sign classes, therefore a low number of prototypes (relative to the four TS classes) is unlikely to perform well. Recalling the thorough data set reshaping issue, the LVQ classifier training requires maintaining statistical properties on the data set and among the prototypes; for this reason it is also necessary to keep the number of background prototypes roughly to the same magnitude order of the total number of traffic sign prototypes (the data set contains as many TS examples as background ones). According to these considerations, six different numbers of prototypes are chosen for each four class $k\eta$ configuration, starting from the same number of prototypes assigned to a single class up to roughly twice the total of traffic sign prototypes (i.e. $[3,3,3,3]_{ts} \times [5,10,15,20,25,30]_{bg}$ and $[10,10,10,10]_{ts} \times [10,20,30,40,50,75]_{bg}$ where the subscript ts indicates the

numbers of prototypes for the traffic sign classes and the subscript $_{bg}$ indicates the numbers of prototypes for the background class).

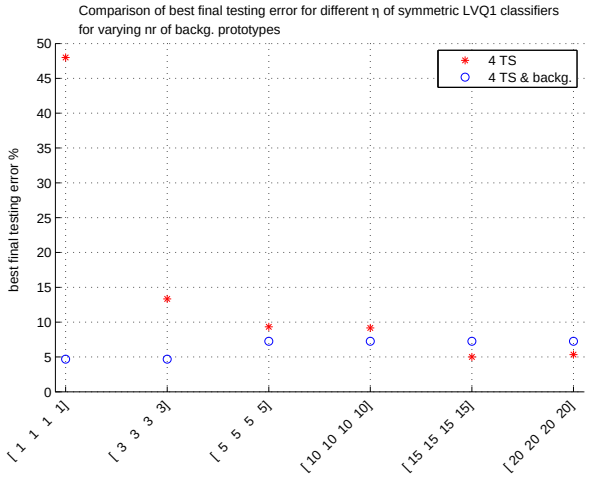
The obtained classification performances are shown in figure 43 on the facing page. Here it is possible to see a number of behaviors. First of all, the small four class k_n are greatly improved by adding background prototypes; this increase disappears for k_n with a larger number of TS prototypes. Second, all the performances roughly are in the $5\% \sim 15\%$ classification error band. Third, a complex performance behavior is paired with the increasing number of prototypes.

For small k_n it seems that increasing the number of background prototypes leads to better performance; this trend is interrupted by some kind of step where all of a sudden the classification performance get worse by 10%. This is to see only for the two smallest k_n.

Looking at larger k_n, again the increase of background prototypes leads to better performance; roughly around $30 \sim 40$ prototypes a performance decrease is visible again; this looks as overfitting by means of too many prototypes. A further increase in the number of prototypes then curiously leads to a new performance increase, for 75 background prototypes and above. This last part can be explained by recalling the 90%/10% validation scheme: from 620 background examples only 62 are used to validate the classification performance; this is a smaller number than the number of background prototypes. This situation is awkward and the corresponding results cannot be seen as reliable; the limits of this validation scheme for the used samples data set are clearly hit. Most likely a larger number of examples (and therefore images with the corresponding traffic sign) would overcome this, but with the present data set and the class-dependent reshaping it is not possible to go beyond.



(a) Symmetric prototypes configurations



(b) Best results asymmetric prototypes configurations

Comparative LVQ classification performance plot by final average classification performance for the explored LVQ parameters for symmetric k_n configurations in the four class problem against the five class problem.
In the plot on the left different background k_n values for the same (k_n,#epochs) pair are grouped into curves; the curves evolve from a low number of background prototypes (to the left) to a high number of background prototypes. The plot on the right show the best performance for each curve.

Figure 43: Comparative symmetric five class LVQ performance plots

Part V

DISCUSSION

The automatic traffic sign problem is analyzed, the data labeling is achieved, data pre-processing and classification is studied; in this last part all the previous results will be put together and finally the whole project will be discussed.

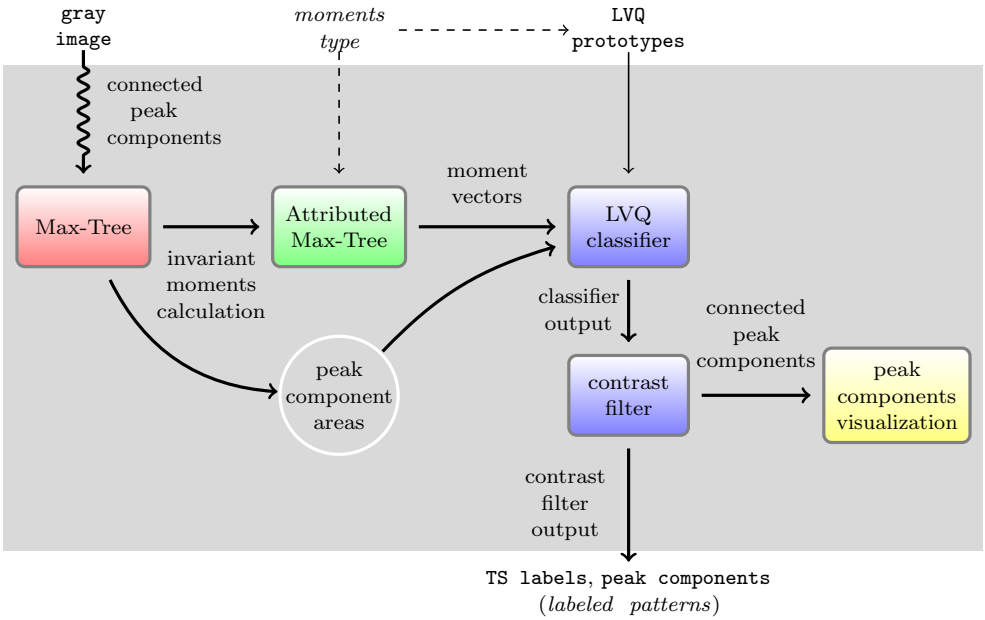
In the next chapter (16) first a demonstration prototype of an application for traffic sign recognition is presented; then in chapter 17 all the conclusions are summarized and finally the future work is discussed in chapter 18.

16 DEMONSTRATION

Once the LVQ classifier is trained and tuned as described in chapter (15), the natural procedure is to store the distilled prototypes; based on those prototypes it is straightforward to implement a classifier. Such a classifier simply follows the classification procedure highlighted in algorithm 6.3 on page 25.

In short, the classification takes place in the moment space; whenever a new image has to be processed to recognize traffic signs, the whole transformation pipeline up to the (invariant) moments has to be walked through. Only at this point the classifier can possibly detect the traffic sign among the peak components by evaluating their shape features.

This process is highly related to the transformation pipelines discussed in chapter 8; in particular the supervised contrast based labeling system of figure 12 on page 40 is the starting point to design a demonstration prototype application for automatic traffic sign recognition. The main reasons are that the supervised reference-based filtering is based on a very similar processing pipeline, and so does the contrast based variant of it. The LVQ-based automatic recognition is likely to show some degree of miss-classification of image entities; to counter this the contrast filtering of the contrast-based supervised system (see chapter 8) is likely to be very useful again.



The shaded box contains the processing stages of the automatic recognition system; on top of it the inputs are shown, below the outputs are shown. Thin arrows represent user driven parameter input, the snake arrow represents the recursive flood-fill procedure to decompose the image into peak components and the dashed arrow represents the future role of moment type choice; for this prototype only the Hu moments are used.
Note the similarity with the contrast-based supervised labeling system shown in figure 12 on page 40; the reference images input branch disappears as well as the input threshold parameter and the LVQ prototypes input appears instead. Also the human selection input and system feedback are removed.

Figure 44: The prototype-based recognition system.

The area information is never presented to the LVQ classifier during its training and therefore it reasonable to leave the user-driven area-based filtering into the system; this implies a marginal intervention from the user and does not affect the relevance of the automatic recognition system. The distance information as well as the reference traffic signs instead are removed from the system; their role is precisely filled in by the LVQ classifier, which substitutes them entirely. The human selection feedback is removed as well; the system is just meant to show the quality and usefulness of the recognition output.

The final processing pipeline for the automatic traffic sign recognition based on the LVQ classifier is shown in figure 44 on the preceding page.

The prototype application accepts any unseen image in the binary PGM file format; here the results for the testing image of figure 1 on page 3 are shown in figure 45 on the facing page. There the plain recognition output by means of the usual labeling visualization is shown, as well as the contrast-filtered output in the case of different minimal contrast levels.

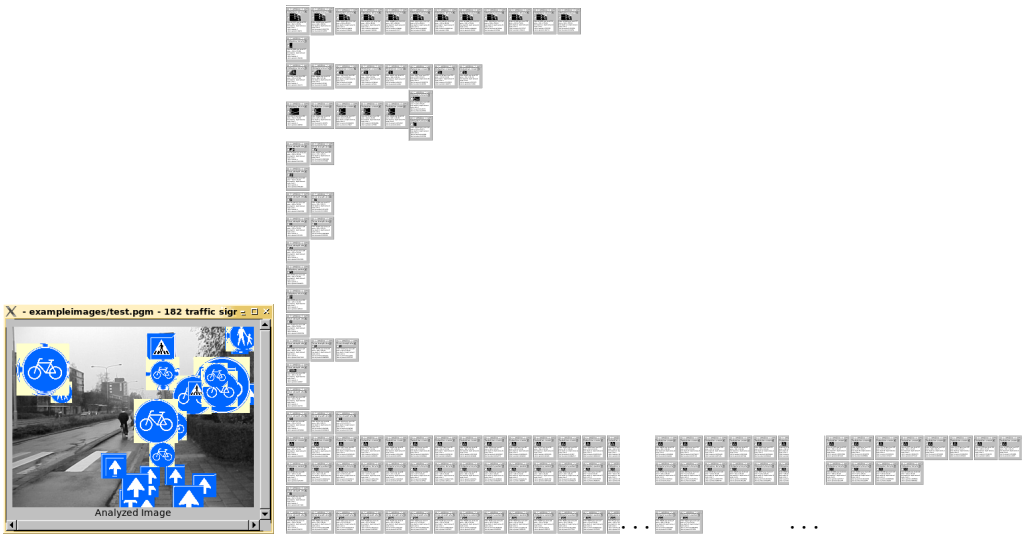
The plain output indeed shows a great deal of missclassified image entities: out of 182 hits the most are false positives. Looking further, the component tree structure shows the presence of many short sub-trees (most of them are just a single leaf) which correspond to a low contrast structure.

One of the conclusions deriving from the data labeling (chapter 9) is that traffic sign are visually salient also by means of a high contrast; for this reason it is perfectly reasonable to filter this recognition output in the very same way, by means of sub-tree contrast.

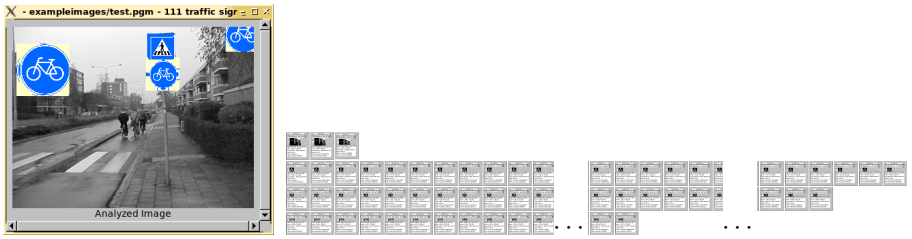
The effect of this contrast-based post-processing is directly evident even for a low minimal contrast level as 10; this minimal contrast level leads to 111 hits, of which most are true positives. The results improve further when raising the minimal contrast level to 20: the corresponding recognition output shows a highly simplified tree with only three chains of peak components, of which only one is made by false positives. The number of false positives decreases to two nodes for the minimal contrast level of 30, which eventually disappear for the minimal contrast level of 32 (not shown).

This qualitative example of LVQ-prototypes based traffic sign recognition in combination with contrast-based post-processing shows very good results. Trials with the larger images of the data set are done too; there the prototype application performs similarly. Because of the larger number of background peak components the recognition suffers from a larger number of false positives, which still are simplified by the contrast-based post-processing.

A drawback shown by this application prototype is the significant computing time increase: while the reference-based filtering takes a couple of seconds to perform, the LVQ-prototypes based recognition requires about a minute processing time. This is supposedly due to the increased amount of distance computations, which is roughly $\mathcal{O}\left(10^2\right)$ because of the number of used prototypes (140). This is not necessarily a problem as it is possible to train LVQ classifiers with a few tens of prototypes and with comparable recognition performance. Moreover, if a large number of prototypes are still required, a space-partitioning data structure like a kd-tree can be used to reduce the number of required distance computations.



(a) zero contrast threshold



(b) contrast threshold 10

(c) contrast threshold 20

(d) contrast threshold 30

(e) TP nodes

Automatic classification results for the traffic sign detection base on the LVQ prototypes. The prototype configuration is ten prototypes for each traffic sign class and hundred for the background class; the used minimal area is 150 and the maximal area percentage is 1%.
The basic detection results are quite polluted by false positives; the relative component tree is cluttered by many singleton nodes.
The application of the contrast filter (which is analogous to a voting scheme) dramatically improves the results already with the minimal "consensus" of ten "votes"; eventually the false positives are completely removed for the minimal contrast of 32 gray levels. Sub-figure 45e shows two of the true positive peak components.

Figure 45: The prototype-based recognition output.

17 CONCLUSIONS

In the scope of this thesis a number of goals were enunciated; here the results achieved within the automatic traffic sign recognition project are resumed.

The project goals are manifold and are conveniently grouped in four sections, the first three of them corresponding to each project stage and the last one for the overall conclusions.

17.1 LABELING

The first stage of this project regards the generation of a labeled data set of moment vectors; such vectors are the patterns offered to the learners. For this task the starting point was a set of road images; each of them was decomposed in peak components and evaluated through the (invariant) moments.

In order to label the moment vectors three concepts were applied: labeling by similarity to (arbitrary) ideal reference moments, labeling by reference moments with human supervision and, last but not least, labeling by reference moments with contrast filtering and human supervision.

The results from the plain labeling by similarity confirm the expectations. Some true positive peak components can be labeled as traffic sign but false positive background elements quickly appear and heavily pollute the labeling output. Eventually the impossibility to set up a pure labeled data set by this labeling method is evident.

The contribution of a human supervisor solves the labeling problem through inspection and correction; the pure labeled data set can be set up by the reference moments with human supervision labeling method. The burden of the peak components inspection/correction turns out to be very high; the number of true positive components to track and especially the number of false positive components to discard is very large. Other steps are needed to simplify the task to a viable work.

The huge amount of false traffic signs hits forms a true problem, so to say, the background discarding problem. Also this novel problem needs to be addressed.

During the labeling practice the traffic sign peak component revealed a remarkable behavior: those components form chains within the Max-Tree; such chains consistently represent the same traffic sign and span across several gray intensity levels.

This motivates the addition of contrast based filtering and leads to the labeling by reference moments with contrast filtering and human supervision; this method shows a strong simplification of the resulting Max-Tree and therefore of the whole labeling task. This is a very valuable result but the background discarding task still is demanding.

A couple of heuristics are introduced to simplify further the background discarding process: the minimal area ratio between accepted parent component and child component and the visualization of the discarded root component and leaves components in the output Max-Tree. The first one proves to remove small regions of pixels obviously belonging to the background while the second one improves the tracking of all the true positive peak components.

Their application with the labeling by reference moments with contrast filtering and human supervision results in an even more improved labeling procedure: the labeling task now still is time consuming but has become viable.

The labeling task is completed in the end. A *ground truth* moment data set is produced for 6 different moment types; this data set has a staggering size of 440GB of data for a total of 45 millions of patterns.

17.2 PRE-PROCESSING

The pre-processing task consists of reducing the number of data patterns, possibly by several magnitude orders, and reshape the data with respect to the class distributions.

The necessity for pre-processing arose in this project after the pure labeled data set was generated. The labeled data set turns out to be too large to be used as it is; moreover traffic sign patterns are relatively scarce against background patterns, mainly because of their huge number, and show quite different class shapes and sizes, so to say a strong class asymmetry. These facts together make the total examples data set not suitable for learning.

Data selection has to be performed; the peculiarity is that it has to be performed while avoiding to discard the valuable information present. Two methods were applied: area-based pre-processing and reference-based pre-processing.

The area-based method aims to discard peak components of relatively rather small size or relatively rather large size; in both cases such peak components are unlikely to be traffic signs or to be useful and clear traffic sign hits.

The reference-based method aims to remove examples which are very dissimilar and therefore metrically far away from the traffic signs; those examples are obviously unlikely to be traffic signs too. As reference elements the moments derived from the ideal traffic signs are used.

For both methods the removal rate is studied by means of ROC curves; both methods lead to the removal of roughly the half of the data.

This analysis was repeated for the reference-based method on all the considered moments and distances. Also class-wise IQR-based data normalization is applied before performing the reference-based pre-processing; this did not improve the background removal rate and at this stage the contribution of data normalization remains unclear.

Unfortunately the removal of roughly $3/4$ of the background patterns is insufficient for the learners: with the background patterns number in the $\mathcal{O}\left(10^8\right)$ order, such removal ratio is too little to lead to an examples data set of treatable size and suitable class distributions.

This forced to look for another solution, namely, random data sampling. In order to overcome the class size asymmetry an asymmetric sampling is performed, where all the traffic sign data is preserved and only 10^5 randomly sampled background patterns are kept. This way a data set is generated which is treatable for the learners by means of size and class distributions.

The pre-processing task is completed and from the total data set a sampled data set suitable for learning analysis is generated.

17.3 CLASSIFICATION

The third task is the classification task where starting from an examples data set a learner is trained; from this learner a classifier is extracted which will perform the traffic sign detection on unseen data.

Three classifiers are studied to tackle the learning task: k Nearest Neighbors (kNN), Class-Conditional Means (CCM) and Kohonen's Learning Vector Quantization (LVQ1).

Four variants of the classification problem were considered: the four class problem (only traffic sign examples), the five-class problem (traffic sign and background

examples) and both problems with IQR-normalized data (with a global IQR derived from the traffic sign examples).

The CCM method shows in all cases a poor classification performance; it is likely that just one or maybe none of the classes is hyper-spherical and therefore these classes in general cannot be adequately represented by the class mean value. This holds for all the four variants of the classification problem.

The kNN method shows in general a quite good classification behavior, for all the four problem fashions. In particular, the traffic sign broad IQR-normalization remarkably improves the recognition rate. The degrading performance for higher k indicates a lack of examples for the two small traffic sign classes as well as a considerable degree of overlap among the traffic sign classes and the background class as well. Larger k imply a larger voting region, which in turn make the consensus more sensitive and difficult to reach in the overlap ("conflictual") regions.

Moreover the learnability analysis through the kNN method reveals that in general a higher moment order leads to a better AUC performance; for this reason the learnability analysis can be simplified by restricting the study to the highest available moment order. The results also show that while different moment types may lead to different performances, different similarity metrics lead to similar results. Based on this fact the customary L_2 metric can be chosen to simplify further the analysis.

For the LVQ1 method only the Hu moments are considered; those moments lead to the best performance with kNN and therefore are chosen as the most relevant ones. LVQ1 needs a considerable work to find the optimal parameters (learning ratio, number of epochs, number of prototypes per class, initialization) but is easily able to hit an error rate of 5% and in some cases also of 3% on the four class problem, quite a good performance for plain data. The $k-MEANS$ initialization proves to be the most valuable, as well as the annealing learning schedule.

Also for the five class problem similar performances are achieved with an adequate number of background prototypes. Here overfitting appears for the first time, as well as the limits of the 90%/10% validation scheme, due to the high number of class prototypes and the high rate between the total number of prototypes (especially for the background class) and the total number of available validation patterns.

In the end it can be stated that the problem of recognizing the traffic signs by shape with a moment description is learnable with good recognition rates; in particular LVQ1 classifiers with reasonable trade-offs between recognition performance and classifier complexity can be developed.

17.4 OVERALL

The last task is to put the previous ones together; the image decomposition, component labeling, pre-processing and classification are linked in a seamless process and implemented within a proof-of-concept application.

In order to implement the automatic traffic sign recognition process, one of the classifiers is chosen, namely the LVQ1 classifier; this for memory and performance reasons. A demonstration program is realized exploiting the same code base developed for the implementation of the labeling application.

A qualitative evaluation of this demo application is positive: there is to see how the multiclass LVQ1 classifier detects the traffic signs. A number of false hits are detected too, spread over background parts of the whole image. This is easily overcome by applying the contrast filtering, the very same used in the labeling application (see section 9.3). The obtained evaluation result is consistent with the

labeling results: the application of a minimal contrast value of $20 \sim 30$ gray levels removes all the false traffic sign hits in the analyzed image.

The unexpected downside is the performance: the labeling application shows quite a good speed while it is dominated by the Max-Tree computation time and especially the moments generation time; the automatic recognition prototype instead is dominated by the classification and contrast filtering computation time which are remarkably larger than the other two computing steps. It is likely that this is due to the large number of used prototypes (140) which make the classification step computationally heavy. This is in any case a good result, considering that it is a proof-of-concept application; better recognition speed can be achieved by developing a more compact classifier.

Finally it can be stated that the automatic traffic sign recognition is achieved to a large extent; a working demonstration prototype is realized to show the potential of this recognition framework.

18 FUTURE WORK

Despite all the results collected so far in the automatic traffic sign recognition project, far more work appeared along the way together with new research questions.

Here the consequences for future research are stated and conveniently grouped in four sections, the first three of them corresponding to each project stage and the last one for the overall consequences and follow-up work.

18.1 LABELING

The Max-Tree decomposition is based on gray intensity connected components; a more effective decomposition could be performed using color information, see for instance [Naegel and Passat, 2009]. An effort in this direction is already taking place within the research carried out by Florence Tushabe[1]; some related work is already published in [Tushabe and Wilkinson, 2007].

The huge number of leaves and the skewed distribution of the connected components against the size turns out to be troublesome. In order to cope with this, alternative Max-Tree decompositions might be analyzed, like the k-flat zones technique put forward in [Ouzounis and Wilkinson, 2010] (a particular case of hyperconnectivities discussed in [Wilkinson, 2009b]) where a minimal contrast is required while flooding new Max-Nodes; this technique could help to reduce this Max-Tree size problem.

During the labeling it is observed that there is a relationship between the parent size and the child size of nested connected components; a heuristic is used to suppress branching singleton pixels. It is interesting to look at such relationships implied by the tree-structure more thoroughly, even by considering the moment vectors; this might lead to an effective way of clustering together the Max-Tree components belonging to the same chain. This kind of clustering could help in decomposing the Max-Tree in a set of component chains, allowing further data set simplifications.

Alternative decompositions to the Max-Tree can be considered as well. For instance, the dual Min-Tree focuses on image minima (the leaves here represent minimal instead of maximal image intensities). Also the binary partition tree can be considered, which focuses on a trade-off between processing and accuracy; different criteria can be used in order to represent the most interesting regions in the BPT. [Ouzounis and Wilkinson, 2010]

Also different tree simplification criteria might be worth considering. In this project the initial assumption was that the tree structure is not likely to be relevant and therefore the filtering choice was the attribute thinning; now that the tree structure shows its relevance, nonpruning strategies (like attribute openings) might turn out to beb appropriate as well. Moreover other pruning strategies like the max rule might be useful too. Also the application of the Viterbi algorithm to the simplification rules might have potential. An overview of the different component trees and simplification criteria is given in [Salembier and Wilkinson, 2009].

1 http://www.cs.rug.nl/~florence/

18.2 PRE-PROCESSING

The classifiers revealed that a traffic sign broad IQR-normalization improves the performance; this might be true also for the pre-processing and therefore a new look at this stage is recommended. Also the kMEANS initialization for LVQ1 revealed to contain a high degree of information; the kMEANS points might be used for pre-processing purposes as well.

In any case, to gain more insight in the data set is necerrary also to reduce its dimensionality where e.g. data dimensions have a little information content or a high noise level; to this end data dimensionality reduction analysis should be performed.

18.3 CLASSIFICATION

For classification purposes, only shape features are used so far, abstracting away gray intensity for invariance. Real pictures contain color information which can be included. Contrast information is used now as filter post-processing; this information is valuable also for a classifier and should be included too.

Within this research the focus laid on the LVQ learner with the Hu moments as feature vectors; the application of LVQ to the other moment data sets might be interesting. The performance superiority of the Hu moments is still far from proven and therefore a features ensemble is worth to study. The relative ensemble data set is likely to contain redundant features but this might not be harmful; in fact at this stage it is not clear how to reach the right trade-off between data and features redundancy on one hand and classification precision on the other hand.

A whole family of sophisticated LVQ classifiers is worth to try as well; in particular rLVQ which is able to find the most relevant dimensions in the data, together with GMLVQ which takes on and is able to learn a full data transformation. The use of relevances is worth to consider also for kNN with the Weinberger metric learning [Weinberger and Saul, 2009].

The classification of traffic sign data is in the end a one-class classification problem; this problem is analyzed in [Tax, 2001]. A new member of the LVQ family is likely to learn the traffic signs one-class problem better: the application of the brand-new *one-class* LVQ [Rozeboom, 2009] is potentially promising.

Also alternative learning schemes can be considered, like windowed LVQ or LVQ with examples selection where the learning scheme is adapted to consider examples next to the class boundaries, the examples which are likely to contribute with the highest amount of information. Also an idea could be the collection of patterns which are miss-classified and use them to tune the learning.

Finally, for a real-world application it is to check if all the traffic sign classes are reasonably discriminable.

18.4 OVERALL

The used set of moments is chosen based on the popularity in the literature; other moments can be interesting as well, like the pseudo-Zernike moments.

In general, it is not clear which moments or which combination of moments contain the most useful information for object detection, and which dimensions are the most relevant. Taking this further, the question is whether features invariance is useful anyway, or it is competing with discriminability (at least, at a certain point).

Also interesting is to study the use of new features, like non-compactness measures or topological measures regarding object cavities and holes.

In this project the ROC and the AUC are put forward as an effective way of comparing and evaluating classifiers and pre-processing steps; this should be applied also to kMEANS as a pere-processing step and LVQ as a classifying step. Moreover, in order to go beyond two-class problems and evaluate multiclass problems, an extension of the ROC to higher dimensions could be applied, where as an evaluation measure the AUC might be extended to the volume under surface or the hypervolume under hypersurface.

Looking at other problem domains, interesting application for this research framework can be the (well known at the Intelligent Systems group[2]) hand eczema recognition (see for instance the masters thesis of van de Wal [2007]) and the recognition of spinal vertebrae within medical image data (see for instance the Phd thesis of Sardjono [2007] and Purnama [2007]).

2 http://www.rug.nl/informatica/onderzoek/

M. Biehl, A. Ghosh, and B. Hammer. The dynamics of learning vector quantization. In M. Verleysen, editor, *Proceedings European Symposium on Artificial Neural Networks ESANN*, pages 13–18, Bruges, 2005a. d-side publications. (Cited on page 2.)

M. Biehl, A. Ghosh, and B. Hammer. Learning vector quantization: the dynamics of winner-takes-all algorithms. *Neurocomputing*, 2005b. submitted. (Cited on page 2.)

M. Biehl, B. Hammer, and P. Schneider. Matrix learning in learning vector quantization. Technical report, Department of Informatics, Clausthal University of Technology, 2006. (Cited on page 2.)

M. Biehl, R. Breitling, and Y. Li. Analysis of tiling microarray data by learning vector quantization and relevance learning. In *International Conference on Intelligent Data Engineering and Automated Learning IDEAL*, volume 4881, pages 880–889, Birmingham, UK, December 2007a. Springer Lecture Notes Computer Science LNCS. (Cited on page 2.)

M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007b. (Cited on page 2.)

M. Biehl, P. Pasma, M. Pijl, L. Sanchez, and N. Petkov. Classification of boar sperm head images using learning vector quantization. In *European Symposium on Artificial Neural Networks ESANN*, pages 545–551, 2008. doi: 10.1.1.90.1412. URL http://www.cs.rug.nl/~biehl/Preprints/boar.pdf. (Cited on page 37.)

M. Biehl, B. Hammer, P. Schneider, and T. Villmann. *Advances in Neural Information Paradigms*, volume 247, chapter Metric Learning for Prototype-based classification, pages 183–199. Springer Lecture Notes in Computer Science LNCS, 2009. (Cited on page 25.)

Fabio Bracci. Learning@max-tree: Extracting feature vectors data sets from morphological attribute filters for traffic signs detection. Research internship report, Institute of Mathematics and Computing Science, Faculty of Mathematics and Natural Sciences, University of Groningen, August 2008. (Cited on pages 2, 6, 30, 37, 42, and 43.)

E. J. Breen and R. Jones. Attribute openings, thinnings and granulometries. *Computer Vision and Image Understanding (CVIU)*, 64(3):377–389, 1996. (Cited on pages 9, 10, and 11.)

K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, and M. Biehl. Discriminative visualization by limited rank matrix learning. Machine learning reports, University of Leipzig, March 2008. URL http://www.uni-leipzig.de/~compint/mlr/mlr_03_2008.pdf. (Cited on page 2.)

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693. (Cited on page 23.)

Tom Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9777. (Cited on pages 26, 28, and 113.)

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006. ISSN 0167-8655. doi: DOI:10.1016/j.patrec.2005.10.010. URL http://www.sciencedirect.com/science/article/B6V15-4HV747X-1/2/c1653cca4db4e94215437a482fcbecbb. ROC Analysis in Pattern Recognition. (Cited on pages 26 and 28.)

J. Flusser and T. Suk. Pattern recognition by affine moment invariants. *Pattern Recognition (PR)*, 26:167–174, 1993. (Cited on pages 1, 18, and 19.)

J. Flusser and T. Suk. Construction of complete and independent systems of rotation moment invariants. In N. Petkov and M. A. Westenberg, editors, *Proc. Comput. Anal. Images Patterns 2003*, volume 2756 of *Lecture Notes in Computer Science (LNCS)*, pages 41–48, Groningen, The Netherlands, August 25-27 2003. (Cited on pages 1 and 18.)

A. Ghosh, M. Biehl, and B. Hammer. Dynamical analysis of LVQ type learning rules. In M. Cottrell, editor, *Proceedings of the 5th Workshop on Self-Organizing Maps WSOM*, pages 587–594, Paris, September 5-8 2005. Université de Paris I. (Cited on page 2.)

R. C. Gonzales and P. Wintz. *Digital Image Processing*. Addison-Wesley, second edition, 1987. (Cited on page 7.)

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, second edition, 2002. ISBN 0201180758. (Cited on page 7.)

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, third edition, 2006. ISBN 013168728X. (Cited on page 7.)

H.J.A.M. Heijmans. Theoretical aspects of gray-level morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 13:568–582, 1991. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.87343. (Cited on page 10.)

M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, 1962. (Cited on pages 1 and 18.)

Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974. (Cited on page iv.)

T. Kohonen. Improved versions of learning vector quantization. In *International Joint Conference on Neural Networks IJCNN*, volume 1, pages 545–550, San Diego, 1990. IEEE Computer Computer Society Press. (Cited on page 2.)

T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995. (Cited on page 2.)

T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, second edition, 1997. (Cited on pages 2 and 25.)

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 0070428077. URL http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html. (Cited on page 22.)

Benoît Naegel and Nicolas Passat. Component-trees and multi-value images: A comparative study. *Lecture Notes in Computer Science (LNCS)*, 5720:261–271, 2009. doi: 10.1007/978-3-642-03613-2_24. (Cited on page 105.)

G. K. Ouzounis and M. H. F. Wilkinson. Mask-based second generation connectivity and attribute filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29:990–1004, 2007. (Cited on page 7.)

G.K. Ouzounis and M.H.F. Wilkinson. Hyperconnected attribute filters based on k-flat zones. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010. URL http://www.cs.rug.nl/~michael/OuzounisWilkinsonPAMI2010fpage.html. (Cited on page 105.)

I Ketut Eddy Purnama. *Imaging the human spine using ultrasound: a preliminary study to follow scoliosis progression*. Dissertation, Faculty of Medical Sciences, University of Groningen, september 2007. URL http://irs.ub.rug.nl/ppn/304089931. (Cited on page 107.)

Tiemen Rozeboom. Seeing faces: Applying vector quantization to census data. Master's thesis, Institute of Mathematics and Computing Science, Faculty of Mathematics and Natural Sciences, University of Groningen, 2009. (Cited on page 106.)

P. Salembier and J. Serra. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on Image Processing (TIP)*, 4:1153–1160, 1995. (Cited on page 8.)

P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing (TIP)*, 7:555–570, 1998. (Cited on pages 1, 8, 9, 13, 14, 38, and 113.)

Philippe Salembier and Michael H. F. Wilkinson. Connected operators - a review of region-based morphological image processing techniques. *IEEE Signal Processing Magazine*, pages 136–157, November 2009. URL http://www.cs.rug.nl/~michael/IEEE_SPM_2009_Salembier_Wilkinson.pdf. (Cited on pages 7, 14, and 105.)

Tri Arief Sardjono. *Spinal X-ray image analysis in scoliosis*. Dissertation, Faculty of Medical Sciences, University of Groningen, september 2007. URL http://irs.ub.rug.nl/ppn/304104620. (Cited on page 107.)

P. Schneider, M. Biehl, and B. Hammer. Relevance matrices in learning vector quantization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks ESANN*, volume d-side, pages 37–43, Bruges, Belgium, April 2007a. (Cited on page 2.)

P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. Submitted, 2007b. (Cited on page 2.)

P. Schneider, F.-M. Schleif, T. Villmann, and M. Biehl. Generalized matrix LVQ for the analysis of spectral data. In M. Verleysen, editor, *European Symposium on Artifiacal Neural Networks ESANN*, volume d-side, pages 451–456, Bruges, Belgium, April 2008. (Cited on page 2.)

Jean Serra and Luc Vincent. An overview of morphological filtering. *Circuits Syst. Signal Process.*, 11(1):47–108, 1992. ISSN 0278-081X. doi: http://dx.doi.org/10.1007/BF01189221. (Cited on page 8.)

Jean C. Serra and Philippe Salembier. Connected operators and pyramids. volume 2030, pages 65–76. SPIE, 1993. doi: 10.1117/12.146672. URL http://link.aip.org/link/?PSI/2030/65/1. (Cited on page 8.)

P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999. (Cited on pages 2 and 3.)

Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. CL-Engineering, 2007. ISBN 049508252X. (Cited on page 1.)

David M. J. Tax. *One-class Classification - Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, June 2001. URL http://ict.ewi.tudelft.nl/~davidt/thesis.pdf. (Cited on page 106.)

Florence Tushabe and Michael. H. F. Wilkinson. Content-based image retrieval using shape-size pattern spectra. In *Proceeding for the Cross Language Evaluation Forum 2007 Workshop, ImageClef Track. 19*. University of Groningen, Cross-Language Evaluation Forum, September 2007. URL http://www.cs.rug.nl/~florence/TushabeCLEF2007.pdf. (Cited on page 105.)

E. R. Urbach and M. H. F. Wilkinson. Shape distributions and decomposition of grey scale images. IWI-report 2000-9-15, Institute for Mathematics and Computing Science, University of Groningen, 2001. (Cited on pages 1 and 21.)

E. R. Urbach and M. H. F. Wilkinson. Shape-only granulometries and grey-scale shape filters. In *Proceedings for the 5th International Symposium on Mathematical Morphology (ISMM 2002)*, pages 305–314, 2002. (Cited on pages 1 and 7.)

E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson. Connected rotation-invariant size-shape granulometries. In *Proceedings for the 17th International Conference on Pattern Recognition (ICPR 2004)*, volume 1, pages 688–691, 2004. (Cited on pages 1 and 21.)

E. R. Urbach, N. J. Boersma, and M. H. F. Wilkinson. Vector-attribute filters. In *Proceedings for the International Symposium on Mathematical Morphology (ISMM 2005)*, pages 95–104, Paris, 18-20 April 2005. (Cited on pages 1, 7, 21, and 31.)

E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson. Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29:272–285, 2007. (Cited on pages 1, 7, 21, 31, and 39.)

Barend van de Wal. Automatic classification of hand dermatitis. Master's thesis, Institute of Mathematics and Computing Science, Faculty of Mathematics and Natural Sciences, University of Groningen, March 2007. URL http://irs.ub.rug.nl/dbi/4b4c31989f446. (Cited on page 107.)

Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, February 2009. doi: 10.1.1.117.5831. (Cited on page 106.)

Michael H. F. Wilkinson. An axiomatic approach to hyperconnectivity. *Lecture Notes in Computer Science (LNCS)*, 5720:35–46, 2009a. doi: 10.1007/978-3-642-03613-2_4. (Cited on page 7.)

Michael H.F. Wilkinson. Hyperconnectivity, attribute-space connectivity and path openings: Theoretical relationships. *Lecture Notes in Computer Science (LNCS)*, 5720:47–58, 2009b. doi: 10.1007/978-3-642-03613-2_5. (Cited on page 105.)

LIST OF FIGURES

Figure 1 example of urban road picture 3
Figure 2 Pixel adjacencies. 7
Figure 3 Example of connected and attribute openings. 11
Figure 4 Max-Tree construction procedure; figure taken from Salembier et al. [1998]. 13
Figure 5 Discrete ROC curve example. Figure taken from [Fawcett, 2004]. 28
Figure 6 Typical input images. 32
Figure 7 Problematic input images. 33
Figure 8 The target traffic signs. 35
Figure 9 THE IDEAL LABELLING SYSTEM. 37
Figure 10 THE REFERENCE-BASED LABELLING SYSTEM 38
Figure 11 THE SUPERVISED LABELLING SYSTEM 39
Figure 12 THE CONTRAST-BASED SUPERVISED LABELING SYSTEM. 40
Figure 13 THE DATA SET PRODUCTION SYSTEM 41
Figure 14 The image window 42
Figure 15 The about window 42
Figure 16 The Max-Tree window 43
Figure 17 The control window 44
Figure 18 An impression of the labeling application 45
Figure 19 Reference-based labeling output. 50
Figure 20 Supervised labeling output 52
Figure 21 Contrast-based Supervised labeling output. 54
Figure 22 Examples of too small traffic signs 55
Figure 23 Examples of ignored traffic signs 56
Figure 24 Area histogram computation scheme 61
Figure 25 Class-wise area histograms 63
Figure 26 Distance histogram computation scheme 65
Figure 27 Data set sampling scheme 67
Figure 28 An example of target class and background histograms and relative ROC curve 68
Figure 29 Examples of ROC curves 69
Figure 30 Synthetic overview AUC 70
Figure 31 Synthetic overview area below ROC normalized 73
Figure 32 kNN plain Hu moments data 4 classes classification results 78
Figure 33 kNN plain Hu moments data 4 classes and background classification results 78
Figure 34 kNN normalized Hu moments data 4 classes classification results 80
Figure 35 kNN normalized Hu moments data 4 classes and background classification results 80
Figure 36 CCM plain data 4 classes best classification results 82
Figure 37 CCM plain data 5 classes best classification results 82
Figure 38 Some LVQ1 error evolution plots for varying η 85
Figure 39 Some LVQ1 error evolution plots for varying number of prototypes 86

Figure 40 Some LVQ1 error evolution plots for varying initialization 88
Figure 41 Comparative symmetric LVQ performance plots 89
Figure 42 Comparative asymmetric LVQ performance plots 91
Figure 43 Comparative symmetric five class LVQ performance plots 93
Figure 44 The prototype-based recognition system. 97
Figure 45 The prototype-based recognition output. 99
Figure 46 Synthetic overview AUC for 99% true positives acceptance 119
Figure 47 Synthetic overview AUC for 90% true positives acceptance 120
Figure 48 Minimum discarded background for 100% true positives acceptance 121
Figure 49 Minimum discarded background for 99% true positives acceptance 122
Figure 50 Minimum discarded background for 90% true positives acceptance 123
Figure 51 Histogram and ROC examples part two for reference moments. 124
Figure 52 Histogram and ROC examples part one for reference moments. 125
Figure 53 Synthetic overview AUC for 99% iqr-normalized true positives acceptance 126
Figure 54 Synthetic overview AUC for 90% iqr-normalized true positives acceptance 127
Figure 55 Minimum discarded background for 100% iqr-normalized true positives acceptance 128
Figure 56 Minimum discarded background for 99% iqr-normalized true positives acceptance 129
Figure 57 Minimum discarded background for 90% iqr-normalized true positives acceptance 130
Figure 58 kNN plain data 4 classes classification results 131
Figure 59 kNN plain data 4 classes and background classification results 132
Figure 60 kNN normalized data 4 classes classification results 133
Figure 61 kNN normalized data 4 classes and background classification results 134
Figure 62 CCM plain data 4 classes classification results 135
Figure 63 CCM plain data 5 classes classification results 136
Figure 64 CCM normalized data 4 classes classification results 137
Figure 65 CCM normalized data 5 classes classification results 138
Figure 66 Some LVQ1 error evolution plots for varying η 139
Figure 67 Some LVQ1 error evolution plots for varying number of prototypes 140
Figure 68 Some LVQ1 error evolution plots for varying initialization 141

LIST OF TABLES

Table 1 Classification confusion matrix 27
Table 2 Distribution of images with traffic signs. 34
Table 3 The system's parameters 40
Table 4 Data set file extensions 47
Table 5 Data set file snippet 47
Table 6 Index file snippet 47
Table 7 Distribution of traffic sign patterns 57
Table 8 Data set sizes 57

LIST OF ALGORITHMS

3.1 Flooding procedure for the Max-Tree creation 15
3.2 Adapted Flooding procedure for the Max-Tree creation 16

6.1 k Nearest Neighbors rule 24
6.2 Class-Conditional Means rule 24
6.3 Learning Vector Quantization rule 25

8.1 Contrast filter 44

90/10 **splitting** Scheme to randomly split a data set in two, one with 90% data and the other with the remaining 10%

AUC Area Under (ROC) Curve

ROC Receiver Operator Characteristics

AMI Affine Moment Invariants

CM Central Moments

CMI Complex Moment Invariants

COTS Common off the shelf

IA Image Analysis

ML Machine Learning

NCM Normalized Central Moments

PA Pattern Recognition

PGM Portable Graymap Pictures

PNG Portable Network Graphics

AMT Attributed Max-Tree

CCM Class-Conditional Means

FLTK Fast Light Tool-Kit

GUI Graphical User Interface

IQR Interquartile Range

KMEANS k clustering means vectors

KNN k Nearest Neighbors

LVQ Learning Vector Quantization

MT Max-Tree

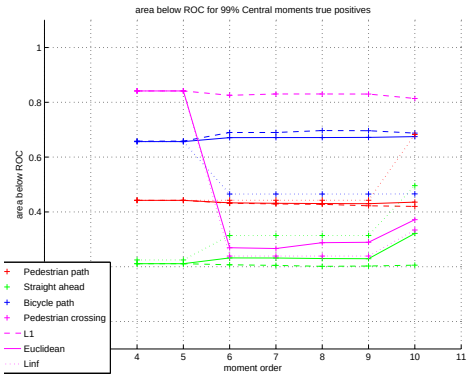
Part VI

SUPPLEMENTARY MATERIAL

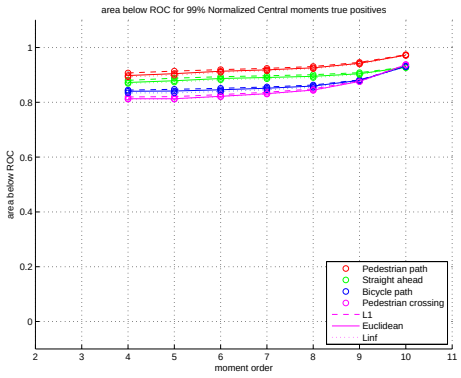
A

ADDITIONAL PRE-PROCESSING GRAPHS

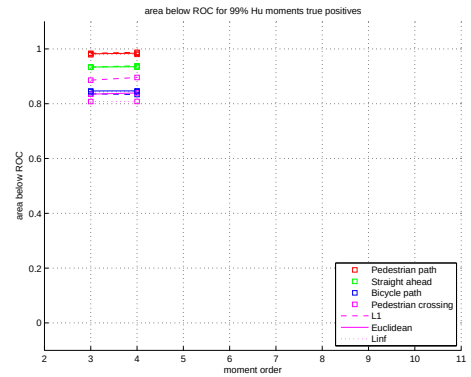
A.1 PLAIN DATA GRAPHS



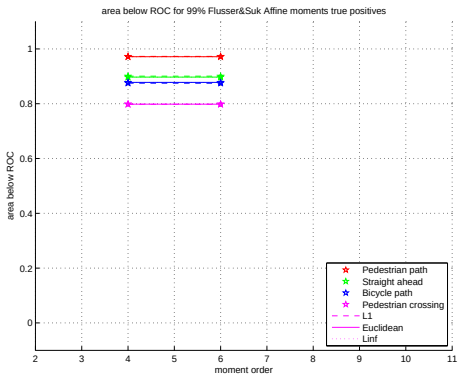
(a) Area below Central moment ROC's



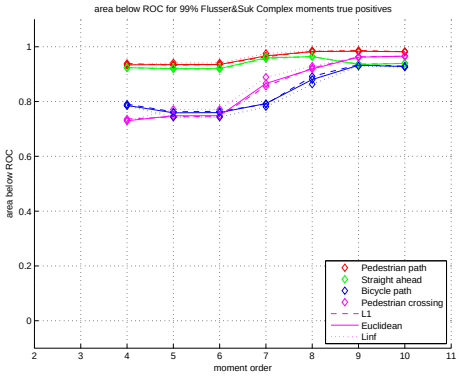
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



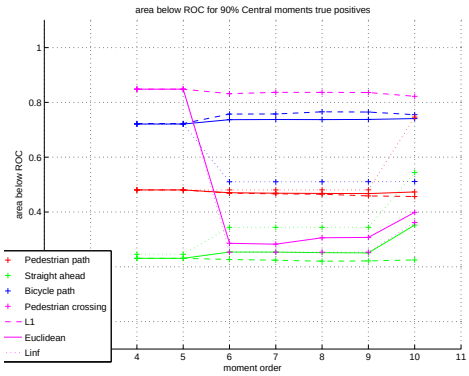
(d) Area below Flusser&Suk affine moment ROC's



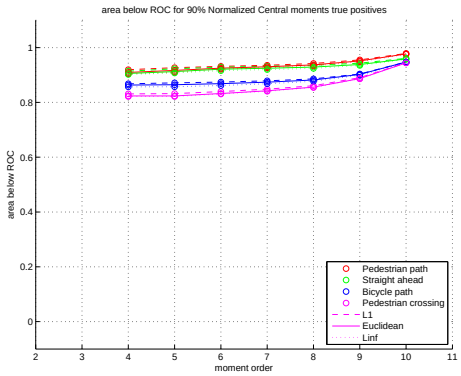
(e) Area below Flusser&Suk complex moment ROC's

Area below ROC for 99% traffic sign acceptance with plain data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

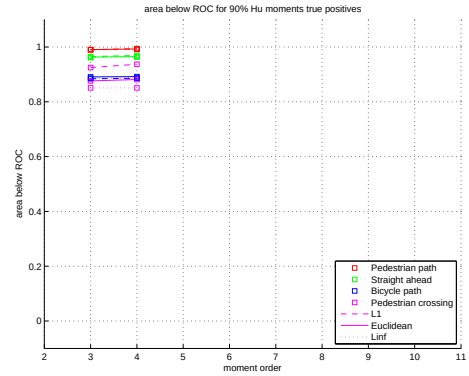
Figure 46: Synthetic overview AUC for 99% true positives acceptance



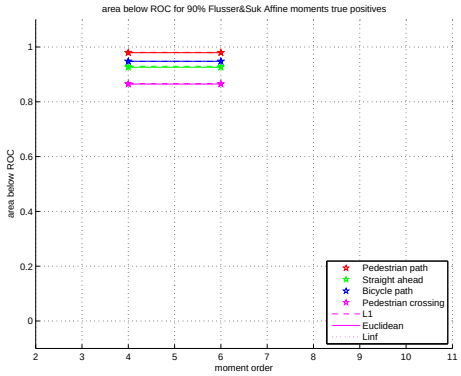
(a) Area below Central moment ROC's



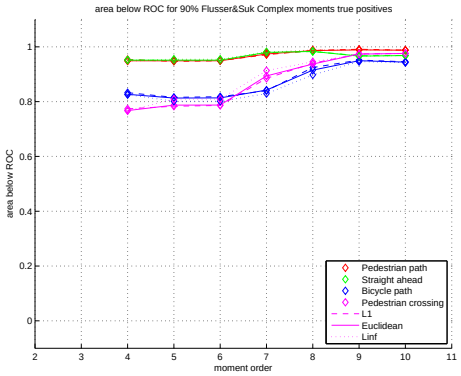
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



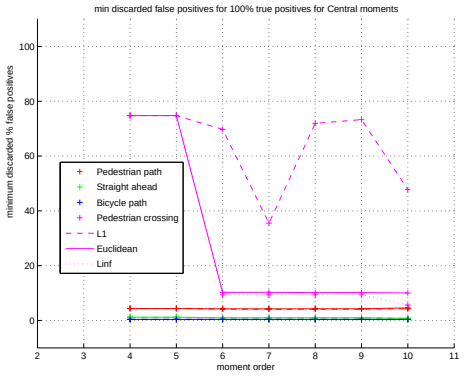
(d) Area below Flusser&Suk affine moment ROC's



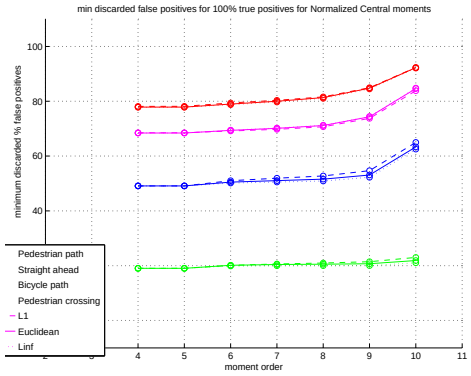
(e) Area below Flusser&Suk complex moment ROC's

Area below ROC for 90% traffic sign acceptance with plain data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

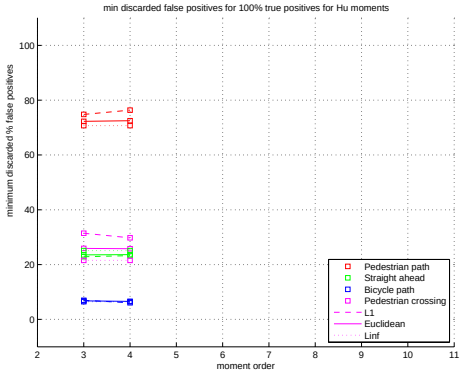
Figure 47: Synthetic overview AUC for 90% true positives acceptance



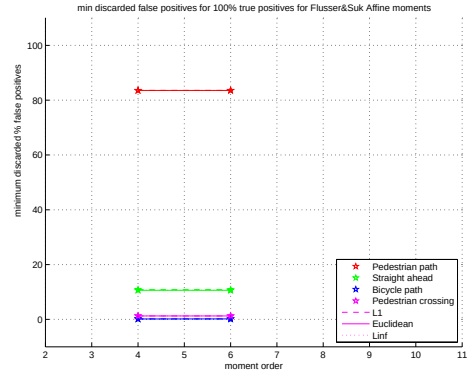
(a) Discarded Central moment background



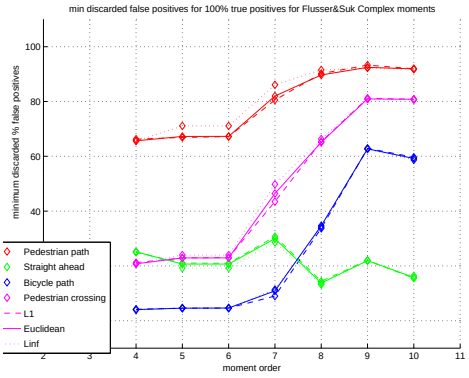
(b) Discarded Normalized Central moment background



(c) Discarded Hu moment background



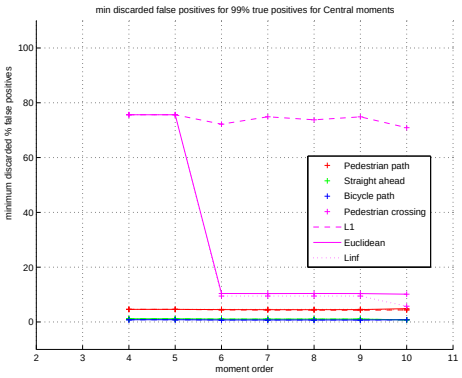
(d) Discarded Flusser&Suk affine moment background



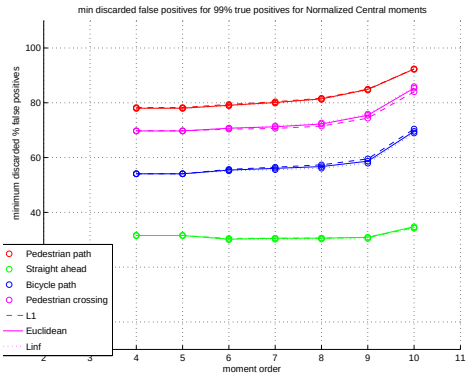
(e) Discarded Flusser&Suk complex moment background

Minimum discarded background data for 100% traffic sign acceptance with plain data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

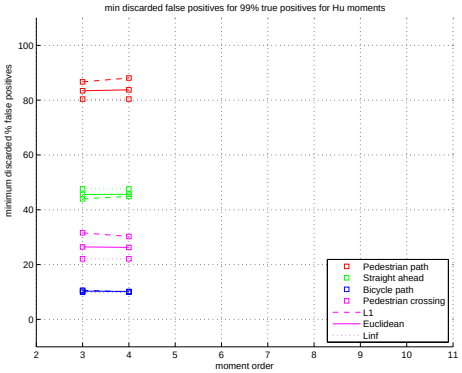
Figure 48: Minimum discarded background for 100% true positives acceptance



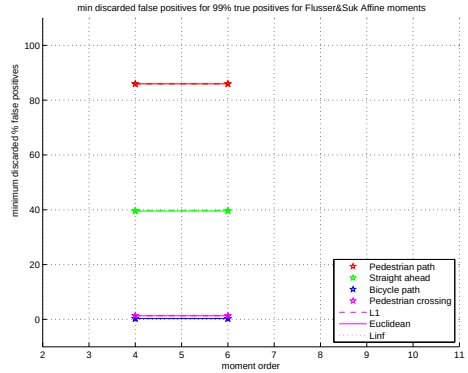
(a) Area below Central moment ROC's



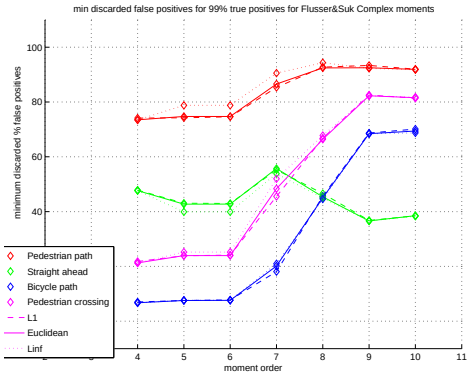
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



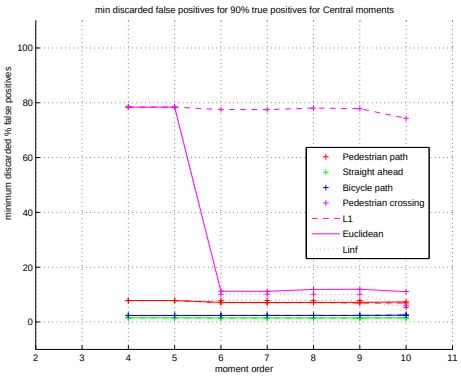
(d) Area below Flusser&Suk affine moment ROC's



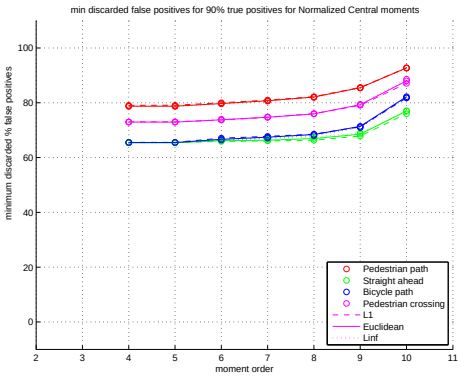
(e) Area below Flusser&Suk complex moment ROC's

Minimum discarded background data for 99% traffic sign acceptance with plain data. Different point markers correspond to different traffic sign classes, different line patterns correspond to different distances.

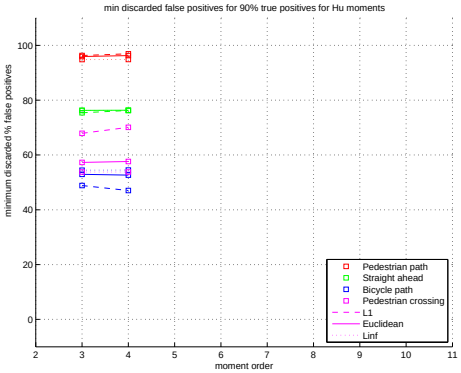
Figure 49: Minimum discarded background for 99% true positives acceptance



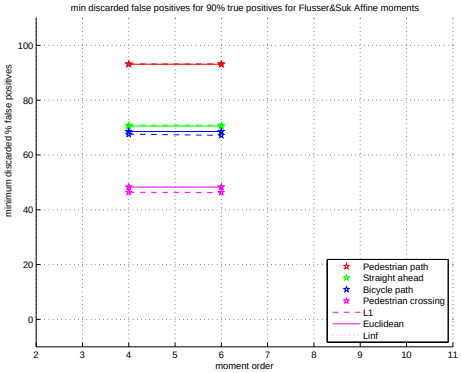
(a) Area below Central moment ROC's



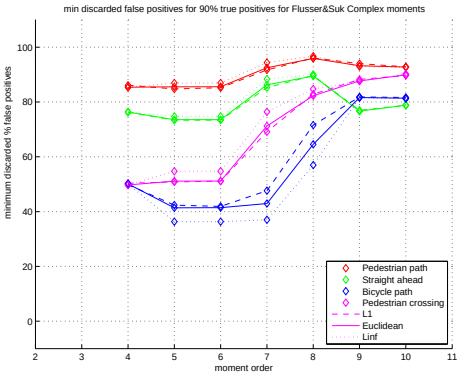
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



(d) Area below Flusser&Suk affine moment ROC's



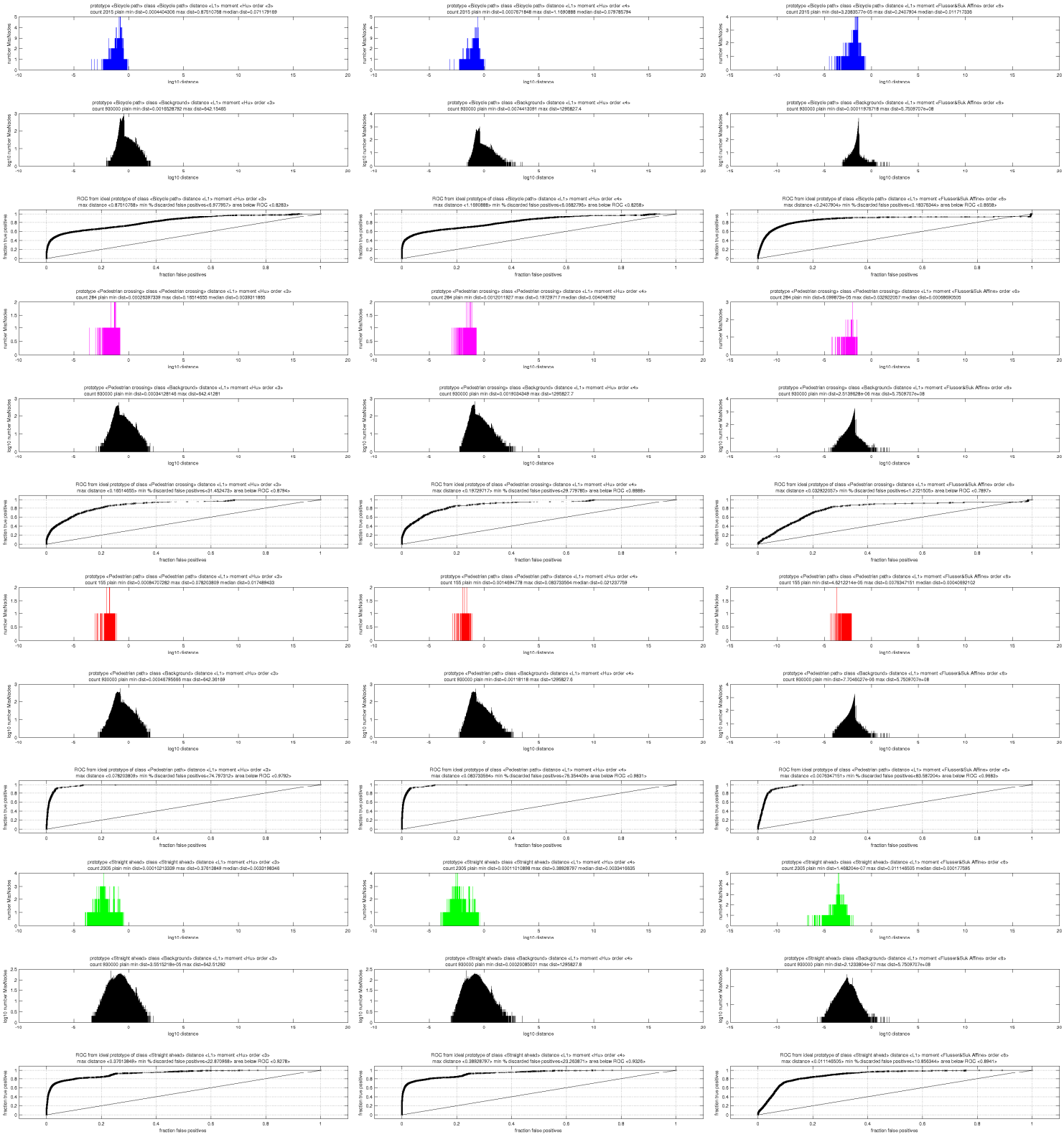
(e) Area below Flusser&Suk complex moment ROC's

Minimum discarded background data for 90% traffic sign acceptance with plain data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

Figure 50: Minimum discarded background for 90% true positives acceptance

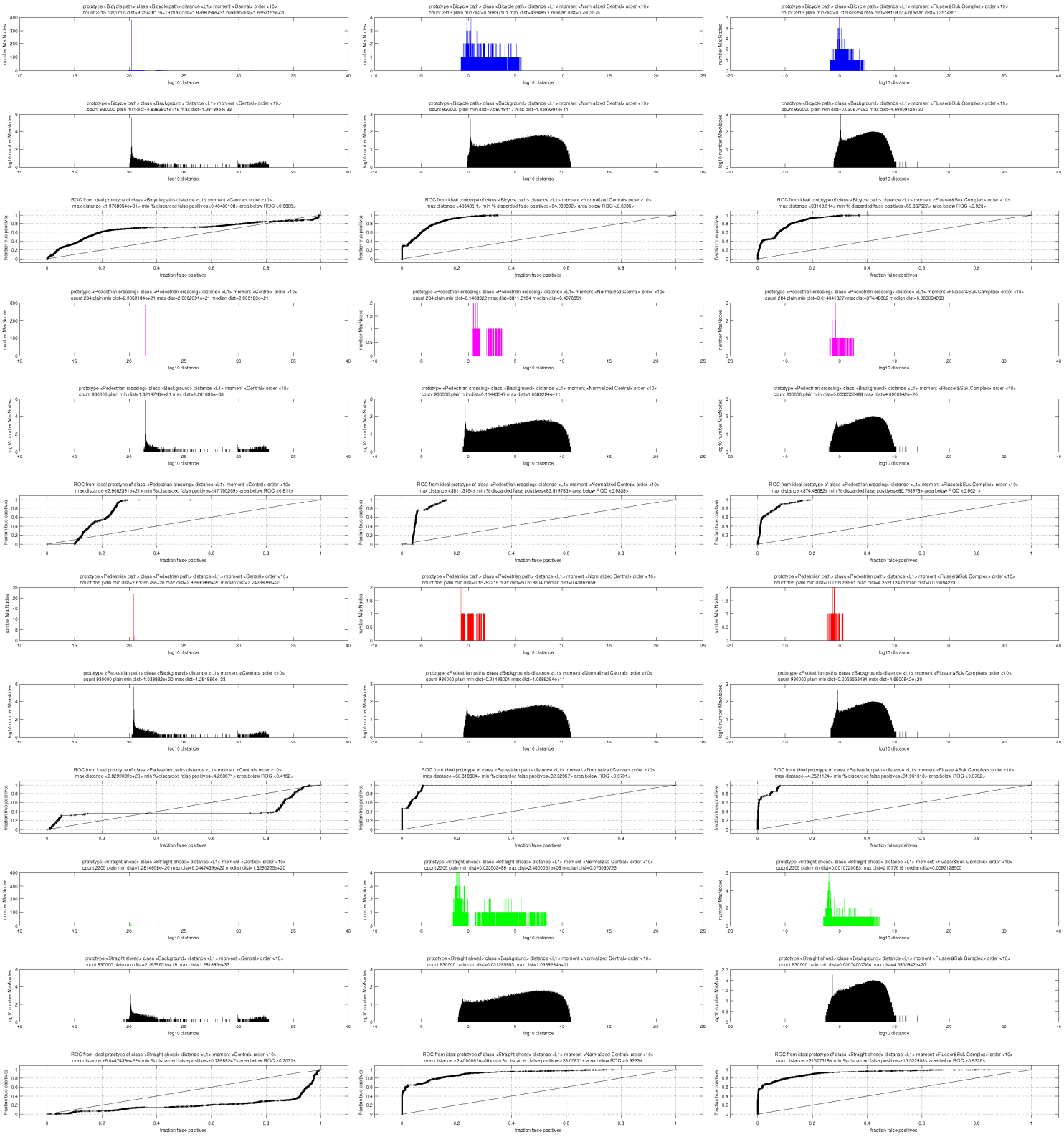
A.2 ROC AND CLASS HISTOGRAMS EXAMPLES

Figure 51: Histogram and ROC examples part two for reference moments.



(a) Hu moments 3rd order (b) Hu moments 4th order (c) Flusser and Suk moments 6th order

Some examples of histogram plots of the distances for traffic sign class and background class data. The bottom subplot is the ROC curve for the reference moment as a receiver.

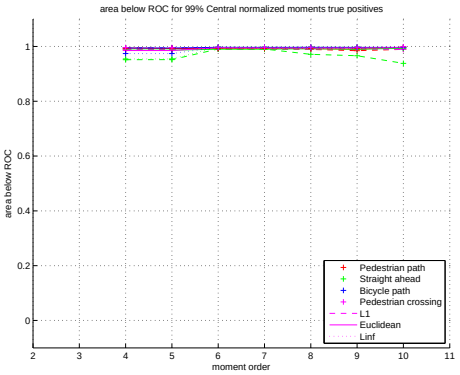


(a) Central moments 10th order (b) Normalized Central moments 10th order (c) Flusser and Suk moments 10th order

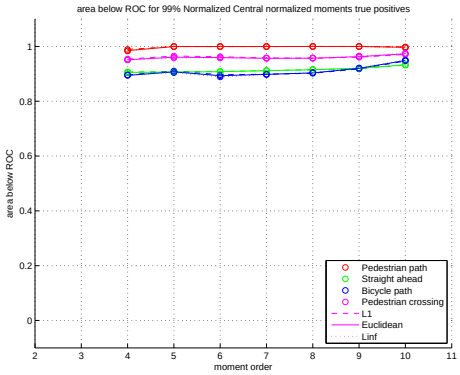
Some examples of histogram plots of the distances for traffic sign class and background class data. The bottom subplot is the ROC curve for the reference moment as a receiver.

Figure 52: Histogram and ROC examples part one for reference moments.

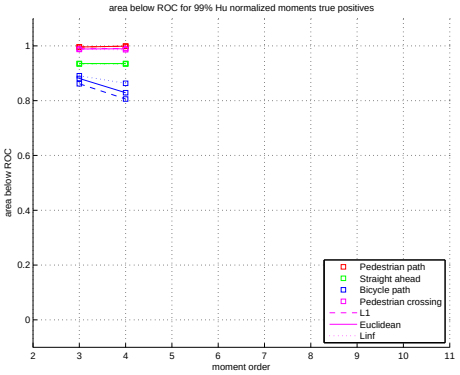
A.3 IQR-NORMALIZED DATA GRAPHS



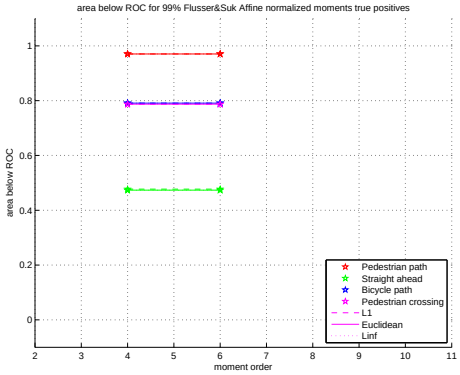
(a) Area below Central moment ROC's



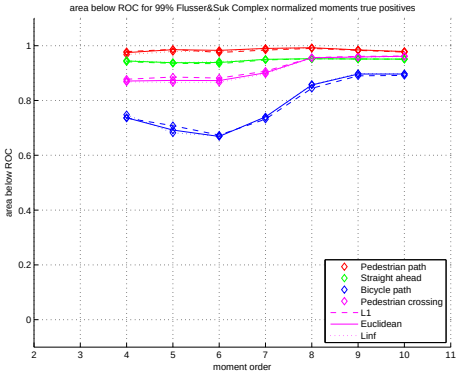
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



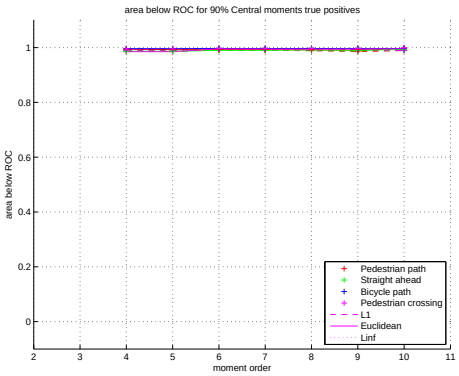
(d) Area below Flusser&Suk affine moment ROC's



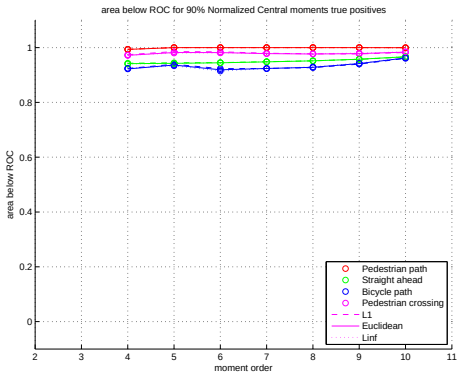
(e) Area below Flusser&Suk complex moment ROC's

Area below ROC for 99% traffic sign acceptance with iqr-normalized data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

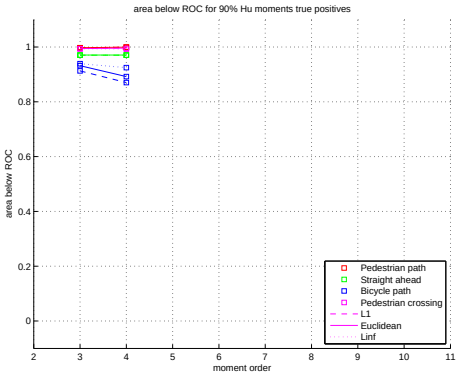
Figure 53: Synthetic overview AUC for 99% iqr-normalized true positives acceptance



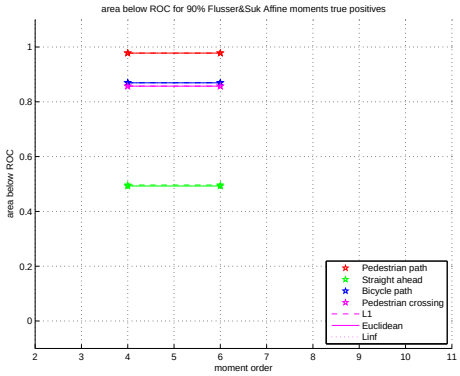
(a) Area below Central moment ROC's



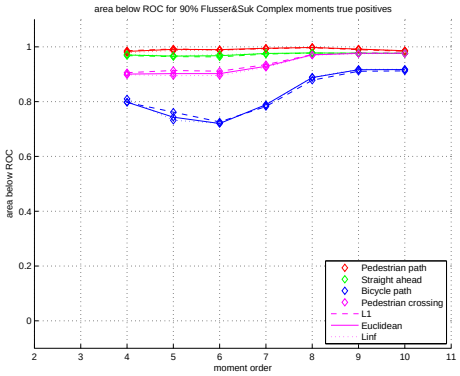
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



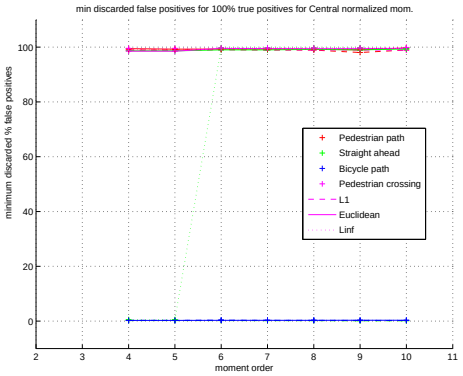
(d) Area below Flusser&Suk affine moment ROC's



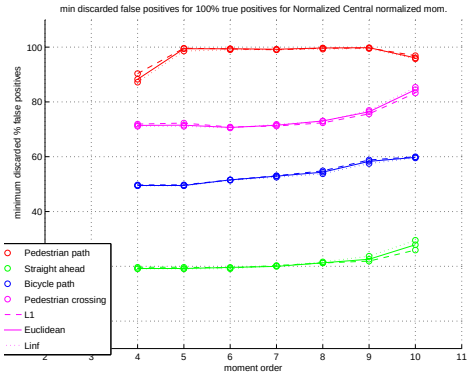
(e) Area below Flusser&Suk complex moment ROC's

Area below ROC for 90% traffic sign acceptance with iqr-normalized data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

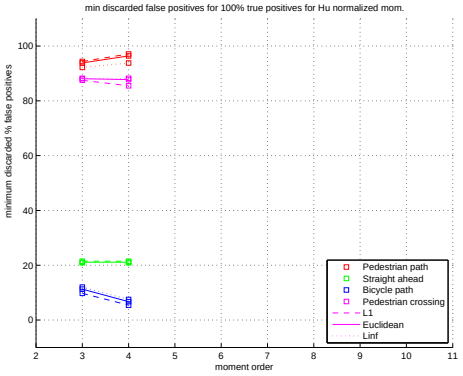
Figure 54: Synthetic overview AUC for 90% iqr-normalized true positives acceptance



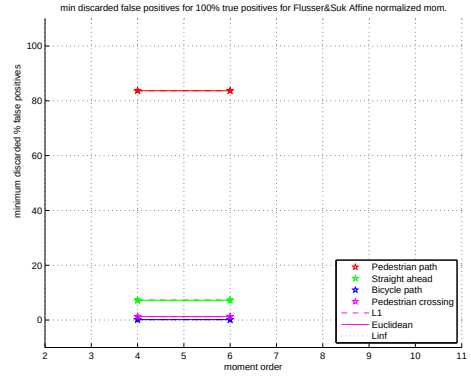
(a) Discarded Central moment background



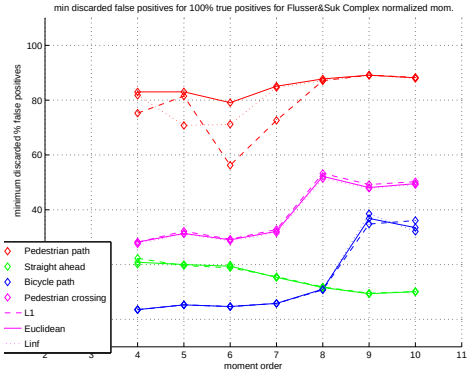
(b) Discarded Normalized Central moment background



(c) Discarded Hu moment background



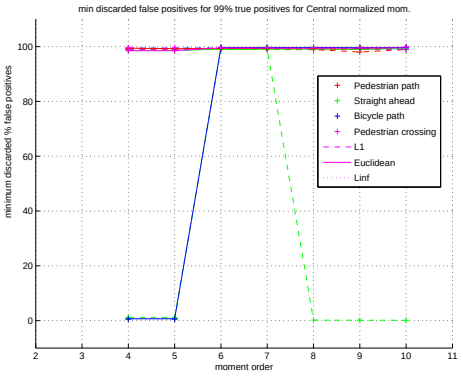
(d) Discarded Flusser&Suk affine moment background



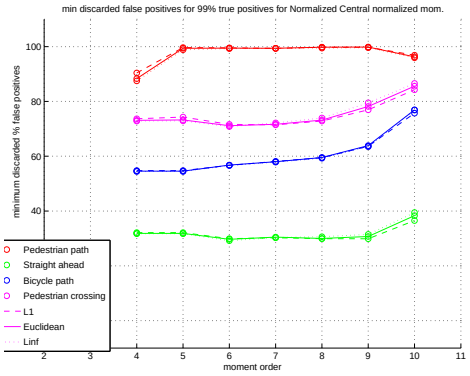
(e) Discarded Flusser&Suk complex moment background

Minimum discarded background data for 100% traffic sign acceptance with iqr-normalized data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

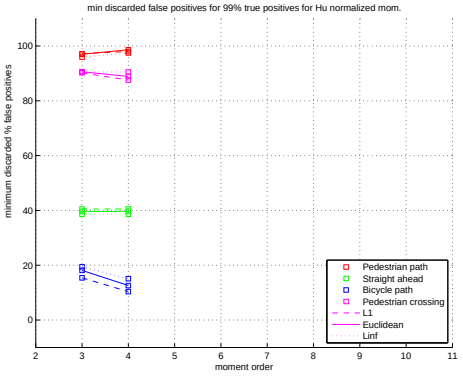
Figure 55: Minimum discarded background for 100% iqr-normalized true positives acceptance



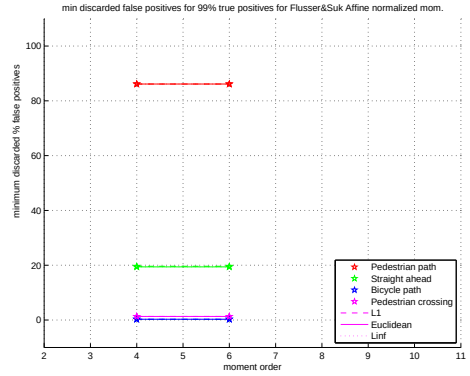
(a) Area below Central moment ROC's



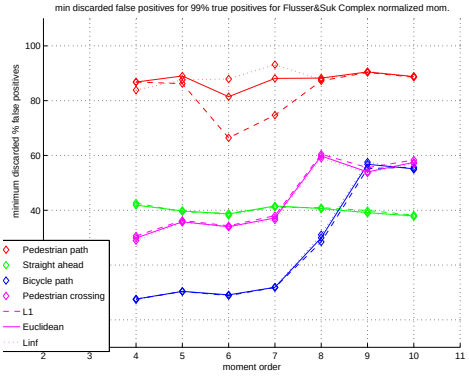
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



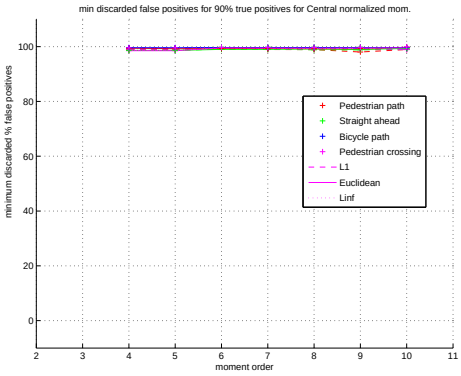
(d) Area below Flusser&Suk affine moment ROC's



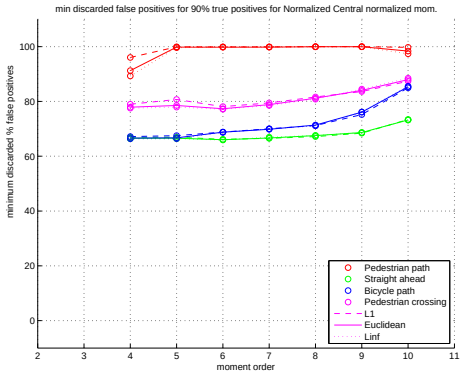
(e) Area below Flusser&Suk complex moment ROC's

Minimum discarded background data for 99% traffic sign acceptance with iqr-normalized data. Different point markers correspond to different traffic sign classes, different line patterns correspond to different distances.

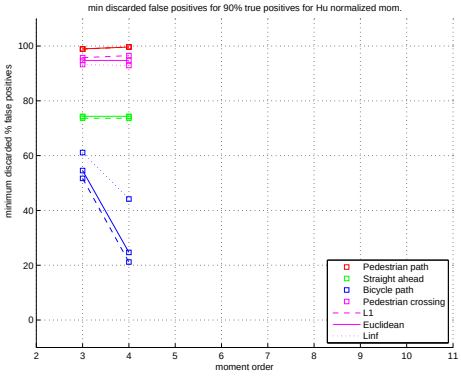
Figure 56: Minimum discarded background for 99% iqr-normalized true positives acceptance



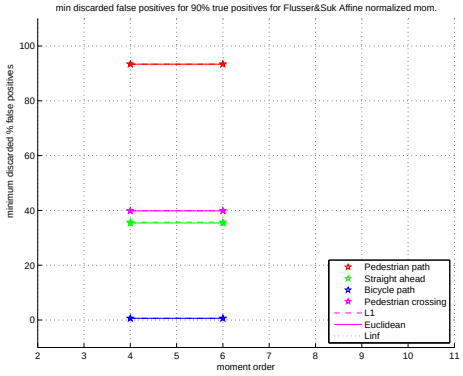
(a) Area below Central moment ROC's



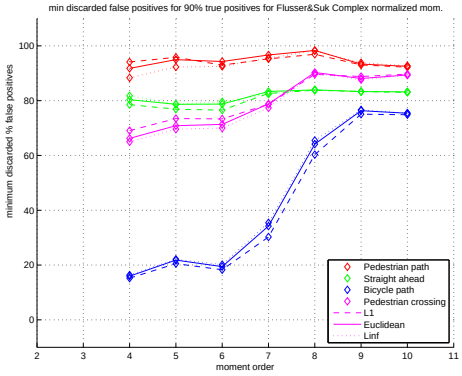
(b) Area below Normalized Central moment ROC's



(c) Area below Hu moment ROC's



(d) Area below Flusser&Suk affine moment ROC's



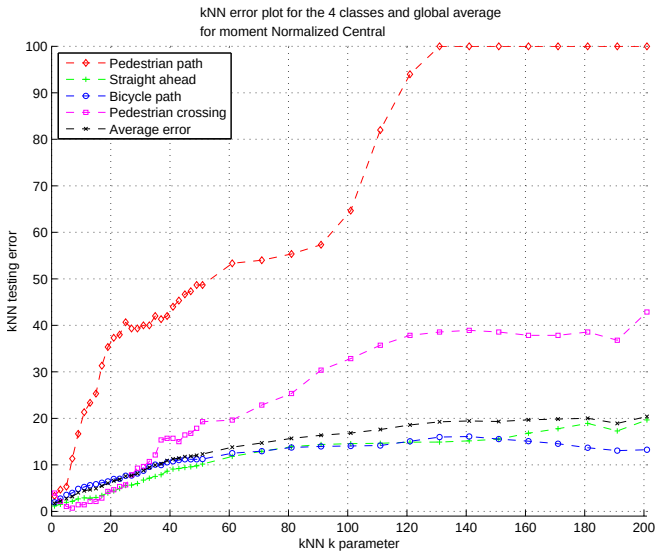
(e) Area below Flusser&Suk complex moment ROC's

Minimum discarded background data for 90% traffic sign acceptance with iqr-normalized data. Different point markers relate to different traffic sign classes, different line patterns relate to different distances.

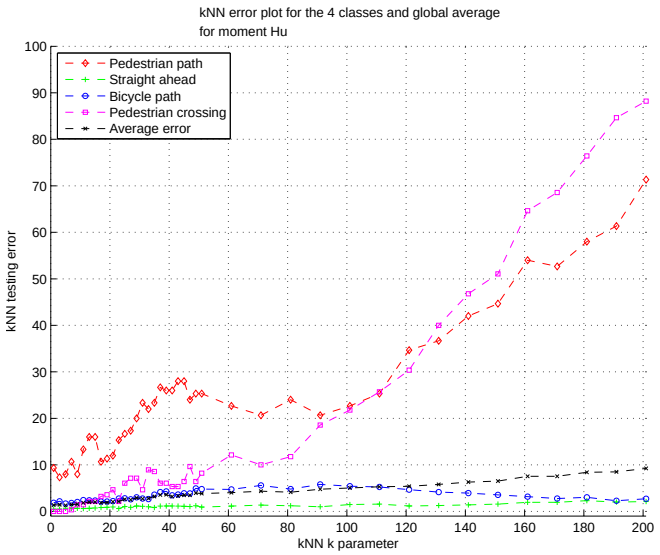
Figure 57: Minimum discarded background for 90% iqr-normalized true positives acceptance

B

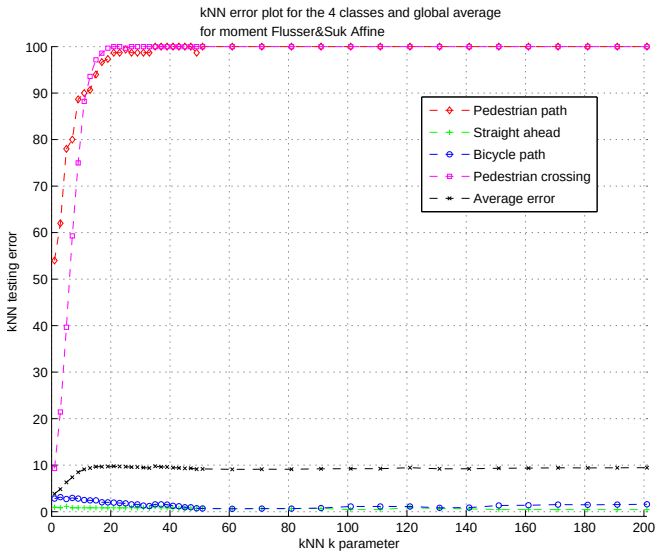
ADDITIONAL KNN CLASSIFICATION RESULTS



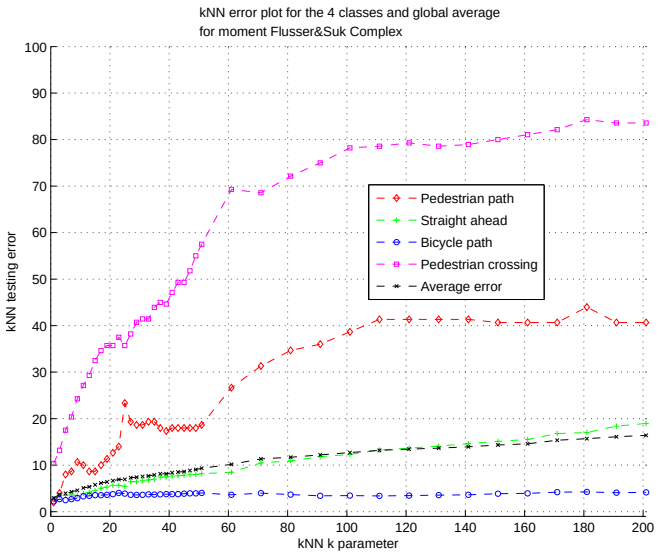
(a) Normalized Central moments classification



(b) Hu moments classification

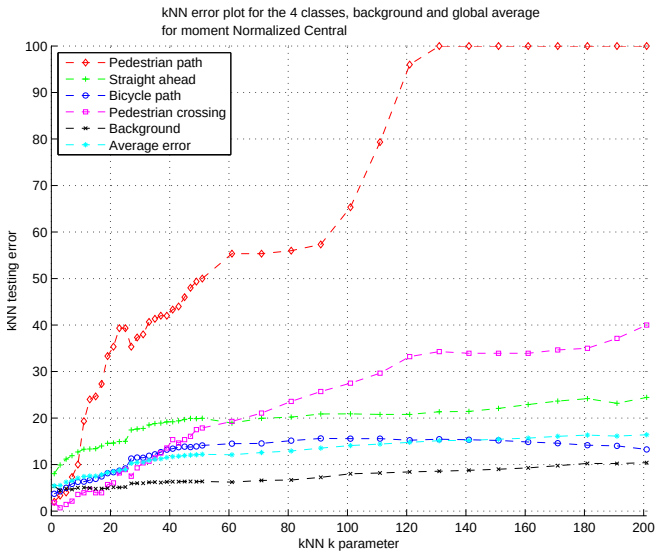


(c) Flusser and Suk affine moments classification

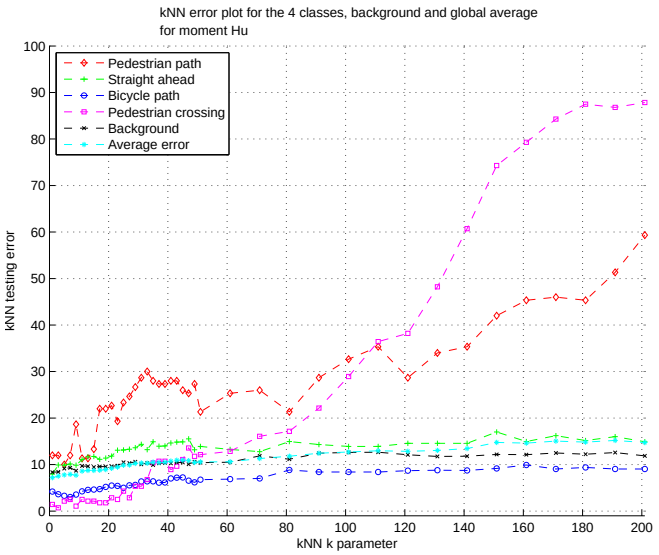


(d) Flusser and Suk complex moments classification

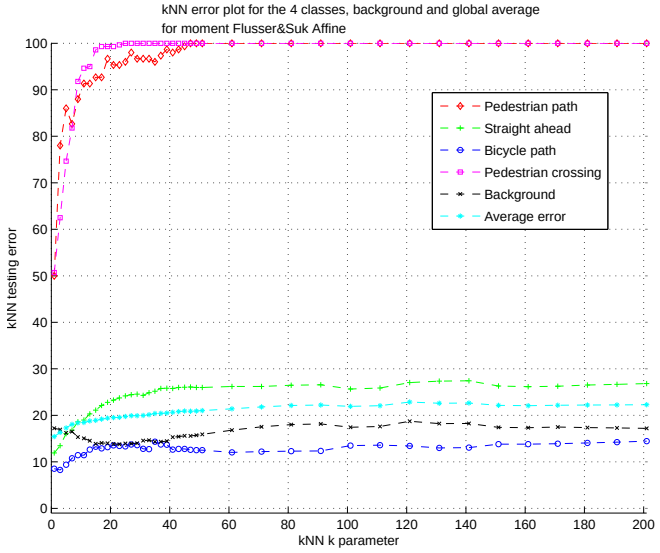
Figure 58: kNN plain data 4 classes classification results



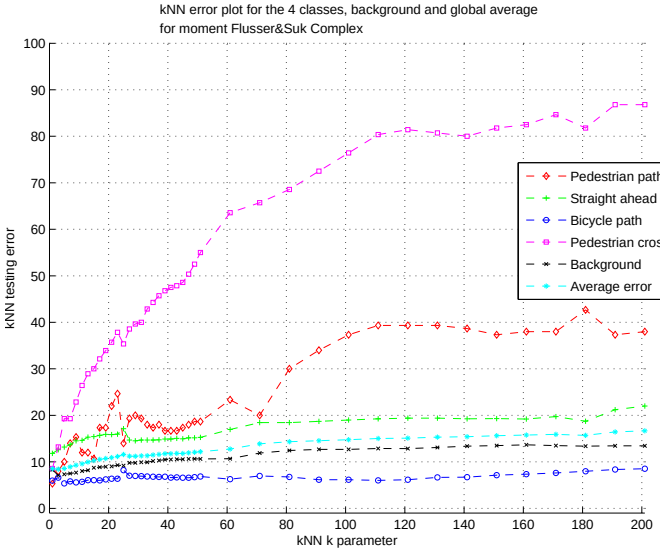
(a) Normalized Central moments classification



(b) Hu moments classification

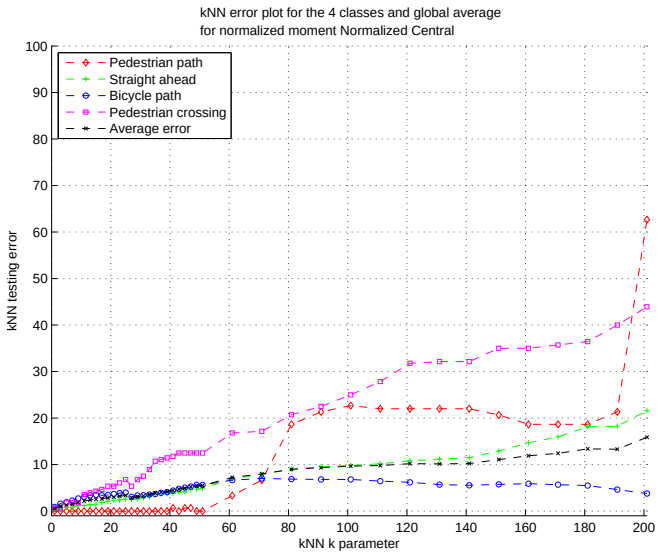


(c) Flusser and Suk affine moments classification

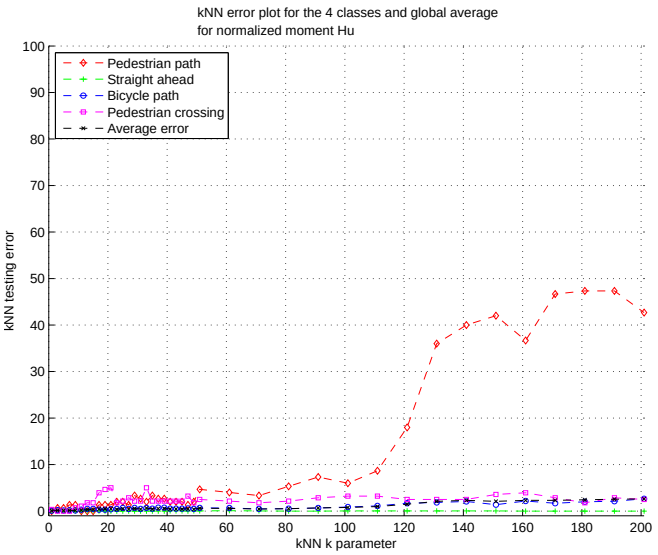


(d) Flusser and Suk complex moments classification

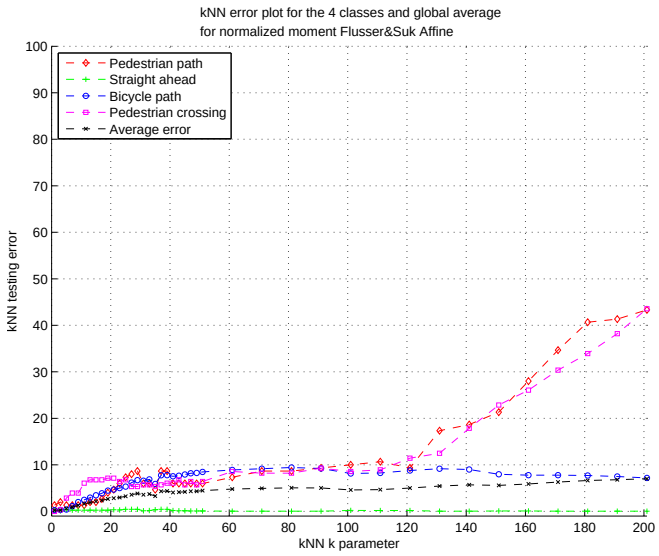
Figure 59: kNN plain data 4 classes and background classification results



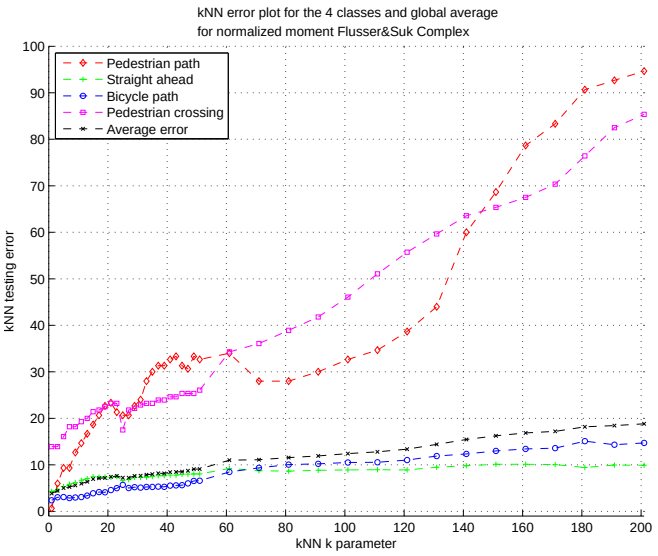
(a) Normalized Central moments classification



(b) Hu moments classification

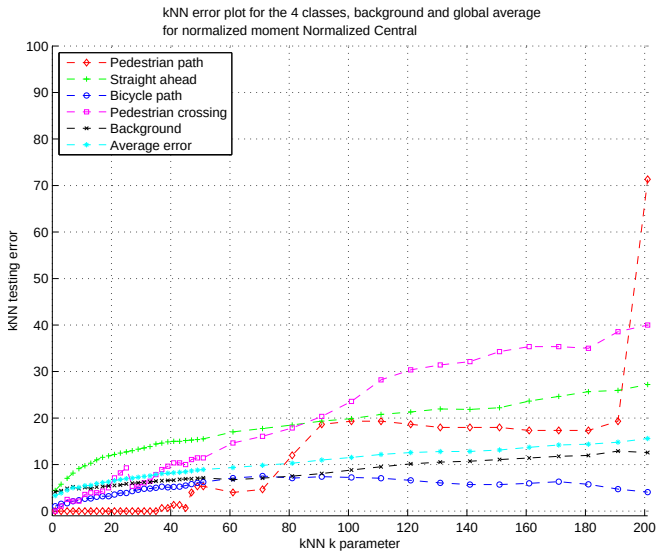


(c) Flusser and Suk affine moments classification

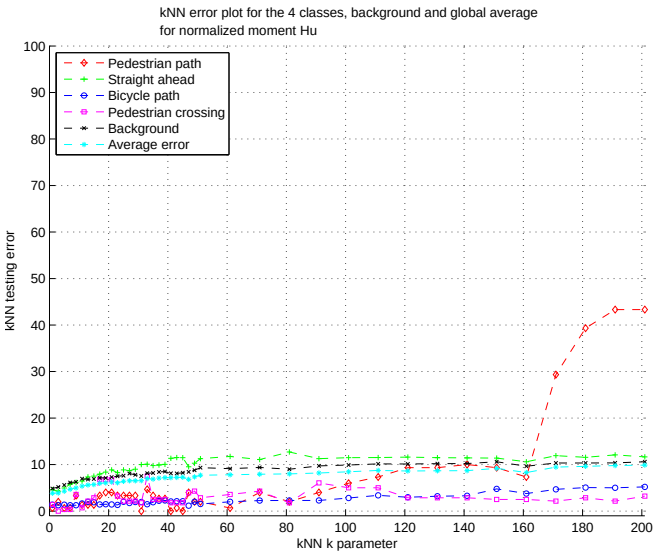


(d) Flusser and Suk complex moments classification

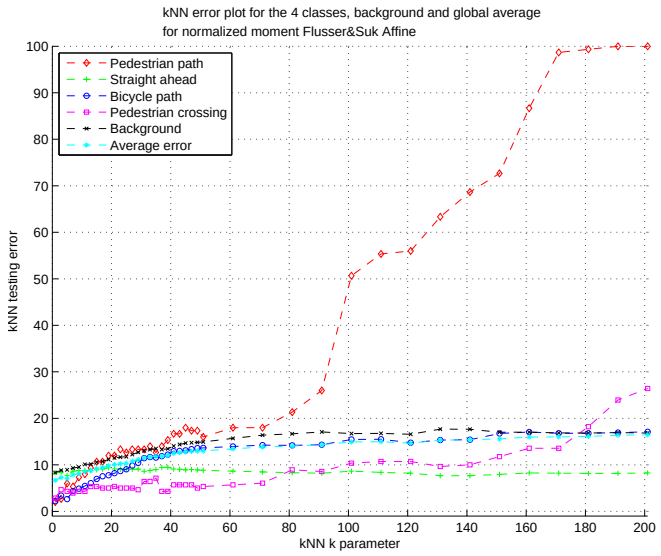
Figure 60: kNN normalized data 4 classes classification results



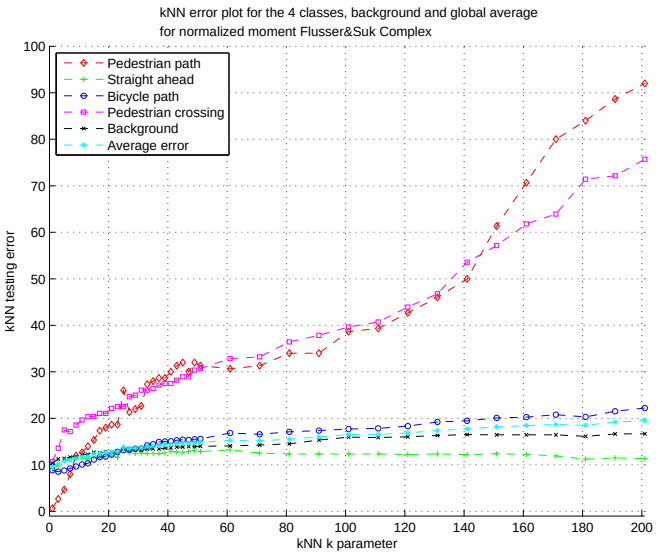
(a) Normalized Central moments classification



(b) Hu moments classification

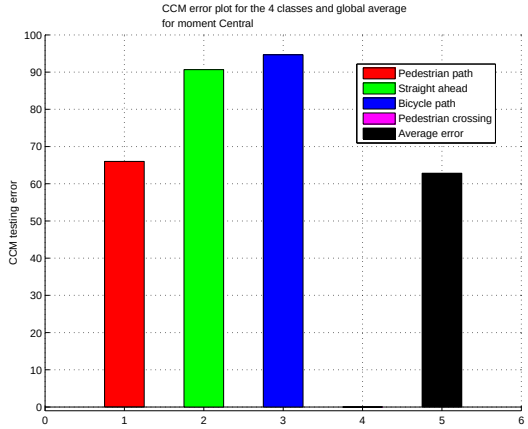


(c) Flusser and Suk affine moments classification

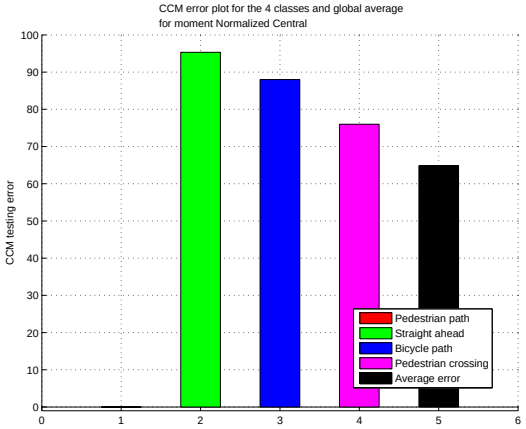


(d) Flusser and Suk complex moments classification

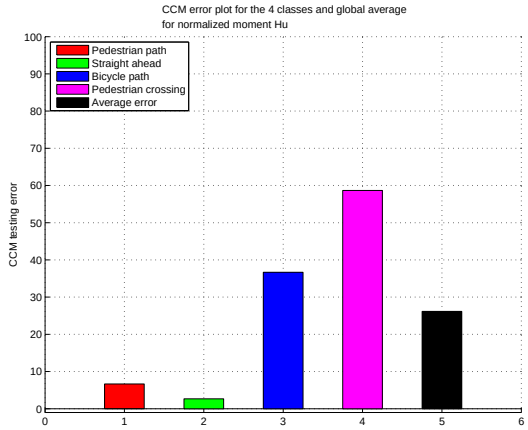
Figure 61: kNN normalized data 4 classes and background classification results



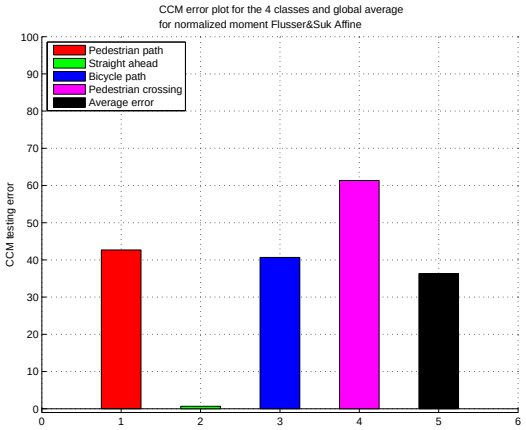
(a) Central moments classification



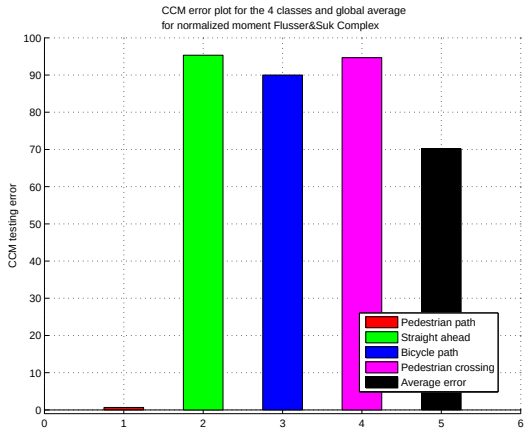
(b) Normalized Central moments classification



(c) Hu moments classification



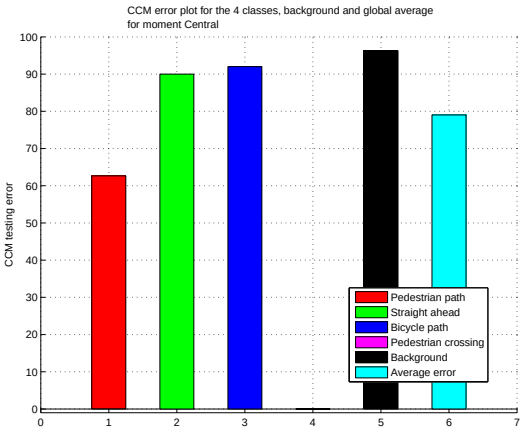
(d) Flusser and Suk Affine moments classification



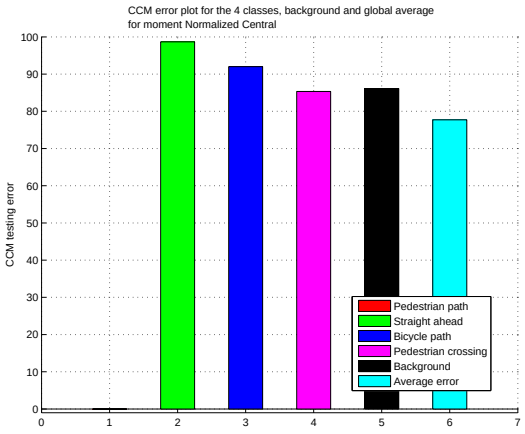
(e) Flusser and Suk Complex moments classification

Bar plots of the CCM classification error for the different moments; the average traffic sign classification error is plotted as well.

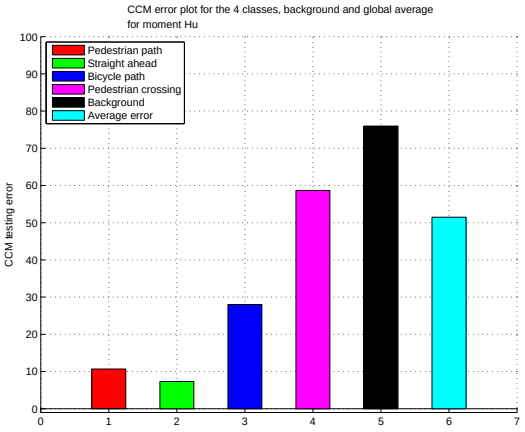
Figure 62: CCM plain data 4 classes classification results



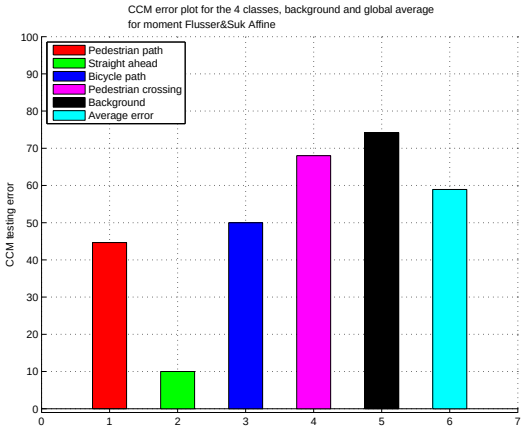
(a) Central moments classification



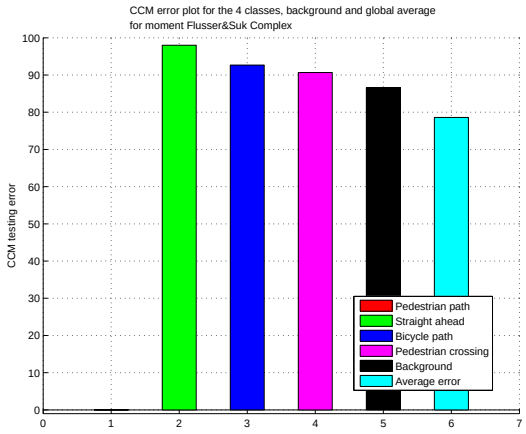
(b) Normalized Central moments classification



(c) Hu moments classification



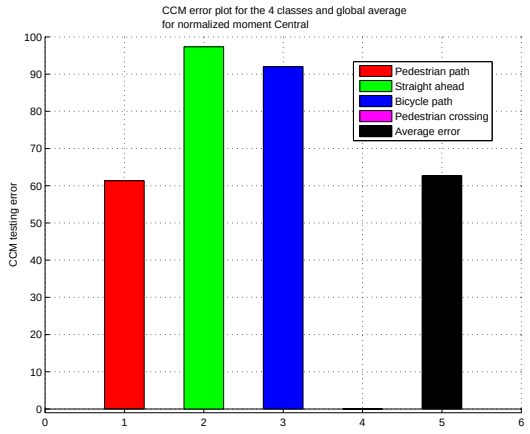
(d) Flusser and Suk Affine moments classification



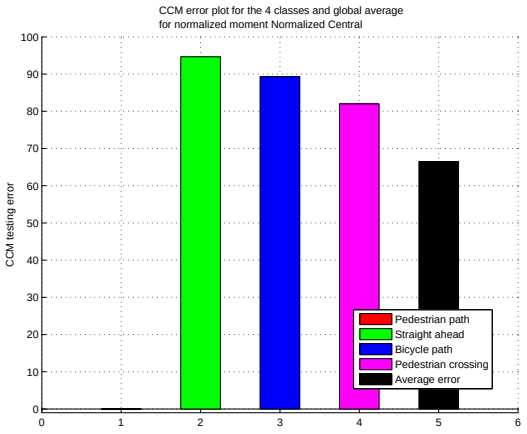
(e) Flusser and Suk Complex moments classification

Bar plots of the CCM classification error with background data for the different moments; the average traffic sign and background classification error is plotted as well.

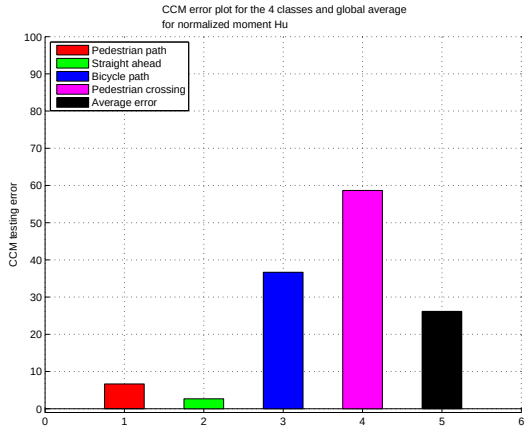
Figure 63: CCM plain data 5 classes classification results



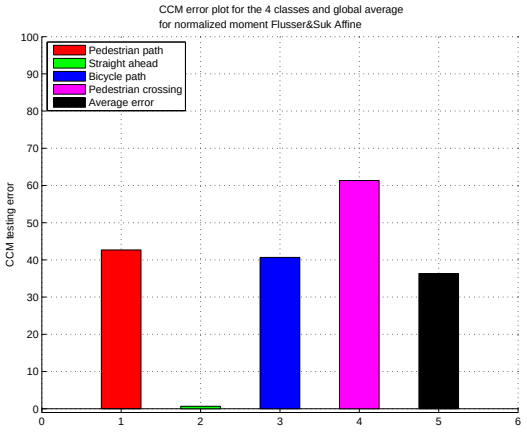
(a) Central moments classification



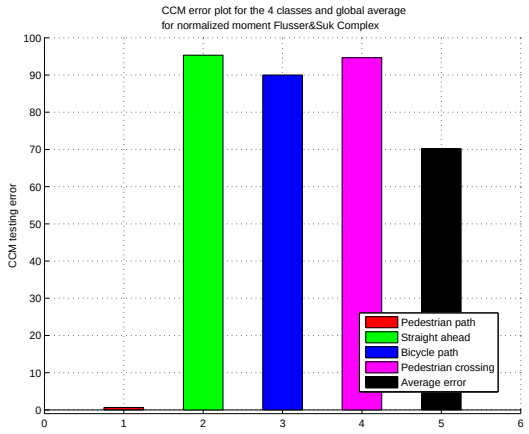
(b) Normalized Central moments classification



(c) Hu moments classification



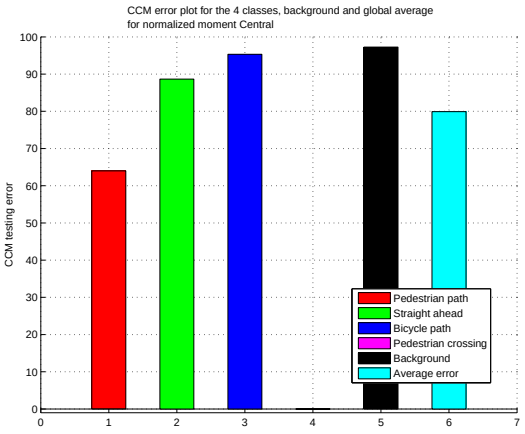
(d) Flusser and Suk Affine moments classification



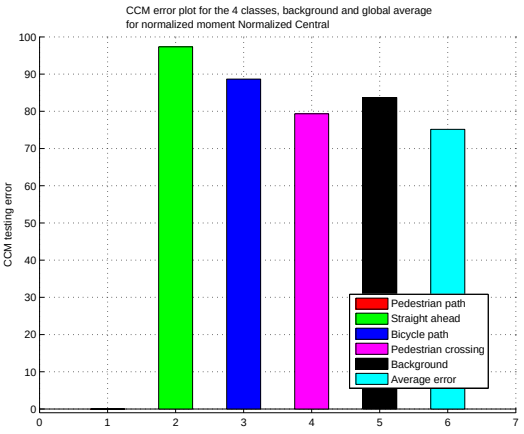
(e) Flusser and Suk Complex moments classification

Bar plots of the CCM classification error for the different moments; an average traffic sign classification error is plotted as well.

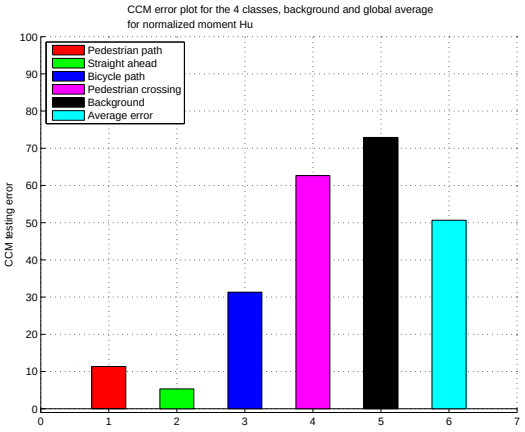
Figure 64: CCM normalized data 4 classes classification results



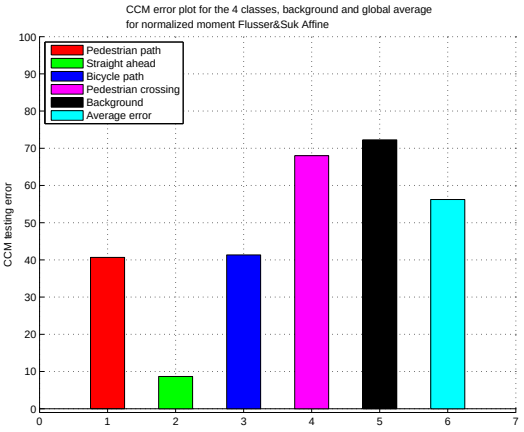
(a) Central moments classification



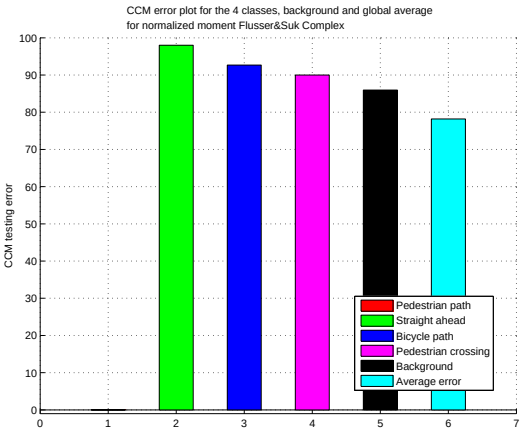
(b) Normalized Central moments classification



(c) Hu moments classification



(d) Flusser and Suk Affine moments classification



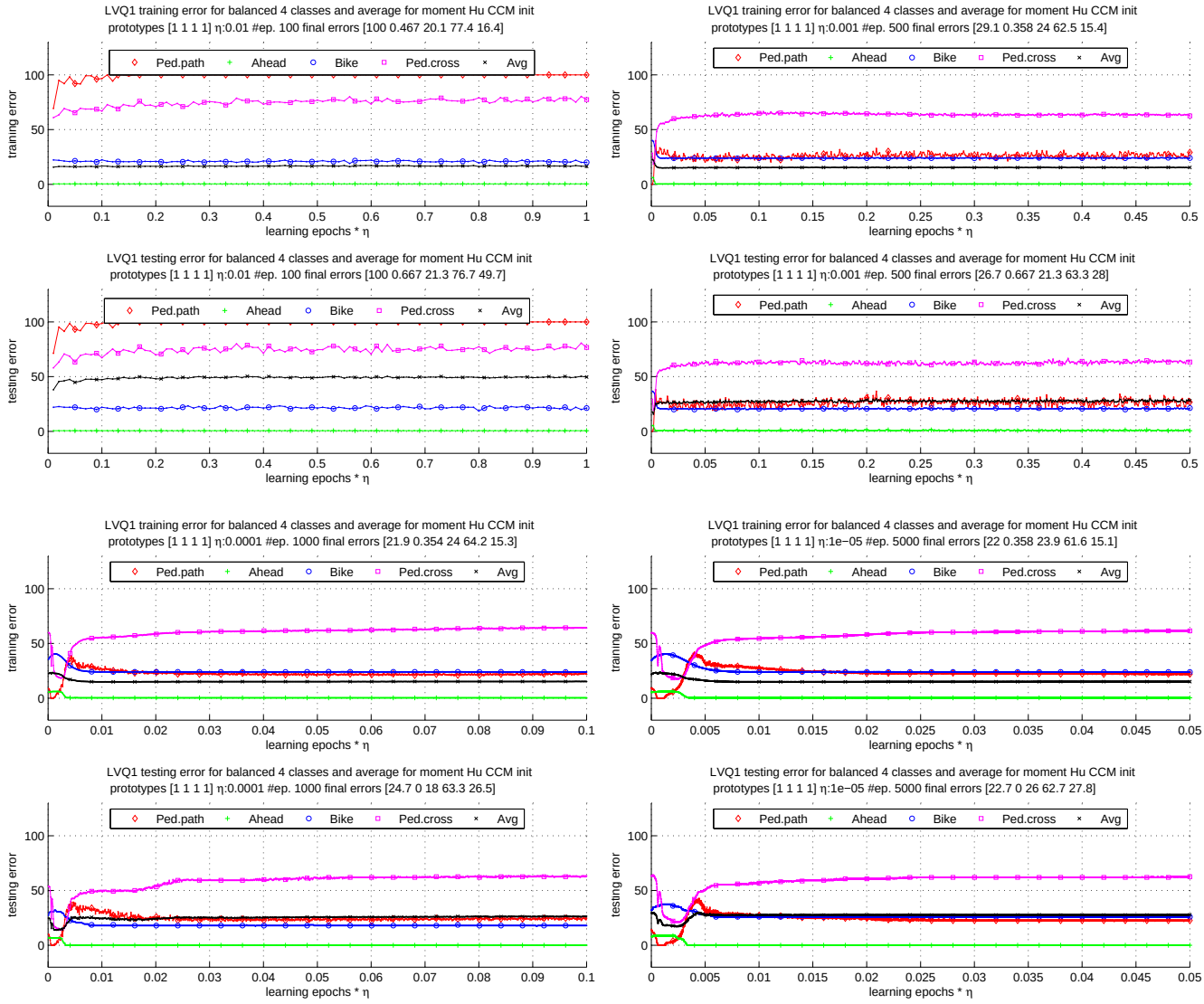
(e) Flusser and Suk Complex moments classification

Bar plots of the CCM classification error with background data for the different moments; an average traffic sign and background classification error is plotted as well.

Figure 65: CCM normalized data 5 classes classification results

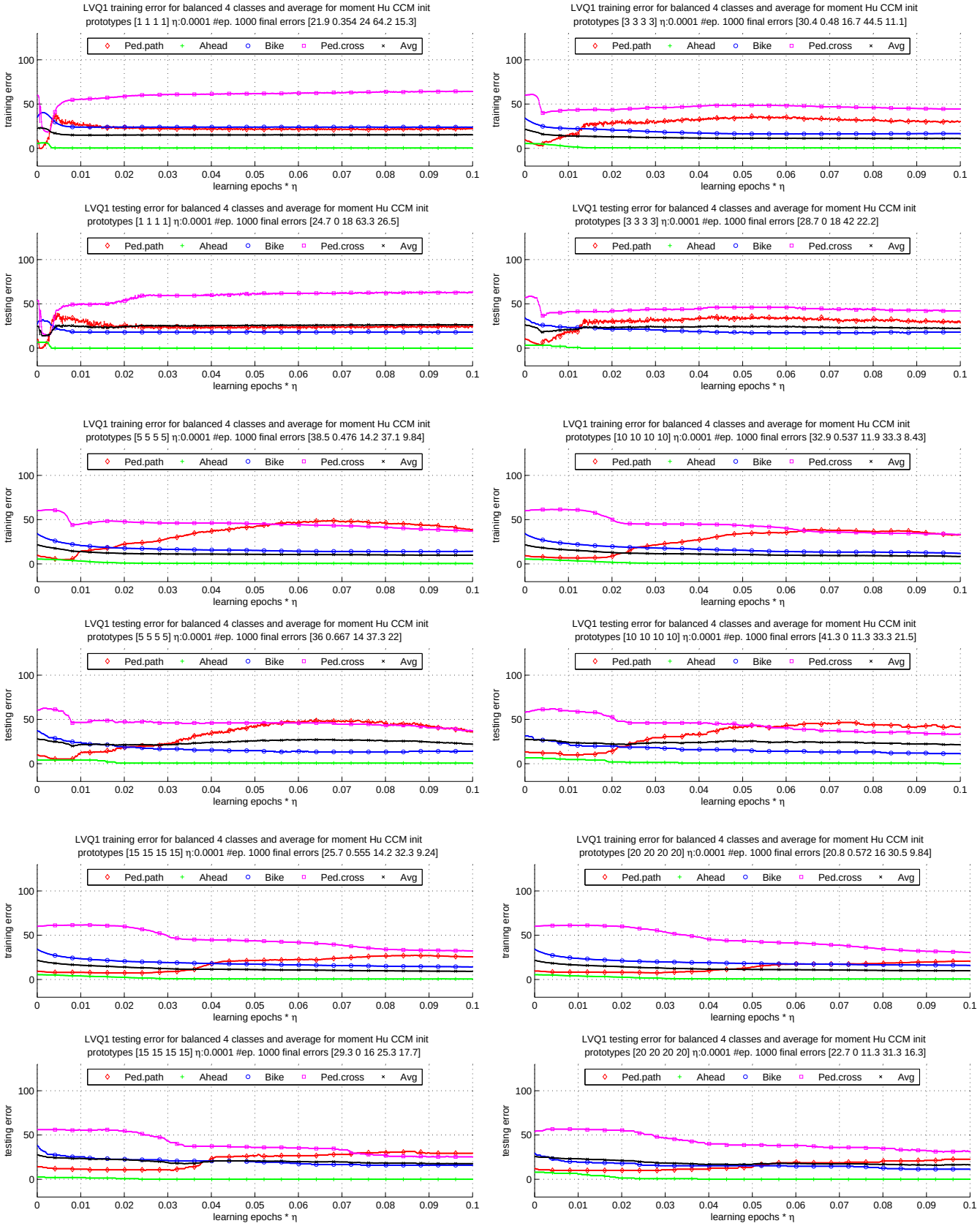
D

ADDITIONAL LVQ CLASSIFICATION RESULTS



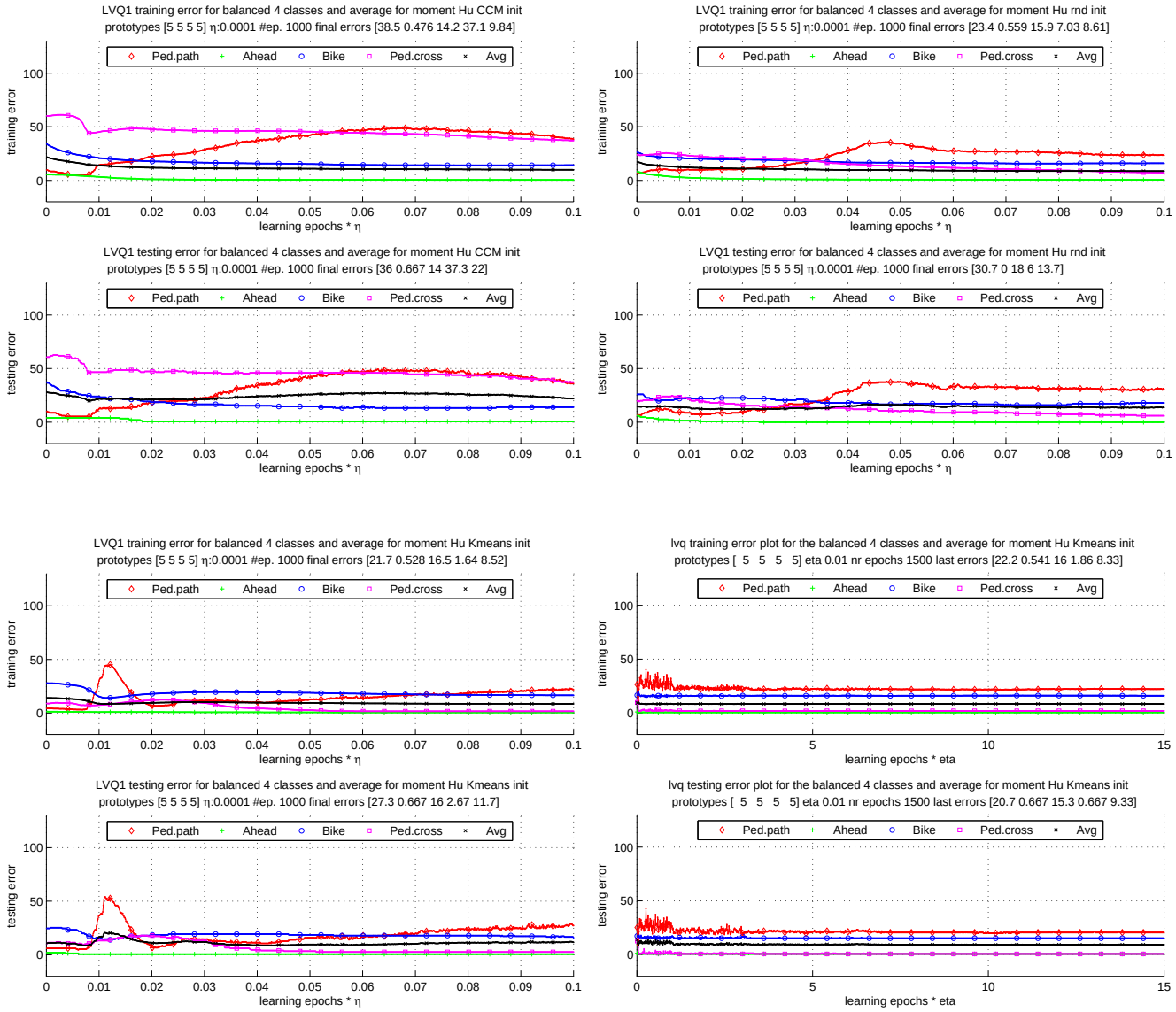
Error plots for the LVQ1 learner. Four class problems, one prototype per class, CCM initialization, varying η and number of epochs. For lower η the number of epochs is not increased proportionally as no learning takes place.
Note how lower η leads to better performance; for the lowest η the initial error evolution of the four prototypes becomes very evident.

Figure 66: Some LVQ1 error evolution plots for varying η



Error plots for the LVQ1 learner. Four class problems, varying number of prototypes per class, CCM initialization, $\eta = 0.0001$ and 1000 epochs.
Note how a larger number of prototypes leads to better performance; at the same time the initial error evolution of the four prototypes becomes increasingly stretched as well.

Figure 67: Some LVQ1 error evolution plots for varying number of prototypes



Error plots for the LVQ1 learner. Four class problems, five prototypes per class, $\eta = 0.0001$, 1000 epochs and varying initialization.
Note how better initialization leads to better performance and a steady learning curve.
The plot to the left on the bottom shows the behavior of the annealing schedule.

Figure 68: Some LVQ1 error evolution plots for varying initialization