

Universiteit
Leiden
The Netherlands

Content-based retrieval of visual information

Oerlemans, A.A.J.

Citation
Oerlemans, A. A. J. (2011, December 22). *Content-based retrieval of visual information*. Retrieved from https://hdl.handle.net/1887/18269

Version:	Corrected Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/18269

Note: To cite this publication please use the final published version (if applicable).

Content-Based Retrieval of Visual Information

Ard Oerlemans

Content-Based Retrieval of Visual Information

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden
op gezag van de Rector Magnificus prof. mr. P. F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 22 december 2011
klokke 10.00 uur

door

Adrianus Antonius Johannes Oerlemans

geboren te Leiderdorp
in 1977

Promotiecommissie

Promotor:	Prof. dr. J.N. Kok
Co-promotor:	Dr. M.S. Lew
Overige leden:	Prof. dr. C. Djeraba (University of Lille)
	Prof. dr. T.H.W. Bäck
	Prof. dr. H.A.G. Wijshoff
	Dr. E.M. Bakker

The cover of this thesis consists of images from the MIRFLICKR-25000 dataset. Each column represents the top results of a color-based query using a specific wavelength of light as the query.

Contents

1 Introduction **1**
- 1.1 Content-based image retrieval . . . 3
- 1.2 Research areas in CBIR . . . 5
  - 1.2.1 Image segmentation . . . 5
  - 1.2.2 Curse of dimensionality . . . 5
  - 1.2.3 Semantic gap . . . 6
  - 1.2.4 Searching with relevance feedback . . . 6
  - 1.2.5 Future CBIR challenges . . . 6
- 1.3 Thesis contents . . . 7

2 Features **9**
- 2.1 Introduction . . . 9
- 2.2 Color features . . . 10
  - 2.2.1 Color histogram . . . 10
  - 2.2.2 Color moments . . . 10
- 2.3 Texture features . . . 11
  - 2.3.1 Local binary patterns . . . 11
  - 2.3.2 Symmetric covariance . . . 11
  - 2.3.3 Gray level differences . . . 12
- 2.4 Feature vector similarity . . . 12

3 Machine Learning **15**
- 3.1 Introduction . . . 15
  - 3.1.1 A sample binary classification problem . . . 16
- 3.2 k-nearest neighbor . . . 16
- 3.3 Artifical neural networks . . . 17
- 3.4 Support vector machines . . . 18

4 Performance Evaluation **21**
- 4.1 Precision . . . 21
- 4.2 Recall . . . 22
- 4.3 Precision-Recall graphs . . . 22

4.4 Average precision 25
4.5 Accuracy 26

5 Interest Points Based on Maximization of Distinctiveness 27
5.1 Introduction 27
5.2 Related work 28
5.3 Maximization Of Distinctiveness (MOD) 28
5.3.1 The MOD paradigm 29
5.3.2 The special case of template matching 30
5.3.3 Detector output 31
5.4 Matching images 36
5.5 Experiments and results 36
5.6 Discussion and conclusions 39

6 Learning and Visual Concept Detection 41
6.1 Introduction 41
6.2 Related work 43
6.3 Maximization Of Distinctiveness (MOD) 43
6.4 Detecting visual concepts 43
6.4.1 Classifiers 44
6.5 Experiments 44
6.5.1 Tree detection 46
6.5.2 Building detection 46
6.5.3 Sky detection 48
6.5.4 Beach classification 49
6.5.5 Face detection 49
6.6 Experiments on MIRFLICKR-25000 dataset 51
6.6.1 Concept 'Animals' 52
6.6.2 Concept 'Indoor' 54
6.6.3 Concept 'Night' 56
6.6.4 Concept 'People' 58
6.6.5 Concept 'Plant life' 60
6.6.6 Concept 'Sky' 62
6.6.7 Concept 'Structures' 64
6.6.8 Concept 'Sunset' 66
6.6.9 Concept 'Transport' 68
6.6.10 Concept 'Water' 70
6.6.11 Overall results 72
6.7 Discussion, conclusions and future work 72

7 Multi-Dimensional Maximum Likelihood 75
7.1 Introduction 75
7.2 Definitions 76
7.3 Detailed description 76

7.4 Related work . . . 78
7.5 Multi-Dimensional Maximum Likelihood similarity (MDML) . . . 79
7.6 Experiments on stereo matching . . . 80
7.6.1 Results - template based . . . 80
7.6.2 Results - pyramidal template based . . . 80
7.7 Future work . . . 83

8 Texture Classification: What Can Be Done with 1 or 2 Features? 85
8.1 Introduction . . . 85
8.2 Related work . . . 86
8.3 Our method . . . 86
8.4 Results . . . 88
8.5 Discussion, conclusions and future work . . . 90

9 Detecting and Identifying Moving Objects in Real-Time 93
9.1 Introduction . . . 93
9.2 Related work . . . 94
9.3 Motion detection . . . 94
9.3.1 Building the background model . . . 95
9.3.2 Adaptive background model . . . 97
9.3.3 Post processing . . . 98
9.4 Object tracking . . . 98
9.4.1 Data structure . . . 99
9.4.2 Object motion prediction . . . 100
9.4.3 Rule-based object tracking . . . 101
9.5 Results . . . 105
9.6 Conclusions and future work . . . 106

10 Hybrid Maximum Likelihood Similarity 109
10.1 Introduction . . . 109
10.2 Related work . . . 110
10.3 Visual similarity . . . 110
10.3.1 The maximum likelihood training problem . . . 110
10.3.2 Hybrid maximum likelihood similarity . . . 111
10.4 Relevance feedback in object tracking . . . 111
10.4.1 Pixel-level feedback . . . 112
10.4.2 Object-level feedback . . . 113
10.5 Conclusions and future work . . . 114

A RetrievalLab 117
A.1 Introduction . . . 117
A.2 Related work . . . 117
A.3 Example usage . . . 118
A.3.1 Image retrieval . . . 118

A.3.2 Visual concept detection 120
A.4 Discussion, conclusions and future work 121

Bibliography **123**

Nederlandse Samenvatting **131**

Acknowledgements **135**

Curriculum Vitae **137**

Chapter 1

Introduction

We live in an Age of Information, a period in time where almost limitless amounts of information are available from a multitude of sources containing text, images, video, audio and other types of information. Take for example Facebook, which has over 10 billion photos, or Google, which has indexed tens of billions of web-pages, or YouTube, which hosts over 140 million videos. Beyond these publicly available sources, there are for example the digitized contents of the libraries and museums worldwide.

Storing this information in a database is not enough to take advantage of the knowledge stored in this data. We also need to be able to search through it. In many situations, text annotation is incomplete or missing in which case it is necessary to turn to content analysis techniques, that is, methods which analyze the pictorial content of the media. It is also noteworthy that even when text annotation is available, it may be possible to improve the quality of search results by also using the pictorial content information.

Searching through digital data is a very active field of research. For each of the types of digital information, specific search methods exist and this thesis aims to add to that research by exploring a specific part of searching in digital information: content-based image retrieval (CBIR). In this type of searching, the pictorial contents of images are automatically analyzed and indexed, to allow search methods to use these contents, instead of relying on descriptions.

In this thesis, we extend existing methods for performing content-analysis of images, but we also try to extend the search process itself by adding an interactive component, which is called relevance feedback. We also look at relevance feedback procedures in video-analysis, or more specifically, object tracking.

When searching for information, a user in general starts with supplying a description of what the user wants to find, also known as the query. The search engine processes the query and presents the results to the user. These results are possibly ranked by relevance, which is usually the similarity of the search results

to the query. This is a very common way of searching, used by all the well-known text search engines on the Internet.

The most widely used method of searching on the Internet is text-based searching. The user supplies a set of descriptive words and the search engine retrieves documents that contain these words.

The common technique for text based searching, is the inverted index [48]. An inverted index contains all known words from the documents in the database and for each word it contains a list of documents that contain that specific word. Search speeds are greatly improved, because not every document has to be compared to the query.

Relevance feedback was originally designed to extend the search process by asking the user to give feedback on the search results to the search system. The search system can combine this feedback with the original query to run a new search with hopefully more relevant results.

An example of this would be a person asking for a book about Africa in a book store. Initially, the employee of the book store will come up with a few books that have Africa as their topic. However, after looking at these books, the customer decides that some of these books describe the African culture and some others describe the species of animals that live there. At this point the customer decides that it was actually these species of animals that he or she was interested in and not the culture.

The customer points out a book that is more like what he or she was looking for and also points out another book that does not contain the desired type of information. Now the person working at the book store knows more about the type of books the customer is looking for and can search for another set of books to show to the customer. Essentially, by giving feedback, the query has been changed in a direction that will result in better search results.

The original relevance feedback algorithm was designed in 1971 by J.J. Rocchio [65] and it was applied to text-based searching. Later, Salton and Buckley [69] improved the original formula, to get to the following result:

$$Q_{i+1} = \alpha Q_i + \beta \sum_{rel} \frac{D_i}{|D_i|} - \gamma \sum_{nonrel} \frac{D_i}{|D_i|} \tag{1.1}$$

In words, this means that the query is adjusted by including knowledge of relevant and non-relevant documents. The new query Q is based on a weighted sum of the previous query and the relevant and non-relevant documents that were selected by the user. Eventually, this process will result in a query point that is at the optimal location for separating the two classes of relevant and non-relevant documents.

For text based searches, this translates to using a weight vector for the words that are used for the document retrieval. Relevant documents increase weights on certain words (or even add new words) and non-relevant documents decrease the weights on other words.

1.1 Content-based image retrieval

Content-based image retrieval (CBIR) uses the actual pictorial contents of images in the search process. In this case, the query is an image, instead of a set of words. The search system uses contents of the image to search for matching images.

There are many reasons to use image contents in the search process, instead of user-supplied tags. Some of them are:

- *Tags could be missing*
  A set of pictures taken at a vacation, is usually not tagged. The entire set probably has a description, but the contents of each individual image are not described.
- *Tags could be incorrect or not descriptive of the contents*
  Users may supply tags that are incorrect, for example, these could be a representation of the situation in which the picture was taken, but not what can be seen on the picture.

  Note that users still might want to search for these higher-level descriptions. This is probably one step further than CBIR, because in this case the contents of images are linked to a notion of a situation or a location. (Photos taken of the crowd at the inauguration of Obama will probably not show the president, but when using this image as input for a search, people expect the search engine to return images of a crowd at this specific event only.)
- *Tags are not always able to capture the true contents*
  For example, for more complex textures such as a view of the Rocky Mountains, there are no words that truly describe the image contents.

These examples explain the need for using different techniques than text-based searching. Content-based techniques do not depend on external descriptions to perform a search task. However, it is also possible to combine the two types of searching.

The contents of images can be analyzed in various ways. Low-level features such as color, texture and shape are commonly used, but higher level features, or concepts are also available for describing images.

A good overview of the history of CBIR systems is given by Veltkamp et al. [91] and Smeulders et al. [83]. However, we would like to emphasize a few notable systems from the past.

QBIC (Query-By-Image-Content) [16] was developed by IBM and presented in 1995 as one of the first systems that enabled searching through image and video databases based on image contents. Even today, the QBIC technology is still commercially used in DB2, a data-management product by IBM. The system can use image properties such as color percentages, color layout and textures in the search process.

In the same year, Chabot [59] was presented. By integrating image information

stored in a database, which can be text and other data types, in combination with properties of image contents, the user can search for 'concepts'.

One of the first systems that used relevance feedback in an image retrieval system was MARS, Multimedia Analysis and Retrieval System. It was first demonstrated in 1996 as a basic image retrieval system [29] by Thomas Huang and was later extended with the relevance feedback component [68] by Rui.

The ImageScape image retrieval system [37] by Michael Lew, used several methods for searching through images, one of them being query by icons, a method that used predefined visual concepts, which made it one of the first systems to use visual concepts for image retrieval. The concepts could be placed on a canvas by the user in the form of icons and the system would then retrieve images for which the concept was detected at the user-specified locations.

In content-based retrieval, several promising research directions have emerged. Some try to reduce content-based searching to text-based searching, others focus on the problems of interest point detection or sub-image searching and yet another direction is the use of relevance feedback techniques.

Usually, image database lack user-supplied tags, so automatically tagging these would be a desirable option. As described in [41], real-time automated tagging of images is already a promising research direction. This research combines low-level features into a concept that can be described with words. Searching for images is then reduced to text-based searching. The query image is translated into tags (in real-time) and the database is queried for the best matching concepts.

Interest points are locations within an image that can be automatically calculated and that define the best input for other algorithms, such as object matching, tracking and image retrieval.

One of the earliest interest point detectors was Moravecs corner detector [52]. Other well-known more recent algorithms are SIFT [44] and SURF [1].

Searching for images, or image contents, is not bound by the area of the entire image. The query contents can be part of a larger image. The research area for sub-image searching tries to solve this problem by subdividing database images into smaller subimages that can be matched to the query.

The same method can also be applied by subdividing the query image into sub-images and to use these as separate queries. The ImageScape system [37] did this by handling each user-placed icon as a query for a visual concept.

Some of the challenges in this research area are:

- How can we subdivide an image into regions that are meaningful to be used for sub-image searching
- What features can be used to describe the sub-images so that they can be matched to other sub-images, that possibly have different shapes or sizes

In a CBIR task, the text-based relevance feedback process can be translated to changing the image contents that the user is searching for. The user-supplied

image is combined with feedback on the search results, resulting in a virtual query image that contains elements of both the user input and the feedback images.

As an example: if the original query contained the color green and a round shape, but the user has given positive feedback for an image that contains the color blue, the new query would probably result in images that contain the color blue and round shapes.

1.2 Research areas in CBIR

This paragraph describes some of the topics in content based image retrieval that have drawn the attention of researchers in previous years and it introduces a few challenges of CBIR that will probably be the subject of many research projects in the future.

1.2.1 Image segmentation

In partial image searches, the question is how to define the image parts. A straightforward way would be to linearly divide the image into several rectangular regions, but this will have problems in that real object boundaries will rarely coincide with the rectangular regions. A better way would be to use image properties as a segmentation guide, so that the segmented regions have the same properties. There are several ways of selecting segmentation properties, but image intensity, color and texture are common choices.

A recent example of such a segmentation method is fuzzy regions [63], used in the FReBIR system.

1.2.2 Curse of dimensionality

One of the first, logical, steps in setting up an image retrieval system is to select a large number of different features, to increase the chance of finding perfectly matching images. For example, one could choose multiple color features to improve color-based matching.

However, there is a downside to increasing the number of features that are used for similarity matching and this is expressed by the 'curse of dimensionality'. This term, which was first mentioned by the mathematician Richard Bellman [2], is used to express the difficulties that arise with using distances between high-dimensional vectors. In high-dimensional spaces, every vector seems to be at a very large distance from any other vector and then the question is, what the usefulness of these distances is in finding the best match based on the selected features.

1.2.3 Semantic gap

In many image retrieval systems, low-level features such as color, texture and shape are commonly used to describe images or parts of images. On the other hand, users tend to think in higher level concepts, such as house, person or desert. (Or even higher level concepts such as 'inauguration of Obama'.) The relation between a set of low-level features and a high-level concept is still a challenge for researchers in the CBIR community and the term 'semantic gap' is often used to describe the lack of a solid theory or methods to overcome this.

In other words, the semantic gap is used to describe the unclear relation, if any, between low-level features and high-level concepts. One would like to say 'if the texture of the area is this and the color is that, there must be a car in this area'. However, there are still no systems that truly bridge the semantic gap by providing these kinds of rules.

1.2.4 Searching with relevance feedback

If only the contents of an image are used as a query for an image retrieval system, ambiguities will definitely arise. A well known saying is 'an image is worth a thousand words' and this also applies to the images that are used as input for image retrieval: one image can have many different meanings to many different users. In other words, two different users may have significantly different goals for their query when the same image is used as a query.

In text-based searches this effect can also be seen, when a word has several meanings, such as 'monitor'. The Wikipedia disambiguation page for monitor lists several different meanings, from the computer monitor to a town in Indiana, US. Without asking the user for feedback, there is no way of knowing what a user is searching for.

1.2.5 Future CBIR challenges

There are many challenges in the field of CBIR research that still need to be addressed. An overview of these challenges was recently given in [40]. The authors conclude that the following five challenges are noteworthy:

- Concept detection in the presence of complex backgrounds
- Multi-modal analysis and retrieval
- Experiential multimedia exploration
- Interactive search
- Performance evaluation

1.3 Thesis contents

This research has focused on two types of digital information: images and video. Chapters 2, 3 and 4 give a general overview of the image features, machine learning techniques and performance evaluation methods that were used. Chapters 5 to 8 contain techniques that are applied to image searching. Chapters 9 and 10 show the results of relevance feedback on object tracking in video. A more detailed description of each chapter is given below.

Chapter 2 gives an overview of existing image features and similarity methods that are used in this research. Chapter 3 gives an overview of the machine learning techniques used in this research. In chapter 4 various performance measures are described that were used to evaluate the experiments.

In Chapter 5 a new interest point detector is presented. The detector uses local dissimilarity to determine the most distinctive points in an area, based on a selected feature or combination of features. We presented this work at the 10th ACM International Conference on Multimedia Information Retrieval (MIR) in Vancouver, Canada in 2008.

Chapter 6 demonstrates the use of relevance feedback for visual concept detection. A visual concept is learned by asking the user for positive and negative examples of the concept. This concept is then used for pointing out parts of images that contain the concept. This contribution was published in the proceedings of the 21st Benelux Artificial Intelligence Conference (BNAIC) in Eindhoven, The Netherlands in 2009.

An improved version of the paper used our new interest point detector combined with an enhanced wavelet representation feature and shows results of experiments on the MIRFLICKR-25000 dataset. This paper was presented at the 11th ACM International conference on Multimedia Information Retrieval (MIR) in Philadelphia, Pennsylvania, USA in 2010.

Chapter 7 presents a novel similarity measure that uses the coincidence of feature values in a training set of similar images and maps this in a 3D space. The resulting surface is used as the similarity measure when searching for new images.

In Chapter 8, a new texture feature is described, which is a generalization of the well-known 3x3 texture unit paradigm, that has shown that the statistical distribution of 3x3 blocks is a very good classifier for textures [25]. The novel texture feature was published in the proceedings of the 6th IEEE International Symposium on Image and Signal Processing and Analysis (ISPA) in Salzburg, Austria in 2009.

Chapter 9 presents a robust, adaptive object tracking system that was presented at the 11th Annual Conference on Computing and Imaging (ASCI) conference in 2005. It was also used as the basis for further research for this thesis.

Chapter 10 builds on the new similarity measure based on multidimensional maximum likelihood. This work was presented at the IEEE International Workshop

on Human Computer Interaction (HCI)in Rio de Janeiro, Brasil in 2007.

Chapter 10 also demonstrates the use of relevance feedback to object tracking. Tracked objects can be selected as positive or negative examples and the tracking system can keep tracking these objects when they are standing still, or it can ignore them. A paper based on this techniques was published in the ACM International Conference on Image and Video Retrieval (CIVR) in Amsterdam, The Netherlands in 2007.

Appendix A describes RetrievalLab, an educational and research tool to illuminate the process of content-based retrieval. RetrievalLab was presented at the ACM International Conference on Multimedia Retrieval (ICMR) in Trento, Italy in 2011.

Chapter 2

Features

This chapter describes the low level features we have used in the content-analysis of images. First, a short introduction is given to explain what a low level feature is and then the features that were used in this thesis are explained in detail. Also, we describe a few measures for calculating the similarity of low level features.

2.1 Introduction

The contents of images need to be described in a form that the search system understands. This can be done in various ways and usually one starts with the extraction of low level features. A low level feature can be extracted from an image by calculating a mathematical formula or by running a simple algorithm on the image data. The result is a number or a set of numbers that represents the feature and this set of numbers is called the feature vector. These vectors are almost always normalized to unit length. A low level feature generally focuses on aspects such as color, texture or shape.

Low level features can be combined to form more complex descriptions of image contents and these are often called high level features or high level semantics. Examples are 'grass', 'building' or 'flag'. The higher level features are difficult to measure directly from the image contents and often need to be trained with examples to be usable as a feature.

As mentioned in the previous chapter, the bridge between these two representations is called the semantic gap and there is still no clear solution on how to define a high level feature in terms of low level features. One might question if there will ever be an unambiguous way of representing high level concepts with low level features.

2.2 Color features

2.2.1 Color histogram

A color histogram represents the distribution of colors in the image. For example, if we take an image with RGB pixel values in the range [0, 255], a histogram of the distribution of these RGB values can be created with 64 bins by quantizing the color information for each channel into 4 ranges: [0 . . . 63], [64 . . . 127], [128 . . . 191], [192 . . . 255]. In other words, this is the same as reducing the bits per channel to 2 and then using the combined 6 bit RGB value as an index in the histogram.

The color space used for the histogram is arbitrary, although it has been shown that using the YUV space has better retrieval performance than the RGB space [74]. Also, the number of bits per channel can be of influence to the performance of the feature.

2.2.2 Color moments

Color moments are also based on the distribution of color values in the image, but this feature tries to capture the distribution in just a few parameters. In statistics, the n-th central moment μ_n of a random variable X or a probability density function $f(x)$ with mean μ is:

$$\mu_n = E[(X - E[X])^n] = \int_{\infty}^{\infty} (x - \mu)^n f(x) dx \tag{2.1}$$

The first central moment μ_1 is defined as zero. The second central moment is equal to the variance of the distribution and the third central moment is termed the skewness, a measure of symmetry for the distribution.

The fourth central moment is the kurtosis of the distribution, a value representing the type of measurements that resulted in the given variance. Higher kurtosis means that the variance is the result of a small number of more extreme measurements, instead of a larger number of measurements with lower variance.

In this research we have used the second, third and fourth central moment of the distribution of color values as a low level feature, which again can be applied to each of the individual color channels of the color space that is used. Note that these moments can also be used on grayscale values, which then results in a feature that is on the boundary of color (intensity) and texture.

2.3 Texture features

2.3.1 Local binary patterns

The Local Binary Patterns (LBP) texture feature is a feature that was introduced by Harwood [24] and Ojala [60] and that is invariant to monotonic changes in gray scale. The basis of the feature is the distribution of grayscale differences in regions of 3x3 pixels. The center pixel in the 3x3 region is used as a threshold for the other 8 pixels and each of these pixels is then converted to a binary value and then multiplied by a fixed value based on its location in the region, after which they are summed to get the LBP value for the 3x3 region.

$$LBP = \sum_i 2^i | I_i > threshold \tag{2.2}$$

Where i ranges over the 8 locations mentioned before. In Figure 2.1, an example is shown of how the LBP value for a 3x3 region is calculated. In this case, the final LBP value is 25.

9	3	4
7	5	6
2	4	4

a

1	0	0
1		1
0	0	0

b

1	2	4
8		16
32	64	128

c

1	0	0
8		16
0	0	0

d

Figure 2.1: a) an example of a 3x3 region with grayscale values, b) the thresholded and converted values of the region, c) the fixed values for each pixel location, d) the values that are taken into account for the overall LBP value

The distribution of LBP values over an image results in a histogram with 256 bins and this histogram can be used for similarity comparisons.

2.3.2 Symmetric covariance

The symmetric covariance texture measure was published in 1993 by Harwood [24]. It forms a histogram of values that are calculated for 3x3 regions of pixels, very much like the LBP texture measure, only this feature focuses on the pair-wise differences of two pixels in the neighborhood, instead of looking at the entire 3x3 region.

Given a 3x3 neighborhood of a pixel as seen in figure 2.2, the SCOV value is defined as:

g_2	g_3	g_4
g_1		g_1'
g_4'	g_3'	g_2'

Figure 2.2: A 3x3 region around a pixel, with each surrounding pixel labeled

$$SCOV = \frac{1}{4}\sum_{i=1}^{4}(g_i - \mu)(g_i^{'} - \mu) \tag{2.3}$$

where μ is the mean grayscale value of the 3x3 region.

2.3.3 Gray level differences

Ojala et al. [60] describe four different texture features, based on the absolute gray level differences of neighboring pixels. The simplest two, DIFFX and DIFFY, create a histogram of the absolute differences in horizontal and vertical directions. The DIFF2 feature creates one histogram for both the horizontal and vertical directions and DIFF4 also includes the diagonal directions. DIFF4 is therefore rotational invariant (with respect to 45 degree angles).

2.4 Feature vector similarity

When comparing image features, there are several methods for calculating the similarity. Examples of commonly used similarity measures are:

- L_P, where P can be $1, 2, \ldots, \infty$.

$$d_{L^P}(X, Y) = \left(\sum_{i=1}^{n}|x_i - y_i|^P\right)^{\frac{1}{P}} \tag{2.4}$$

  For $P = 1$, this results in the sum of absolute differences and for $P = 2$, it is the Euclidean distance, or the commonly used distance between two vectors in geometry.

- EMD, or earth-movers-distance. The EMD computes the difference between two distributions in terms of the amount of work it takes to redistribute the values in one distribution to end up with the values of the second distribution. It is defined as

$$EMD(P, Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{2.5}$$

where d_{ij} is the distance between two elements of the distribution and f_{ij} is taken from the flow $F = [f_{ij}]$, that is the result of minimizing:

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij} \tag{2.6}$$

However, although these methods do result in a ranking of search results, there is not much intuition behind using these formulas if one does not take a very close look at what feature values really represent. For example, when a YUV feature with three elements is used, does a difference of 0.1 in Y represent the same visual difference as a difference of 0.1 in U? The L_1 distance assumes this, but it is clearly not intuitive.

Chapter 3

Machine Learning

This chapter gives an overview of the automated learning techniques that were used in this thesis. First, a short introduction to machine learning for classification is given and then all methods used are described in detail.

3.1 Introduction

Searching for images in a database requires some form of similarity measure to determine if an image is a match to the query. The result of the search is then a list of images, ranked by similarity to the query image. The similarity can be calculated in many ways, for example by using the low level features and the feature similarity measures that were mentioned in the previous chapter. However, classification methods based on high level semantics are also commonly used. A classifier would then be an algorithm that can answer a question like 'Does this image contain grass?'

First, we introduce some mathematical notation for describing the datasets that we have used in the machine learning tasks. We denote a dataset by D, one data point is represented by x, the number of dimensions of a data point is denoted by n, the number of data points in the set by m and the classification of a data point as c. Note that the features and the resulting feature vectors we have described in the previous chapter, can be concatenated to form one large vector that forms one data point in the dataset.

A binary classification problem can be seen as a mapping between data point and two possible outputs. We define these outputs as -1 and 1. The formal definition of a dataset with binary labels can then be given as:

$$D = \{(x_i, c_i) | x_i \in \mathbb{R}^n, c_i \in \{-1, 1\}\}_{i=1}^{m} \tag{3.1}$$

3.1.1 A sample binary classification problem

This section shows an example of a toy binary classification problem. In this example, the objective is to determine if a car is a sports car. Given a few properties of a car, we would like to automatically determine if the car is a sports car.

First, let us take a look at a the example in table 3.1.

Car	Weight	Engine displ.	Supercharger	Sports car?
Audi TT	1290	1.8	yes	yes
DAF Truck	4000	8.0	no	no
Ford Focus	1200	2.0	no	no
Ferrari	1500	4.0	no	yes

Table 3.1: A sample dataset for binary classification.

This short list of examples can be used to train a binary classifier. After training, the classifier would then hopefully be able to classify new samples based on weight, engine displacement and the presence of a supercharger. If we would present a new sample to the classifier, for example $(900, 1.4, no)$, then the classifier would probably output that this is not a sports car.

The following sections demonstrate a few classification techniques that were used in this thesis.

3.2 *k*-nearest neighbor

The k-nearest neighbor classification algorithm is an algorithm that needs to keep all training data within reach when classifying new examples. First, the algorithm needs a distance function for objects that need to be classified. This function is then used to calculate the distance between the new sample and all training samples.

The simplest form of nearest-neighbor classification is to find the closest matching training sample and to classify the new sample with the same classification that this closest training sample has. However, a more robust version of this is to use a few of the closest training samples, to see if they all have the same classification.

The k in k-nearest-neighbor classification stands for the number of close training samples that are used in determining which label to assign to the new sample. In case of a 3-nearest-neighbor classification, the three closest training samples are selected and the label that has the highest occurrence is selected as the label for the new sample. Figure 3.1 illustrates this.

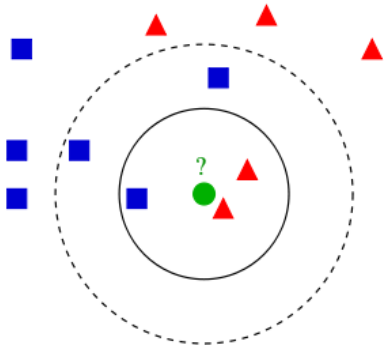


Figure 3.1: Example of the k-nearest neighbor classification. Based on the three nearest neighbors, the input will be classified as the type represented by the triangles.

3.3 Artifical neural networks

An artificial neural network is a biologically inspired method for learning mathematical functions. Neural networks have many more applications than binary classification, but they are well suited for them. Several recommended surveys of neural network research are [13] [18] [98].

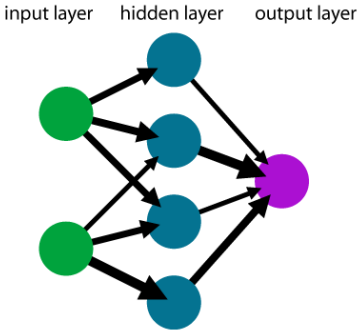


Figure 3.2: An example of a simple three layer neural network with seven artificial neurons. The thickness of an arrow represents the weight of the connection.

Neural networks are based on a simple computational element, which is used in a network-like structure to perform complex computations. This simple element is called an artificial neuron, a simplified model of the main component of the human brain, the neuron.

An artificial neuron can have several weighted inputs and the sum of these inputs is fed into an activation function to determine the output of the neuron.

$$output = f\left(\sum_{i=1}^{n} x_i w_i\right) \tag{3.2}$$

where x_i is input value i and w_i is the weight for input i. Inputs usually have a value between -1 and 1. The activation function is a function that outputs a value between -1 and 1, based on the input value. There are several options for this activation function, but a sigmoid is a common choice.

A combination of several artificial neurons that are connected to each other, results in a neural network. Some neurons process the inputs and other neurons process the outputs of these input neurons. These neurons are usually organized in layers and in each successive layer, the number of neurons decreases.

For our binary classification problem, a neural network that ends in just one neuron can be used. The network learns to classify samples by applying a learning algorithm such as the back-propagation algorithm to a set of training samples. If a well-suited network size is chosen, the network will generalize the training samples and it can then be used to classify new samples.

3.4 Support vector machines

Support vector machines, or SVMs in short, is a technique that was developed by Vladimir Vapnik [90]. The basic idea is that the input data are handled as vectors in a vector space and that a hyperplane is determined that best separates the positively labeled input vectors from the negative input vectors. The hyperplane is said to have maximum margin, as it forms the best possible separation of the two classes and has maximum distance to the closest vectors of each class.

To find this maximum margin hyperplane, two other hyperplanes are used that are placed at the boundaries of both classes. By maximizing the distance between these two hyperplanes, the resulting maximum margin hyperplane can be determined.

Non-linear classification is accomplished by transforming the vector space with a given kernel function and trying to find the hyperplane in this new vector space. If the kernel function is not linear, the resulting hyperplane in the transformed space is in fact a representation of a non-linear shape in the original vector space.

A good tutorial can be found in [4]. We have used a library by Joachims [34] in our experiments.

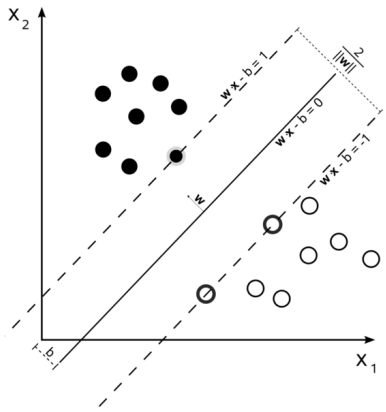


Figure 3.3: An example of the maximum margin hyperplane that was found after training the support vector machine. Vectors on the two margins are called the support vectors.

Chapter 4

Performance Evaluation

For testing and comparing the effectiveness of retrieval and classification methods, ways of evaluating the performance are required. This chapter discusses several of these methods, such as precision, recall, precision-recall graphs and average precision. Note that we use the term 'documents' in the descriptions because most of these methods were originally designed for evaluating text search engines, but in evaluating the performance of CBIR systems, 'documents' can be directly translated to 'images'.

4.1 Precision

In a retrieval task, precision is defined as the number of relevant documents retrieved as a fraction of the total number of documents retrieved:

$$precision = \frac{\#_{retrieved\ and\ relevant}}{\#_{retrieved}} \tag{4.1}$$

Precision values can be between 0.0 and 1.0. As an example, suppose a query returns 10 search results and 4 of these results are relevant for the user, then the precision of this result set is said to be 0.4. Note that a precision value always needs the number of documents in the result set to be meaningful in comparisons. A precision of 0.5 when returning just two documents gives a different perspective than a precision of 0.5 when the retrieval system returns 100 documents.

For a classification task, the precision is defined as:

$$precision = \frac{\#_{true\ positives}}{\#_{true\ positives} + \#_{false\ positives}} \tag{4.2}$$

This states that the precision of a classifier with respect to positive classification is the fraction of correctly classified positives in the total number of positively classified documents.

4.2 Recall

Recall is another measure of the effectiveness of a retrieval method. Recall is defined as the number of relevant documents retrieved as a fraction of the total number of relevant documents that are in the database:

$$recall = \frac{\#_{retrieved\ and\ relevant}}{\#_{relevant\ in\ database}} \tag{4.3}$$

Recall values have the same range as precision values, between 0.0 and 1.0. As an example, if a retrieval result set with 10 documents contains 4 relevant documents and there are 40 relevant documents in the database, the recall value for this result set is 0.1. Note that the recall value also needs to be accompanied by the number of documents that were retrieved. Also note that the recall value will always be 1.0 if the entire database is returned as a result set in response to a query.

In a classification task, the definition of recall is given by:

$$recall = \frac{\#_{true\ positives}}{\#_{true\ positives} + \#_{false\ negatives}} \tag{4.4}$$

This formula tells us the fraction of positively classified documents with respect to all positives that are in the data set.

4.3 Precision-Recall graphs

To give a graphical impression of the performance of a retrieval method, precision-recall graphs can be created. (Some researchers refer to them as recall-precision graphs, probably a more correct naming, but the term precision-recall is most widely used.) To generate such a graph, a single query is repeatedly executed and the number of returned results is varied. For each of these results sets, the precision and recall are determined and both these values are plotted as a single coordinate in the graph. The shape of the resulting graph gives a quick indication of the performance of a retrieval method. Note that for very low recall values, it is often not very elegant to plot the corresponding precision values, since the result sets used are probably very small and the values can fluctuate rapidly. It is common to start the graph at a recall value of 0.1.

First, the two extreme shapes of a precision-recall graph will be discussed. The ideal shape of a precision-recall graph would be the situation where all returned

documents are always relevant. For each recall value, the precision would always be 1.0. Only if no more relevant documents can be found, the search results will contain irrelevant documents. Note that the graph has only been plotted up to the first recall value of 1.0.

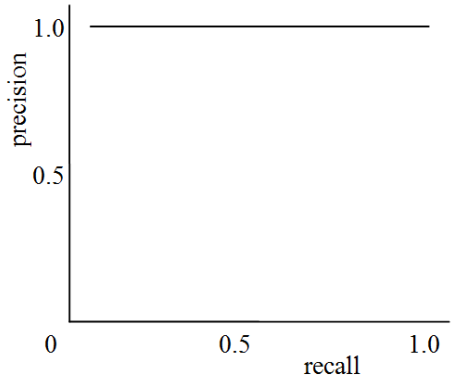


Figure 4.1: The optimal precision-recall graph, with every precision value at 1.0.

The worst case scenario would be when all relevant results only show up after all irrelevant documents have been returned. In this case, when recall values increase from 0.0 to 1.0, the precision would increase slowly from 0.0 to a value specific for the database. (Note that we can plot the coordinate 0.0, 0.0 now.) Assuming that the database contains 100 documents and 10 relevant documents, the final precision would be 0.1 for a recall value of 1.0.

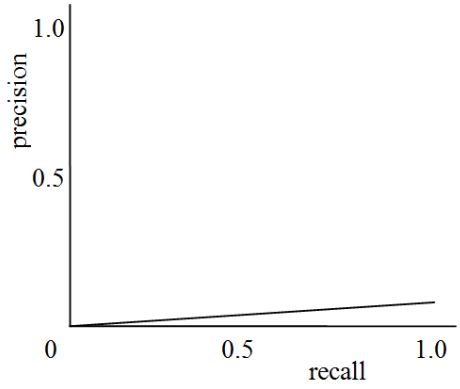


Figure 4.2: The worst case scenario for a precision-recall graph: all relevant documents are ranked lowest.

Some examples of precision-recall graphs are given below, together with an explanation on how to read these graphs.

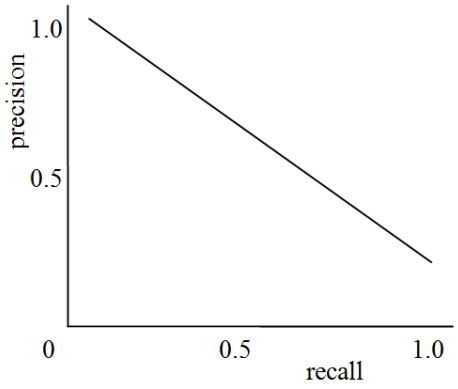


Figure 4.3: A linear relation between recall and precision.

This graph shows a linear relation between precision and recall. For this graph, with increasing recall, precision decreases until the point where all relevant documents are retrieved. This graph tells us that for any given number of results, the percentage of relevant documents has an inverse linear relation with the number of retrieved documents.

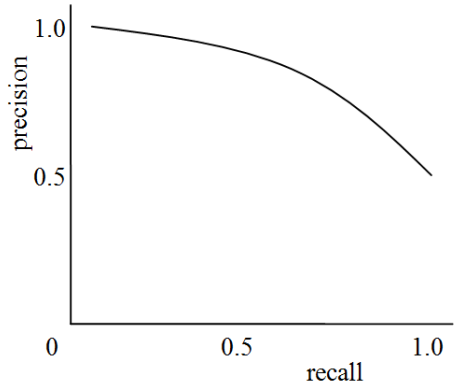


Figure 4.4: A precision-recall graph that indicates high retrieval precision.

This graph shows higher precision values for lower recall values, compared to the first graph. This tells us that for short result sets, the precision is very high, but as the number of retrieved documents increases, the precision decreases. In other words, this probably means that most of the relevant are results are returned in

the top of the result set, but some of the relevant documents are not detected by the retrieval method and are mixed with the rest of the results.

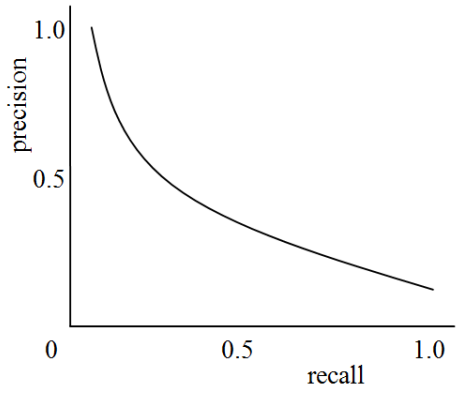


Figure 4.5: A precision-recall graph that indicates a rapidly decreasing precision with increasing result set sizes.

This graph shows a lower precision at low recall levels than the first graph. This translates to the notion that the percentage of relevant documents in the search results will decrease sharply when the number of search results is increased. In other words, many irrelevant documents show up in the top results.

4.4 Average precision

Another method for measuring the performance of a retrieval method is the average precision. This is a value that does not need a fixed length of the result set to be usable in comparisons. The average precision is calculated by averaging the precision values at each relevant document in the result set, usually up to the point where recall is 1.0.

Assume the last relevant document is retrieved at position N in the result set and that the function *relevant* returns 1 when a document is relevant and that *precision* returns the precision of the result set up to a certain point. Then the formula for the average precision can be given as:

$$average\ precision = \frac{\sum_{i=1}^{N} relevant(i) precision(i)}{\sum_{i=1}^{N} relevant(i)} \tag{4.5}$$

With this value, the overall performance of a retrieval method can be assessed with one number, without the need for a graph or a fixed number of returned documents.

4.5 Accuracy

In classification tasks, another measure is available to determine the performance of the classifier: the accuracy. It is defined as

$$accuracy = \frac{\#_{true\ positives} + \#_{true\ negatives}}{\#_{true\ positives} + \#_{true\ negatives} + \#_{false\ positives} + \#_{false\ negatives}} \tag{4.6}$$

This yields the fraction of correctly classified documents with respect to the total number of documents. It can be seen as the probability that a classification, either positive or negative, is correct.

Chapter 5

Interest Points Based on Maximization of Distinctiveness

Interest or salient points are typically meaningful points within an image which can be used for a wide variety of image understanding tasks. In this chapter we present a novel algorithm for detecting interest points within images. The new technique is based on finding the locations in an image which exhibit local distinctiveness. We evaluate our algorithm on the Corel stock photography test set in the context of content based image retrieval from large databases and provide quantitative comparisons to the well known SIFT interest point and Harris corner detectors as a benchmark.

5.1 Introduction

In a typical content-based image retrieval [40] task, image features are compared for matching images. When the image features are close, it is assumed the images are similar. These features can be computed globally (over the entire image) or locally (over small parts of the image). For locally computed image features, it is necessary to determine which image points should be used for describing the image content. These image points are called interest points and various methods exist to select these points.

We introduce a novel method of computing interest points based on local uniqueness and evaluate the effectiveness.

5.2 Related work

Many interest point detectors are available [1] [40] [44] [43] [51] [52] [89], and depending on the application, different performance measures can be chosen. Arguably, the original interest point detector was created by Moravec [52] who needed to find extremely computationally efficient methods for performing real time robotic navigation. In the 70s, it was impossible to perform real time video analysis on a mobile computer so his necessity led to the invention of interest points. In current times, there are now other data intensive tasks, one of which is content based image retrieval from large databases typically measured in the thousands to millions of images. In this image retrieval context, it is again important to have information efficient descriptors to perform content based searches in a user acceptable response time.

A good overview is given in Sebe, et al. [72] and also Schmid, et al. [71]. In these works, it is clear that one of the best performing interest or salient point detectors is the Harris corner detector. The Harris corner detector [23] is an interest point detector that is invariant to rotation, scale and illumination changes. It uses the auto-correlation function for comparing a small part of an image to the area around it. SIFT [44] [43] features are invariant to changes in scale and rotation. Trujillo and Olague [89] use genetic programming to detect salient points.

There are a wide variety of methods of evaluating different interest point detectors. Schmid et al. [71] use two evaluation criteria for interest points: repeatability rate and information content. The former criterion determines the stability of the interest point under various transformations. The latter is a measure of the distribution of the feature values for those interest points. A distribution that is spread out indicates more information content. Sebe, et al. [80] suggest a good measure is using the information content as measured by the average information content of all messages or the entropy. Tian et al. [88] use the retrieval accuracy in a content based image retrieval task to evaluate their wavelet-based salient point method. So far we have briefly discussed what different methods exist. In the next section we will discuss why different methods are interesting and explain the fundamental motivation behind our own interest/salient point paradigm.

5.3 Maximization Of Distinctiveness (MOD)

In Moravecs [52] original interest operator from 1979, the main intuition was to use points which had high x and y gradients which in principle would be distinctive just as there are typically far fewer edge pixels than non-edge pixels within an image.

Nearly a decade later, Harris [23] came up with a robust method for detecting corners which also had high x and y gradients and are an intuitive method for

salient point detection. The usage of Moravecs and Harris work as salient points was intuitive but also in our opinion adhoc.

Recently, Lowe [44] proposed the Scale Invariant Feature Transform (SIFT) method which focuses on looking through the neighboring scales to find extrema in the difference of the Gaussian which is an approximation to the Laplacian of the Gaussian. The fundamental notion was to find scale invariant interest points on the assumption that scale invariance was important to good features:

L = set of extrema in the Laplacian of the Gaussian
S = set of stable points in L
H = set of higher contrast points in S (see page 98 of [44])
P = set of H where edge responses are eliminated.

In addition to P, one or more orientations are assigned to each element of P and a descriptor is computed using the gradient magnitudes and orientations for each level of the image pyramid where the orientations are adjusted for the assigned orientation.

While we agree that the scale invariance is a useful aspect of a good feature, it depends on the particular context. In many areas such as texture classification, image retrieval, video retrieval, stereo matching, and motion estimation contexts, the scale is assumed to be very similar between the correspondences or between the query and the results images. For example, in texture classification, one does not want to match a fine grain texture with a coarse grain texture. In summary, there are many areas where the scale is important and where the variation of scale is beneficial in reducing candidates and maximizing the accuracy of a matching algorithm.

5.3.1 The MOD paradigm

In the scale dependent visual matching areas such as stereo matching or motion estimation, the typical techniques are variations of feature vector matching, of which the most popular and intuitive method would be template matching.

Unlike the SIFT method which strives to find points which are scale invariant, we strive to find points which optimize distinctiveness in matching. In matching, the most common problem to address is the one to many mapping, where one point may have many good matches. We first compute a distinctiveness of matching measure which finds the most distinctive point in a local region based on the matching method to be finally used. This means searching the neighboring region around a point, computing the dissimilarity to each point in the region and then estimating the distinctiveness as the minimum of the dissimilarity values in the region. One benefit this has beyond the local gradient interest point methods is that it can remove points which are only similar from the perspective of the matching algorithm.

We were motivated to design a new paradigm for salient point detection which would both be intuitive, adaptable to diverse image matching fields, and be centered on optimizing a criterion function.

Our fundamental notion is that we want to minimize the probability of a mismatch when we select a match based on a distortion or dissimilarity measure. Therefore, we want to select salient points which will have a lower number of similar candidates in any local region.

We assume for simplicity that the distinctiveness of a point is inversely related to the similarity of that point to the closest wrong match.

Let $D(x, y)$ represent the distinctiveness of a pixel at (x, y). Then the function we are optimizing can be expressed elegantly as

$$D(P) = \underset{R}{\operatorname{argmin}}[-Similarity(R, P)] \tag{5.1}$$

where R represents a region based on a pixel location P.

and the constellation of salient points would be

$$C = maxima\ of\ D(P) \tag{5.2}$$

This means that we select the set of pixels which are local maxima of distinctiveness with regard to the similarity function used in the matching algorithm.

One of the advantages of this paradigm is that the similarity function can be adapted to the area of computer vision or pattern recognition. It can be adapted to be rotation invariant, scale invariant, color invariant, etc. As mentioned earlier, different problem areas have different constraints.

Another advantage of the MOD paradigm is that the similarity function can be also utilize sets of imagery such as all of the frames in a video shot because R can include the region near a pixel in the video shot specified as (x, y, t) where t represents the frame number. This would mean that we could find all of the salient pixels over a video shot, not merely a single frame.

5.3.2 The special case of template matching

The implementation of interest point method for the special case of template matching is based on selecting points in the image that maximize local distinctiveness. In this case, the distinctiveness value of a point is determined by the distance to the best matching neighbor in an area surrounding that point. We determine the distance between two points by calculating the SAD of grayscale values in a square template window.



Figure 5.1: Example images from the 'aviation' class.

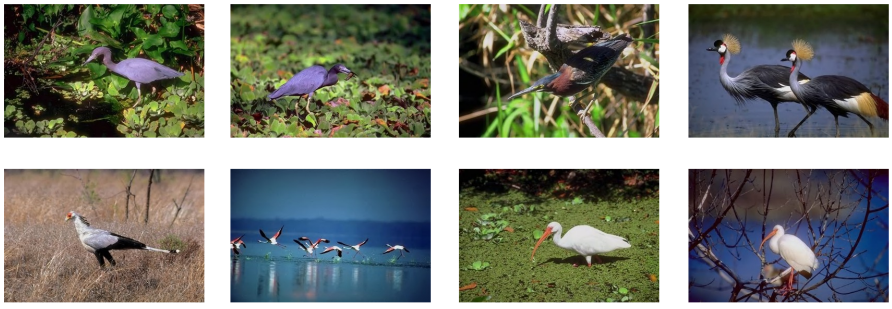


Figure 5.2: Example images from the 'wl_bird1' class.

5.3.3 Detector output

Figures 5.1 to 5.3 show example images from three classes of the Corel database.



Figure 5.3: Example images from the 'dogs' class.

Figures 5.4 to 5.12 show the output of the Harris corner detector, the SIFT interest point detector and the MOD interest point detector for the same input images. Visually, the Harris detector seems to capture the structure of the objects in the image better than the MOD detector, but we will show the effect of this for the retrieval results in section 5.5.

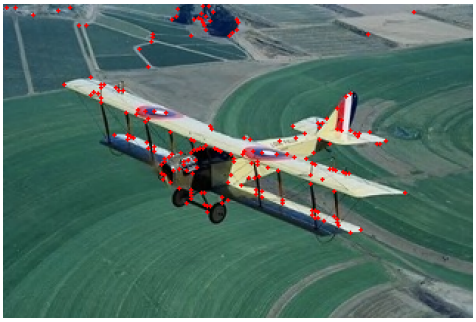


Figure 5.4: An image with the output of the Harris corner detector.

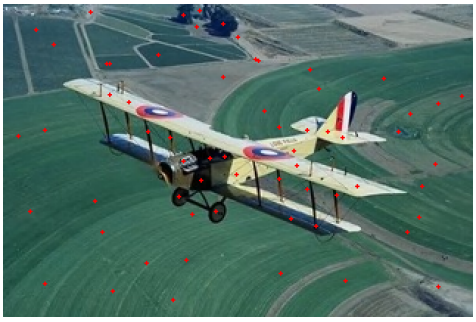


Figure 5.5: An image with the output of the SIFT interest point detector.

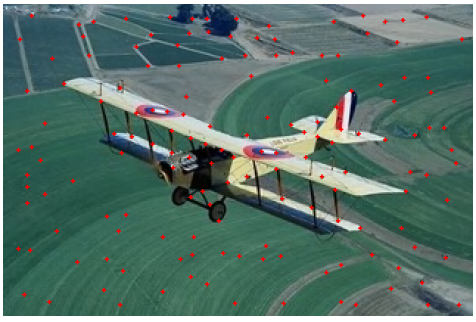


Figure 5.6: An image with the output of the MOD interest point detector.

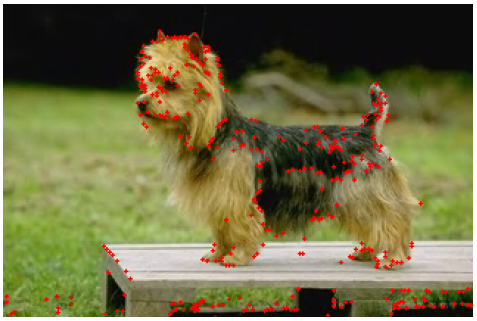


Figure 5.7: An image with the output of the Harris corner detector.

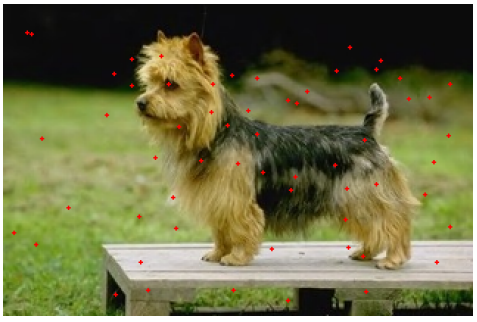


Figure 5.8: An image with the output of the SIFT interest point detector.

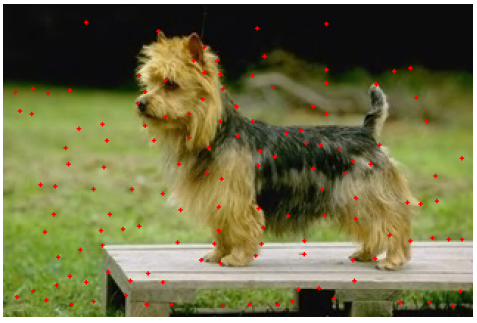


Figure 5.9: An image with the output of the MOD interest point detector.

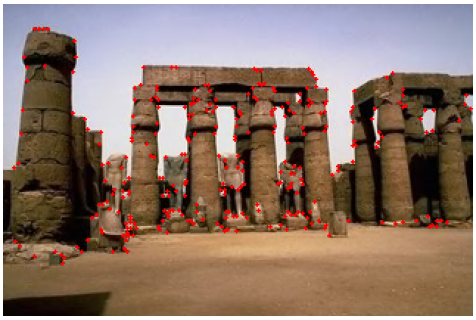


Figure 5.10: An image with the output of the Harris corner detector.



Figure 5.11: An image with the output of the SIFT interest point detector.



Figure 5.12: An image with the output of the MOD interest point detector.

5.4 Matching images

For each interest point, a feature vector is created that contains color and texture information. We have chosen color moments [87] as the color feature and local binary patterns [61] as the texture feature. Color moments are calculated for a 3x3 region around the interest point. The local binary patterns feature is calculated for a 19x19 region around the interest point.

We then use these feature vectors to compare images, by determining the best matches between interest points. The sum of all these best matches is the distance between the images.

In other words, for each point in an image I, the closest matching point in image J is searched for and this distance is used for calculating the overall distance of image I to image J. The distance between two images I and J is defined as the sum of the distance from I to J and the distance from J to I:

$$d(I,J) = \sum_x bestmatch(I_x, J) + \sum_y bestmatch(J_y, I) \tag{5.3}$$

where I_x is interest point x in image I.

5.5 Experiments and results

For testing the new salient points method, we used a subset of the Corel photo database. We selected 18 classes of images, each class containing 100 images. We have used 20 randomly selected images from each class as the query images. The results are averaged over these 20 queries.

Each image was resized to have a width of 320 pixels, to speed up computation and to make sure the calculated features are more scale-invariant. The settings for our interest point detector were set to a neighborhood size of 15 and a template size of 3, which was empirically determined by looking at the detector output.

Table 5.1 gives an overview of retrieval accuracy for the first 15 images returned, for the three interest point detectors.

Image class	Harris	SIFT	MOD
aviation	0.110	0.370	0.390
beaches	0.213	0.597	0.320
butterfly	0.140	0.157	0.307
cactus	0.127	0.303	0.257
castles	0.147	0.137	0.150
cats	0.243	0.267	0.493
dogs	0.207	0.157	0,333
horses	0.070	0.067	0.080
mammals	0.103	0.077	0.130
models	0.357	0.228	0.253
mountain	0.130	0.077	0.260
orchids	0.137	0.197	0.323
pyramids	0.170	0.373	0.627
roses	0.373	0.357	0.703
tulips	0.137	0.170	0.143
waterfall	0.210	0.170	0.383
wl_bird1	0.217	0.237	0.283
wl_fish	0.120	0.333	0.309

Table 5.1: Average precision of the three methods, using 20 query images and a result set of 15 images.

Among the best results for the MOD algorithm are 'pyramids' and 'roses', of which the results are shown in Figures 5.13 and 5.14.

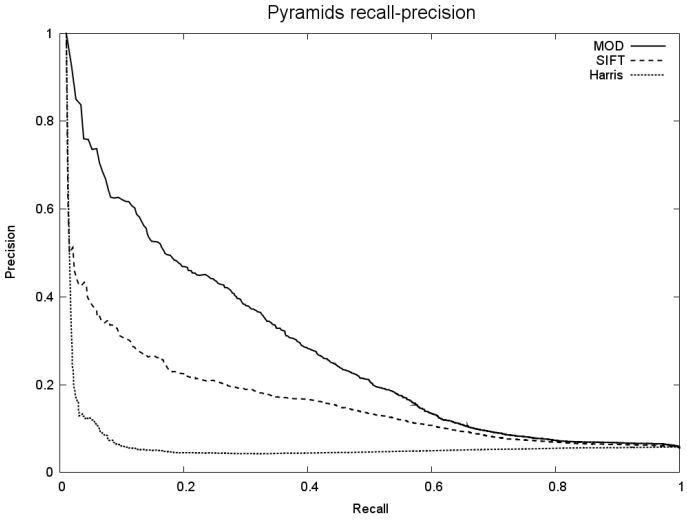


Figure 5.13: Recall-precision for the 'pyramids' class.

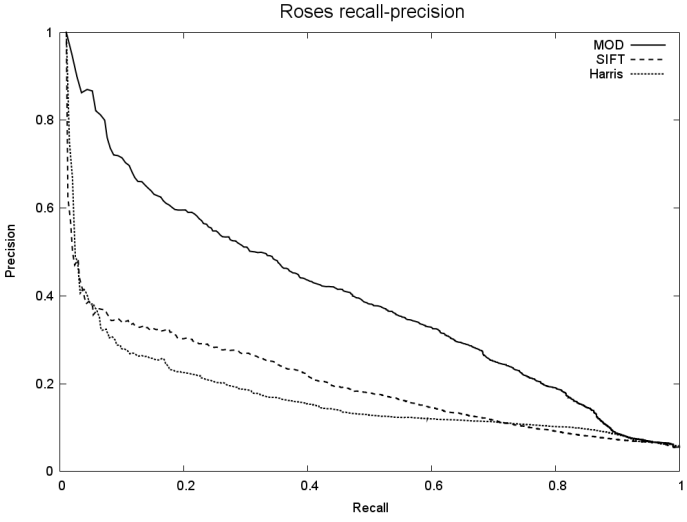


Figure 5.14: Recall-precision for the 'roses' class.

Figure 5.15 shows the overall retrieval results for our interest point method compared to the SIFT and Harris methods. It is clear that the MOD gives significantly better results in the context of image retrieval than either the SIFT method or the Harris points.

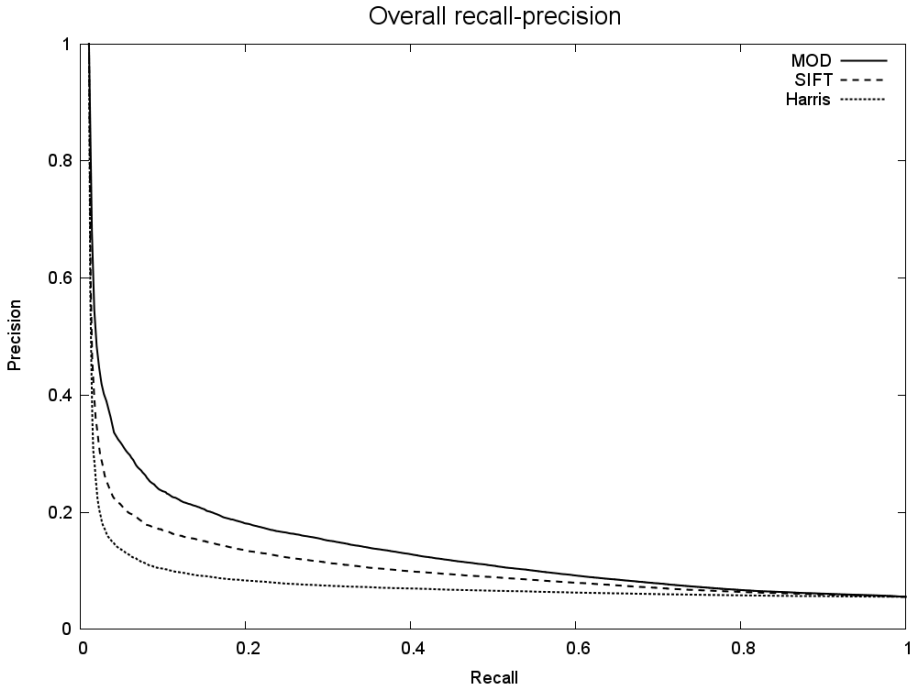


Figure 5.15: Overall retrieval recall-precision.

5.6 Discussion and conclusions

In this chapter we introduced a novel paradigm for interest point detection based on a criterion of maximization of distinctiveness. We have used two features for describing each interest point, the local binary pattern for texture and color moments for color. We compared the MOD method in a content-based image retrieval experiment to the SIFT interest point and the Harris corner detector and we have showed that it outperforms both detectors for this task.

Intuitively, our interest point detector places points at locations that are more uniform in color or texture. These points clearly are useful for content-based retrieval tasks, since these areas also contain information.

Chapter 6

Learning and Visual Concept Detection

Visual concept detection is the automated detection of image semantics. Detecting image semantics is particularly useful in content based retrieval because it allows us to annotate media with semantically meaningful information. One computationally efficient approach toward subimage annotation is to focus on regions which are considered salient. The novel contribution in this chapter is using the regions found from the maximization of distinctiveness (MOD) saliency approach for automatic visual concept detection. We present results based on real images and compare nearest neighbor classification, support vector machines and a neural network.

6.1 Introduction

Bridging the gap between low level features such as color histograms and semantic descriptions such as 'trees' is highly useful for searching image databases and digital libraries. From the panels of CIVR and MIR conferences, it has been said repeatedly that one of the primary goals of the community is automatic annotation of images and video. It would be ideal to have a program which receives as input an unknown image, analyzes the pictorial content of the image and then outputs a set of keywords. In this chapter we present our ongoing work in detecting visual concepts toward automatic image annotation.

The basis of visual concept detection is the need for automatically describing the content of images. For image retrieval tasks, these generated descriptions of images can be very useful. For example, the following image in Figure 6.1 could be classified as containing buildings and trees:



Figure 6.1: An image which could be labeled with trees, buildings and sky.

With this annotation, it is possible to search for these images using keywords. A high level overview of our system frame work is shown in Figure 6.2. In our system, a visual concept is learned interactively by our program using positive and negative responses from the user. Once the visual concept has been learned, it is applied to an unknown image and automatically outputs descriptive text.

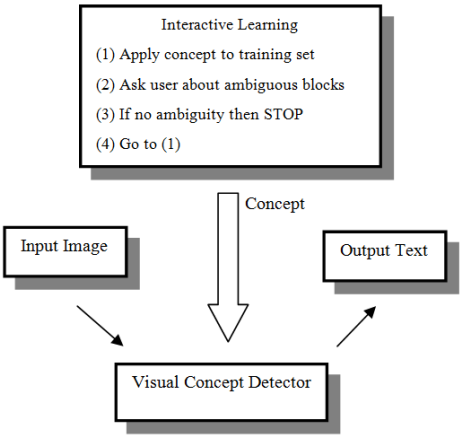


Figure 6.2: The overall visual concept detection system.

For detecting visual concepts in images, we have compared the following methods: nearest neighbor classification, support vector machines and a neural network. We also use several approaches for segmenting images into sub-images using wavelet-based salient points and maximization of distinctiveness (MOD) based interest points, of which the visual concepts are determined.

6.2 Related work

In the recent years, visual concept detection or automated image annotation has become a very active field of research. As the amount of visual information available increases, the need for new methods for searching it also increases. For example, Srikanth et al. [84] describe an image annotation method that uses an ontology for linking annotation words. They measure the retrieval performance when annotation words are connected to other words in a hierarchy. Retrieval accuracy improves using this knowledge.

Other closely related work in the research literature for visual concept detection is face detection in complex images. Representative examples would be the neural network work by Rowley and Kanade [66] or the information theoretic approach by Lew and Huijsmans [39] in which the authors detect the specific visual concept of a human face. An excellent survey of the work done in face detection is found in Yang, et al. [95].

6.3 Maximization Of Distinctiveness (MOD)

Recently, we have proposed the Maximization of Distinctiveness paradigm [56] for detecting interest points in images. These interest points have the property that they are optimized for visual content matching using feature vectors. The selected points have the highest distinctiveness with respect to certain features, in a local region around the selected point.

For computing the MOD interest point, each pixel in an image is first assigned a distinctiveness value that is based on the dissimilarity of the point to all pixels in a neighborhood around it. The dissimilarity for a pixel is estimated as the inverse of the similarity value for the closest matching point in the neighborhood, when the features are considered that will be used for matching. The dissimilarity values are combined to form a distinctiveness map of an image and the local maxima in this distinctiveness map will yield the MOD interest points. The details of this method are described in chapter 5 of this thesis.

6.4 Detecting visual concepts

We used two steps in detecting visual concepts in images: Visual concept description and visual concept matching.

First, the visual concepts need to be described. This can be done by selecting positive and negative examples and to use a classification method which uses these positive and negative examples to create a general model of the concept.

For each of the positive and negative examples, a number of feature vectors is extracted. We have used an HSV based color feature, color moments and a texture

feature created by Ojala [62], which is invariant under grayscale variations and rotation.

The second step for visual concept detection is to match each image to the visual concepts. Images are compared to the generalized model and are classified as either matching or non-matching for each visual concept. The list of matching concepts is then an annotation for the image.

For describing the visual concepts, we have used the interactively selected lists of positive and negative images.

For matching images with visual concepts, we have looked at three different methods: Nearest neighbor classification, support vector machines and a neural network. These methods are explained in detail in the next three sections. We expect classification using SVMs or the neural network to outperform the nearest neighbor method because of the generalization capabilities of these machine learning techniques, but we included the nearest neighbor method as a benchmark.

6.4.1 Classifiers

For our experiments, we have used three different classifiers. The details of each of these classifier types can be found in chapter 3 of this thesis. Here we only describe the parameters used (if any) for each of the classifiers.

First, as a benchmark, we used a k-nearest neighbor classifier that uses the label of the closest positive of negative example of a concept as a classification.

Secondly, we have used a neural network with one variable sized hidden layer and one output unit. In most experiments, the hidden layer consisted of 25 units.

The third classification technique we have used is the support vector machine (SVM). In these experiments, we have chosen to use a third order polynomial kernel, because we assume the feature vectors are not linearly separable.

6.5 Experiments

We have tested our system on a dataset of tourist-like images of the cities of Leiden and Amsterdam and several other images. This section shows the user interface of the program, the results of applying concepts to a few of these images and also the results of an application of visual concept detection to face detection.

Figure 6.3 shows the images that were selected from the tourist database for detecting trees, buildings and blue sky. For the beach and face concepts, we have selected images from the web.

Figure 6.4 shows the interface of our visual concept detection program. Users can add concepts and assign image regions to the positive or negative examples for each concept.



Figure 6.3: Images selected from the Tourist Database.

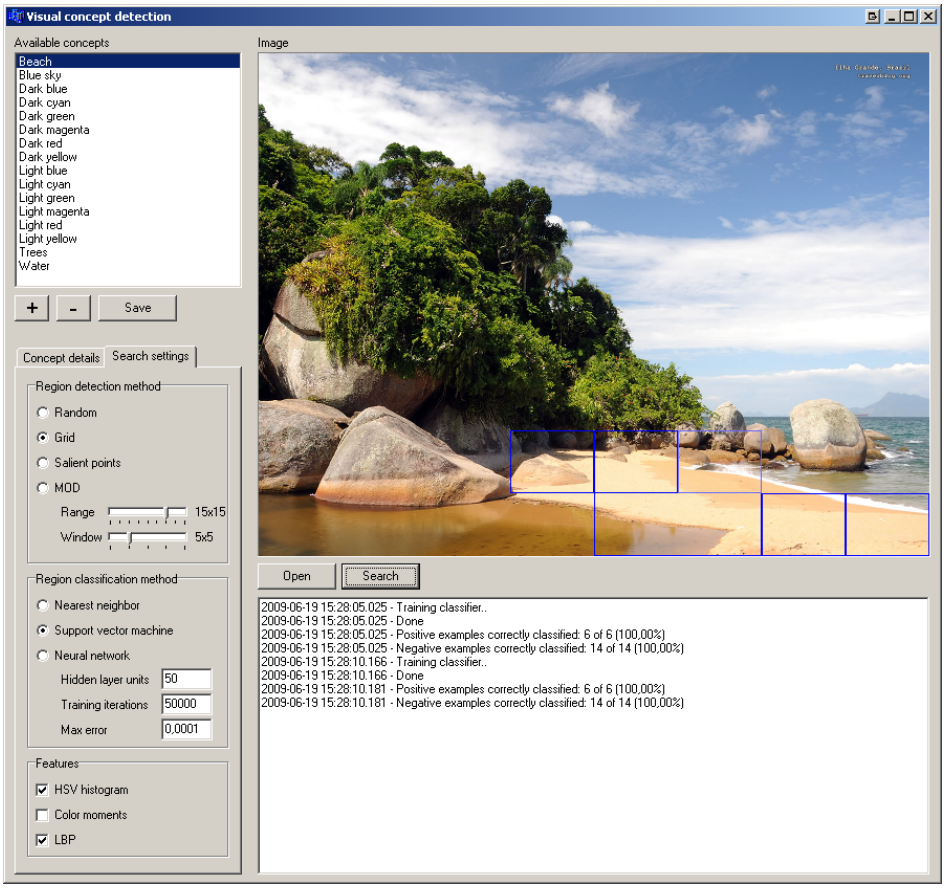


Figure 6.4: The user interface of the Visual Concept Detection program.

6.5.1 Tree detection

This section shows where the concept 'Tree' was detected in some of the Tourist Database images. Figures 6.5 to 6.7 contain the examples.

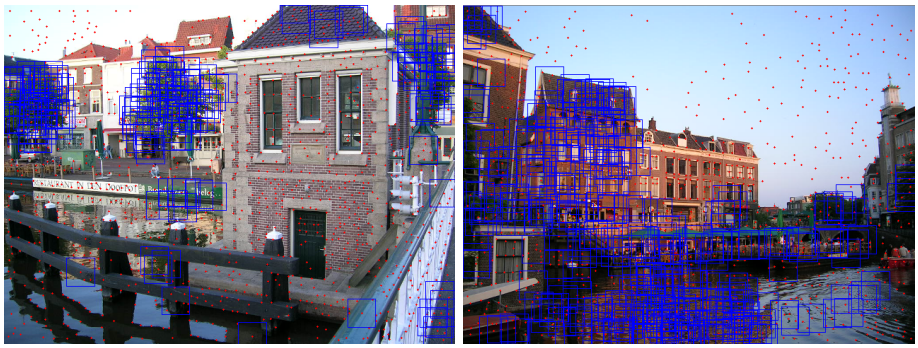


Figure 6.5: Tree detection with nearest neighbor classification.



Figure 6.6: Tree detection with SVM classification.

6.5.2 Building detection

This section shows where the concept 'Building' was detected in some of the Tourist Database images. All examples are contained in figures 6.8 to 6.10.



Figure 6.7: Tree detection with neural network classification.

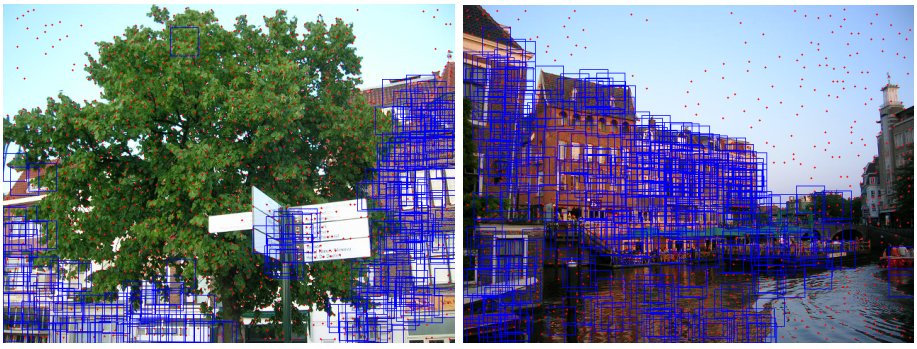


Figure 6.8: Building detection with nearest neighbor classification.

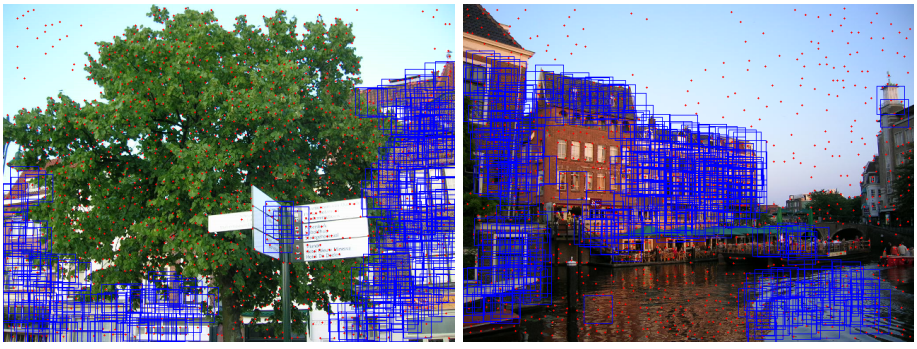


Figure 6.9: Building detection with SVM classification.

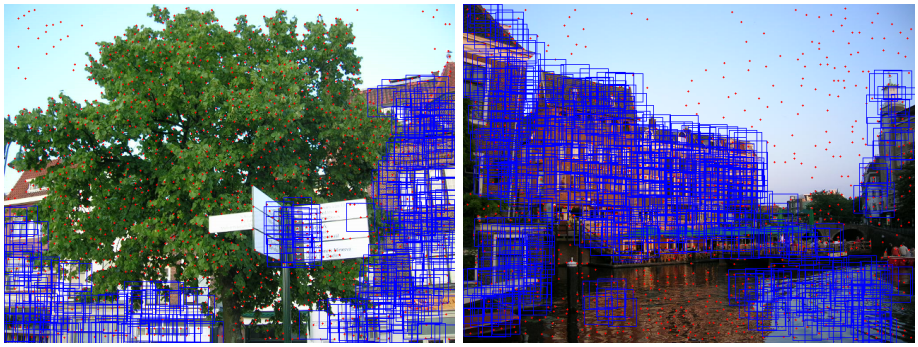


Figure 6.10: Building detection with neural network classification.

6.5.3 Sky detection

This section shows where the concept 'Sky' was detected in some of the Tourist Database images. All examples are contained in figures 6.11 to 6.13.

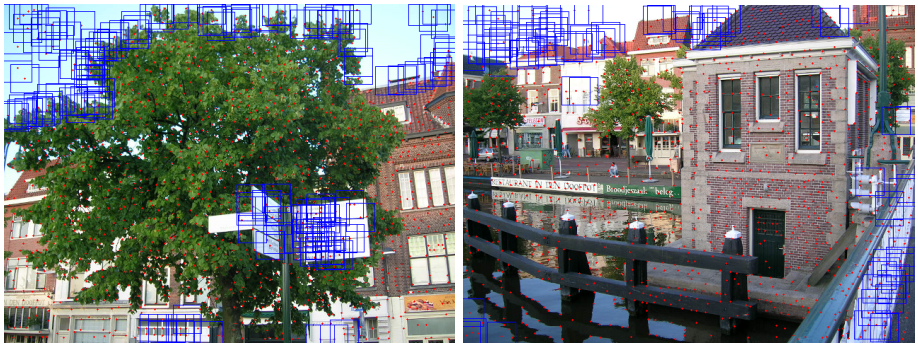


Figure 6.11: Sky detection with nearest neighbor classification.

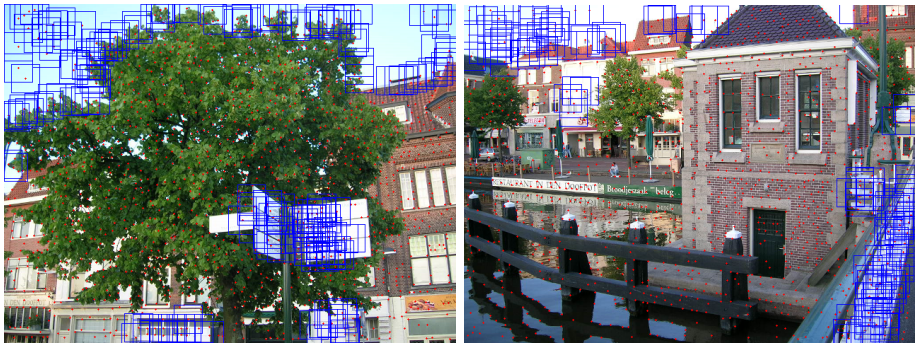


Figure 6.12: Sky detection with SVM classification.

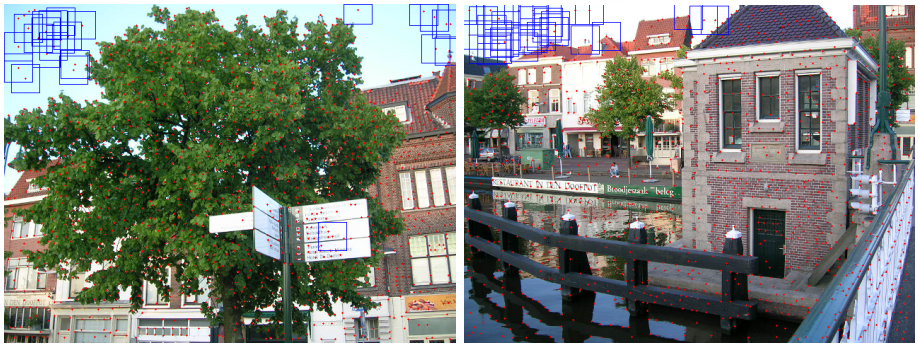


Figure 6.13: Sky detection with neural network classification.

6.5.4 Beach classification

This section shows the detection of the 'Beach' concept on an image from the web. Figure 6.14 shows the example.

6.5.5 Face detection

Figure 6.15 shows the results of applying a 'Face' concept to an image. In this case, the training image set and test set were very small, but we present this result as an example of other uses of the visual concept detection.

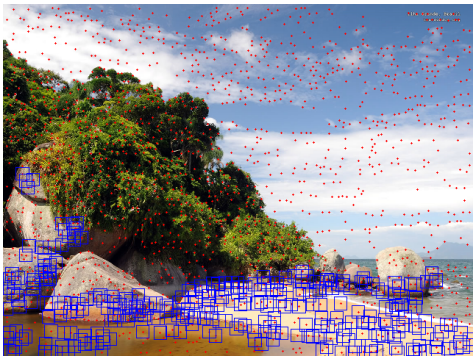


Figure 6.14: Beach detection with neural network classification.

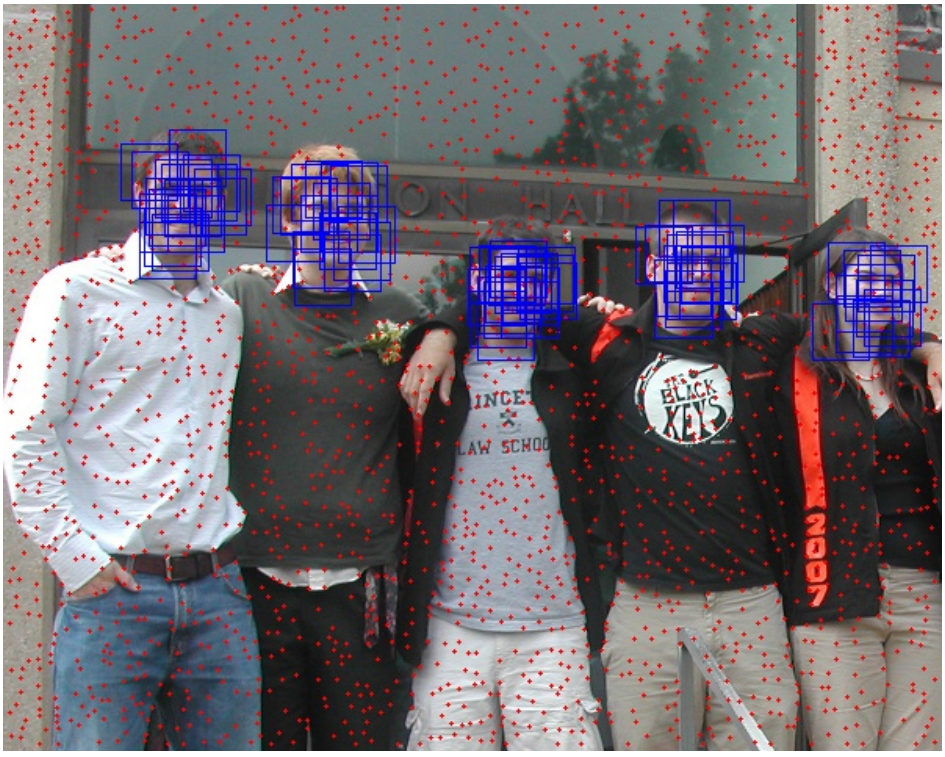


Figure 6.15: Output of the visual concept detector for the 'face' concept.

6.6 Experiments on MIRFLICKR-25000 dataset

The MIRFLICKR-25000 dataset was presented at the MIR conference held in 2008 [31]. The dataset contains 25000 images that were retrieved from the Flickr website. All original user annotations are available, as well as the EXIF metadata. Also, a ground truth is supplied for a large number of visual concepts, from general to very specific topics. For example, the concept 'people' is annotated, but also 'baby'.

The ten general topics of the dataset annotations are:

- Animals
- Indoor
- Night
- People
- Plant life
- Sky
- Structures
- Sunset
- Transport
- Water

We have used these general annotations in our experiments. We have compared the classification accuracy of SIFT [44] features with a support vector machine (SVM) as a classifier to MOD interest points with both an SVM and a neural network as classifiers with a basic set of features.

For the MOD based classification, we used the EWF, an Extended Wavelet Feature set which combines the wavelet [85] [86] [92] representation of a grayscale version of the image region [73], with the HSV Histogram [74] and Local Binary Patterns [60] [75].

For a comparative benchmark, SIFT features with an SVM classifier have shown to be a good classifier for visual concepts [3], so we have chosen to use that as a baseline method.

The SVM classifiers were constructed using a radial basis function (RBF) kernel function, which has been suggested as the optimal kernel in our context [6].

For our tests, we have selected a set of 50 training images for each concept. The training set contained 25 positive examples and 25 negative examples. Each positive example was manually segmented into regions that contain the concept. Only interest points within these regions were used in the tests.

For the test set for each concept, we have randomly selected 5000 images from the dataset, of which 50% were positively labeled and 50% were negatively labeled, although not all concepts have 2500 positive examples in the dataset. In those cases, more negatives were added.

First, we give the detailed annotation results for a few of the general concepts. We show graphs of the true positive rate compared to the true negative rate. The true positive rate is defined as the fraction of positives detected:

$$true\ positive\ rate = tp/(tp + fn) \tag{6.1}$$

Note that in a classification context, this is also known as the 'recall' value. The true negative rate is then defined as the fraction of negatives detected:

$$true\ negative\ rate = tn/(tn + fp) \tag{6.2}$$

In order to give a proper idea of the diversity and difficulty level of the concepts, we show in Figures 6.16 to 6.45 examples of interesting concepts in each subsection below along with the performance graphs and the detected salient points corresponding to the visual concept. In Figure 6.46 we show the averaged results over all of the concepts.

6.6.1 Concept 'Animals'

Figure 6.16 shows some example images for the 'Animals' concept. These images show the wide variety of images that can be classified as containing animals. Not only real animals that are clearly visible, but also hand drawn animals or parts of an animal result in the same annotation. Also note that the animal does not have to be the subject of the image, but it might also be seen in the background.



Figure 6.16: Some example images for the 'Animals' concept.

Figure 6.17 shows the classification results for the 'Animals' concept and figure 6.18 shows some detection examples of the MOD based concept detection.

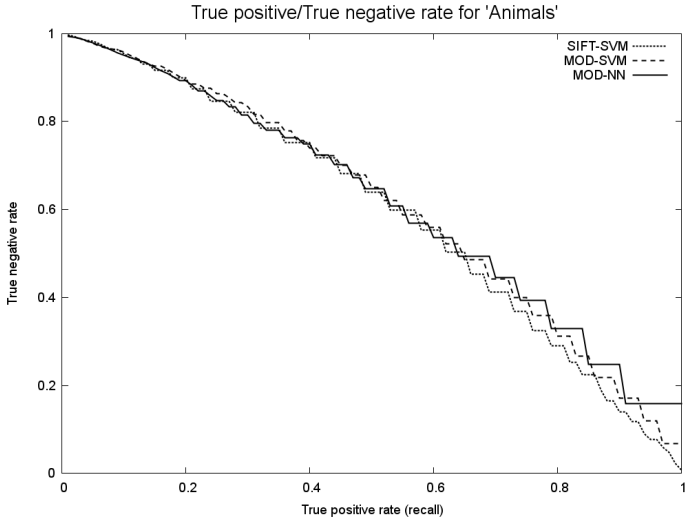


Figure 6.17: Classification results for the 'Animals' concept.



Figure 6.18: Some MOD based concept detection results for the 'Animals' concept.

As explained above, the Animals concept contains a wide variety of images and it seems that 25 training images is probably not enough to create a reliable classifier for all images with animals. As there are many more different types of animals than there are training images, a larger training set should be used.

6.6.2 Concept 'Indoor'

Figure 6.19 shows some example images for the 'Indoor' concept. The concept 'Indoor' contains images that were taken indoors. However, the images do not explicitly have to show a typical indoor situation like a room with a view to the outside world. Intuitively, the concept can be described as for example 'not containing sky' or 'not containing grass'.



Figure 6.19: Some example images for the 'Indoor' concept.

Figure 6.20 shows the classification results for the 'Indoor' concept and figure 6.21 shows some detection examples of the MOD based concept detection.

The detection of the concept 'indoor' shows to be very difficult. The real question is if there is a direct relationship between visual properties of an image region and the concept indoor. The concept cannot be pointed out in an image, although the image does get the label 'indoor'. The results show that the number of detected regions does give a hint of the probability of the image being indoor.

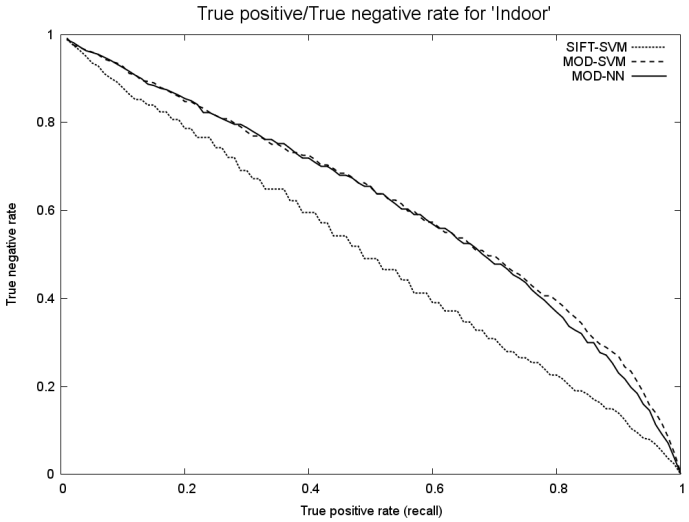


Figure 6.20: Classification results for the 'Indoor' concept.



Figure 6.21: Some MOD based concept detection results for the 'Indoor' concept.

6.6.3 Concept 'Night'

Figure 6.16 shows some example images for the 'Night' concept. The concept 'night' contains images that were taken at night. Although this sounds like a trivial concept to detect, because the images would usually be dark in color, also the contents are needed to determine if an image was really taken at night. Not all areas of an image need to be dark for it to be taken at night and not all images with many dark areas were taken at night.



Figure 6.22: Some example images for the 'Night' concept.

Figure 6.23 shows the classification results for the 'Night' concept and figure 6.24 shows some detection examples of the MOD based concept detection.

The results of the MOD based detector show that many image areas that are not dark, are still classified as being part of the night concept. We anticipated this earlier when discussing what images in this concept class would look like. The retrieval performance of the MOD based methods clearly outperform the SIFT based method.

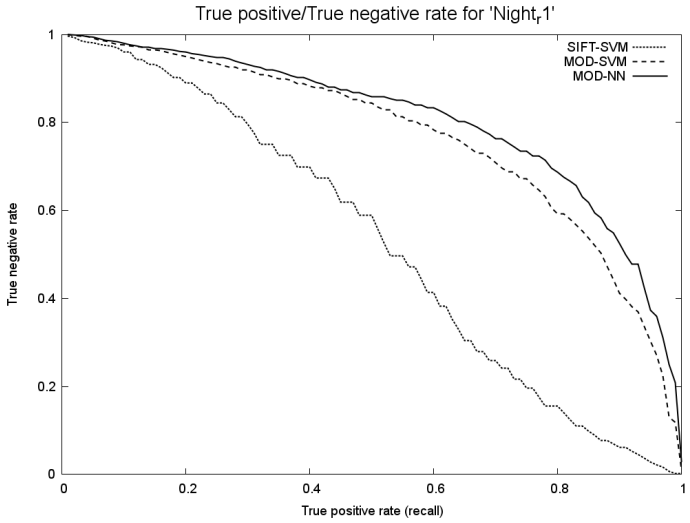


Figure 6.23: Classification results for the 'Night' concept.

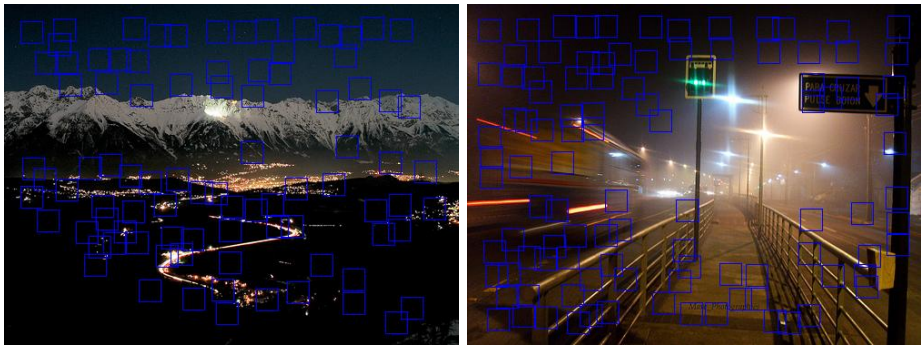


Figure 6.24: Some MOD based concept detection results for the 'Night' concept.

6.6.4 Concept 'People'

Figure 6.25 shows some example images for the 'People' concept. In the MIRFLICKR-25000 dataset, there are two different annotations for the 'People' concept. The annotation 'people' is a less strict annotation than 'people_r1', in which one or more persons are really clearly visible, probably even the subject of the picture. We have used the 'people_r1' annotation. The example images show that in some images only a face is visible, but in other images the entire person can be seen or even a large crowd of people.

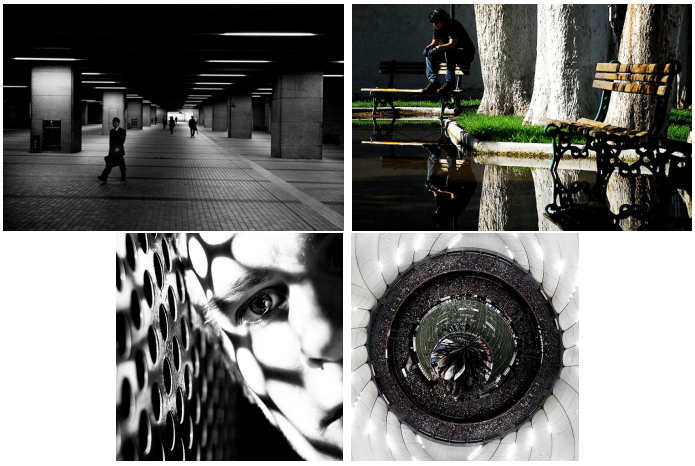


Figure 6.25: Some example images for the 'People' concept.

Figure 6.26 shows the classification results for the 'People' concept and figure 6.27 shows some detection examples of the MOD based concept detection.

Compared to the results for 'Animals', the 'People' concept detection results are more promising for the MOD method. Both the SVM and the NN classifiers slightly outperform SIFT-SVM.

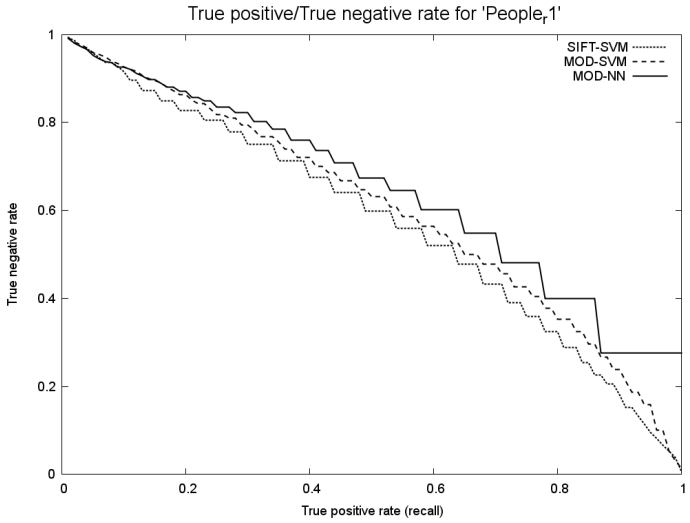


Figure 6.26: Classification results for the 'People' concept.

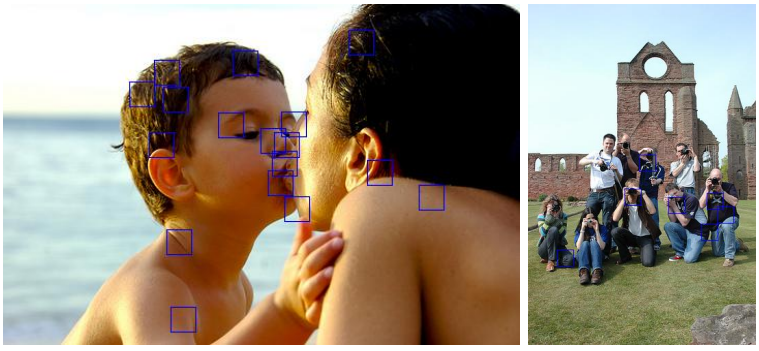


Figure 6.27: Some MOD based concept detection results for the 'People' concept.

6.6.5 Concept 'Plant life'

Figure 6.28 shows some example images for the 'Plant life' concept. The 'plant life' concept is intuitively linked to green plants, but the images again show the wide variety of images that are in the dataset. The 'Plant life' concept contains plants, trees, grass, flowers.



Figure 6.28: Some example images for the 'Plant life' concept.

Figure 6.29 shows the classification results for the 'Plant life' concept and figure 6.30 shows some detection examples of the MOD based concept detection.

The graph again shows an improved detection rate for the MOD based methods, however in this case the MOD-SVM outperforms the MOD-NN method, which we have not seen for the previous two concepts.

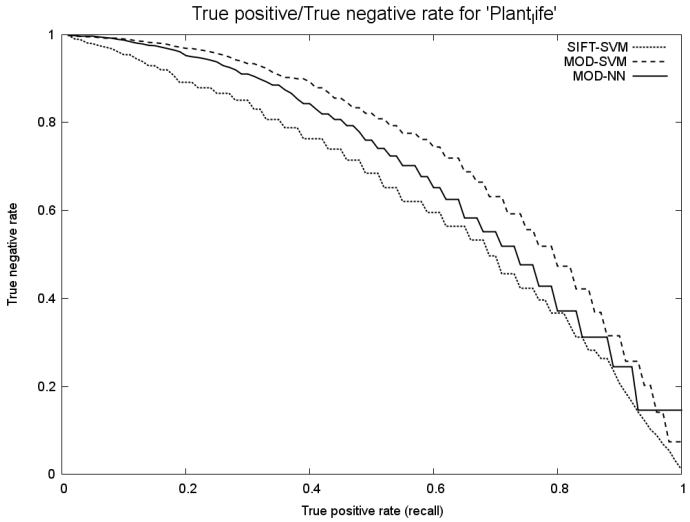


Figure 6.29: Classification results for the 'Plant life' concept.



Figure 6.30: Some MOD based concept detection results for the 'Plant life' concept.

6.6.6 Concept 'Sky'

Figure 6.31 shows some example images for the 'Sky' concept. The 'Sky' concept is usually regarded as an easy concept to detect. Blue sky has a very specific color and almost no texture. However, in this dataset, the sky concept can also be considered a reasonably difficult one. Cloudy skies and night skies, or just a vague notion of sky somewhere in the image are the main reasons for this.



Figure 6.31: Some example images for the 'Sky' concept.

Figure 6.32 shows the classification results for the 'Sky' concept and figure 6.33 shows some detection examples of the MOD based concept detection.

The graph above shows the classification results for the 'sky' concept. Again, the MOD based methods outperform the SIFT method.

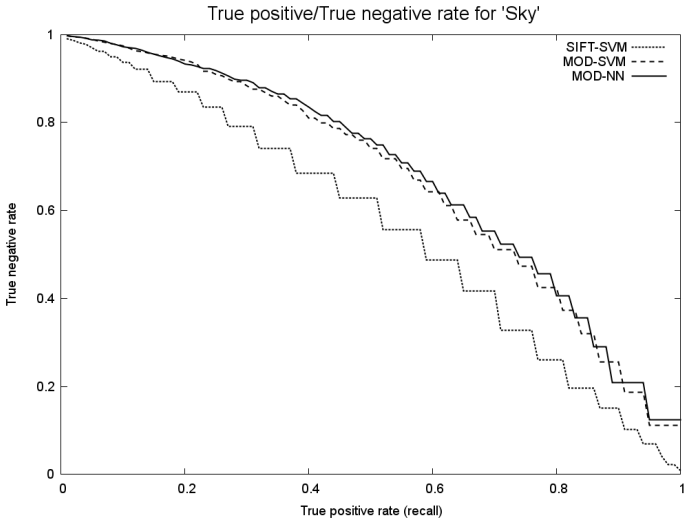


Figure 6.32: Classification results for the 'Sky' concept.

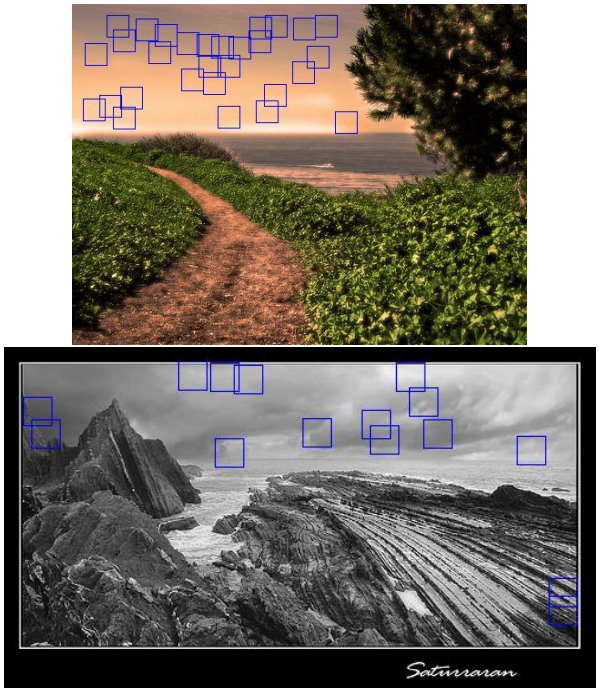


Figure 6.33: Some MOD based concept detection results for the 'Sky' concept.

6.6.7 Concept 'Structures'

Figure 6.34 shows some example images for the 'Structures' concept. The 'Structures' concept contains images with man-made structures on it. One can think of buildings, roads, bridges or fences.



Figure 6.34: Some example images for the 'Structures' concept.

Figure 6.35 shows the classification results for the 'Structures' concept and figure 6.36 shows some detection examples of the MOD based concept detection.

In this case, the increase in performance is less obvious, although the MOD methods perform slightly better.

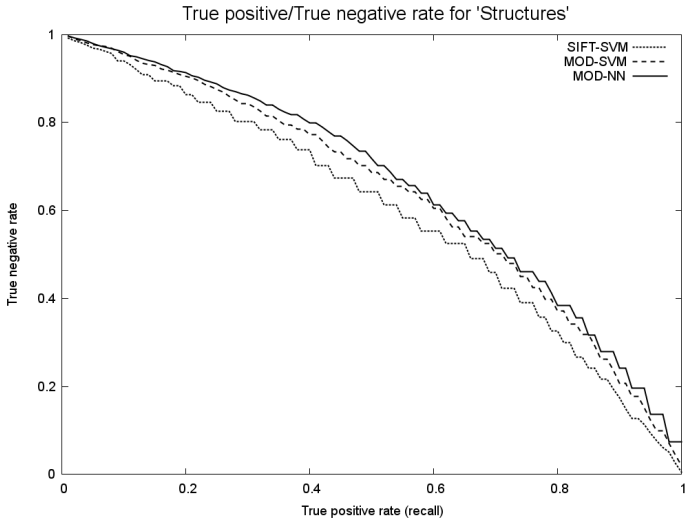


Figure 6.35: Classification results for the 'Structures' concept.



Figure 6.36: Some MOD based concept detection results for the 'Structures' concept.

6.6.8 Concept 'Sunset'

Figure 6.37 shows some example images for the 'Sunset' concept. The 'Sunset' concept is a typical color-based concept. Usually the colors red, orange or dark blue can be found and one would expect higher classification accuracy because of the more obvious relation between image features and the concept.



Figure 6.37: Some example images for the 'Sunset' concept.

Figure 6.38 shows the classification results for the 'Sunset' concept and figure 6.39 shows some detection examples of the MOD based concept detection.

The test set for the sunset concept did not contain 50% positively labeled images, as there are not enough positive images in the MIRFLICKR dataset. So, if the accuracy would be plotted in a graph (not visible here), it would flatten around 0.58, which is the percentage of negatively labeled images. For very high thresholds, all images will be classified as not containing the concept, so for 58% of the images this will be true, instead of the expected 50% as with the other tested concepts.

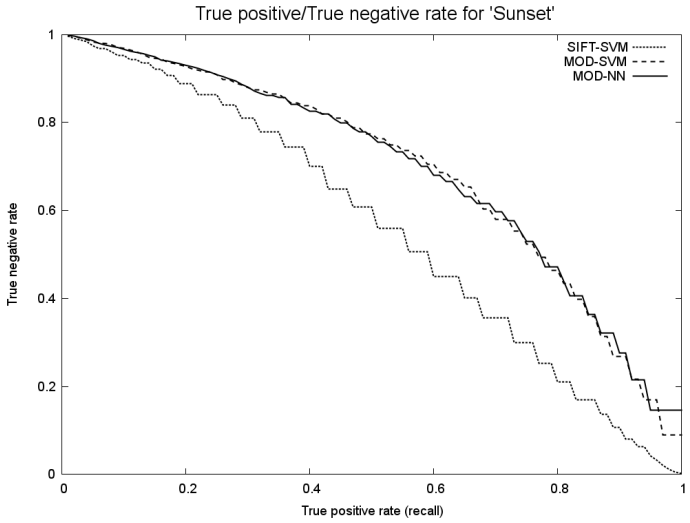


Figure 6.38: Classification results for the 'Sunset' concept.



Figure 6.39: Some MOD based concept detection results for the 'Sunset' concept.

6.6.9 Concept 'Transport'

Figure 6.40 shows some example images for the 'Transport' concept. The 'Transport' concept covers all kinds of transport, such as cars, boats, bicycles. Like with other concepts, often only partial views of a concept are visible in the images. A few common shapes exist among the cars and bicycles, for example the tires. These are round and black. For boats, water is usually present around them.



Figure 6.40: Some example images for the 'Transport' concept.

Figure 6.41 shows the classification results for the 'Transport' concept and figure 6.42 shows some detection examples of the MOD based concept detection.

Just like the concept indoor, the transport concept is a difficult concept to grasp with visual descriptors. Many different objects and situations relate to the word transport and as such, a high retrieval performance was not expected from our classifiers with the limited training set. Still, the MOD based methods perform better than the SIFT based method.

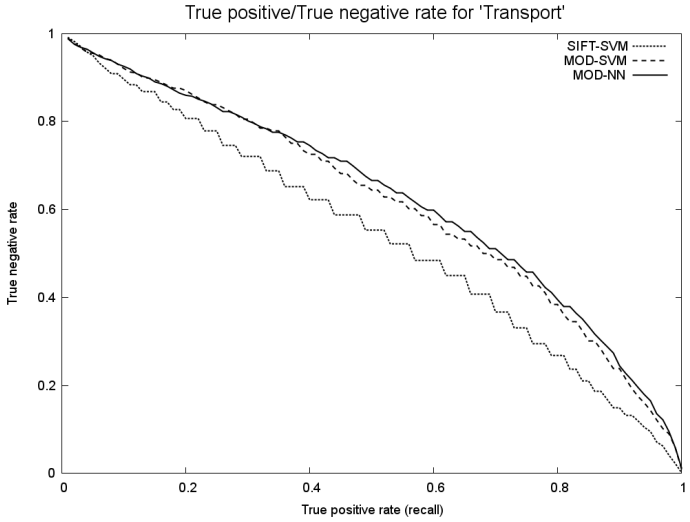


Figure 6.41: Classification results for the 'Transport' concept.

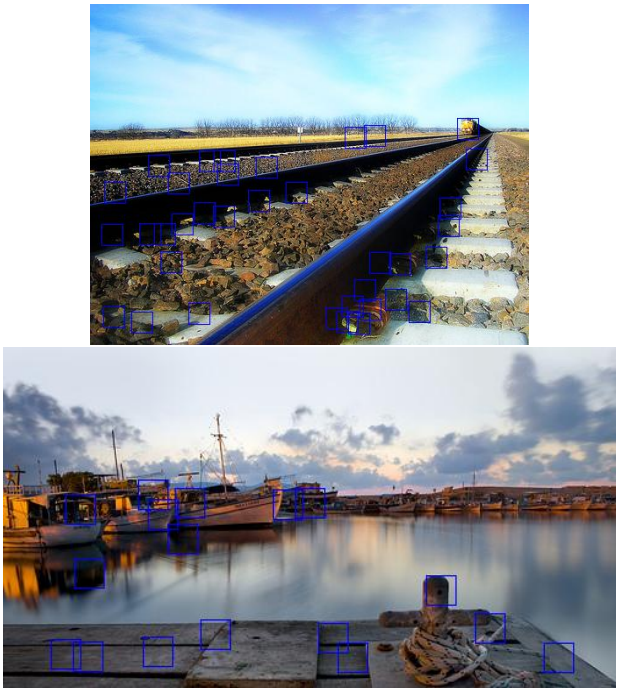


Figure 6.42: Some MOD based concept detection results for the 'Transport' concept.

6.6.10 Concept 'Water'

Figure 6.43 shows some example images for the 'Water' concept. The concept 'Water' refers to more than what one would first expect: not just water like a river or an ocean, but also rain or an aquarium are covered by this concept. The standard 'blue and ripples' description is clearly not sufficient.



Figure 6.43: Some example images for the 'Water' concept.

Figure 6.44 shows the classification results for the 'Water' concept and figure 6.45 shows some detection examples of the MOD based concept detection.

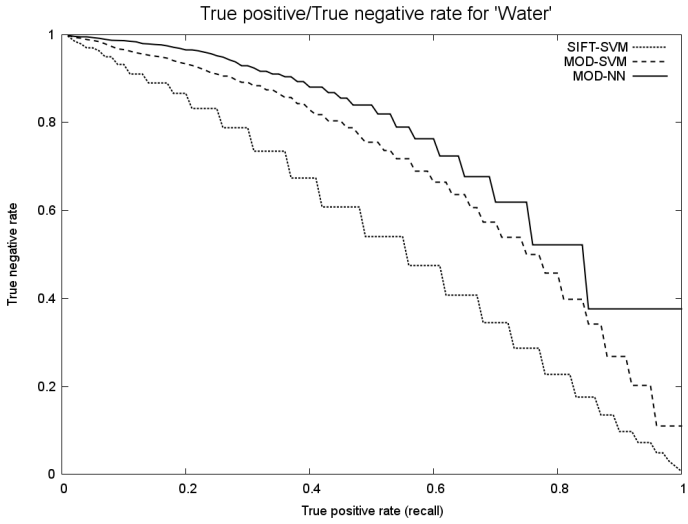


Figure 6.44: Classification results for the 'Water' concept.



Figure 6.45: Some MOD based concept detection results for the 'Water' concept.

6.6.11 Overall results

Figure 6.46 shows the overall Recall versus True Negative rate. The MOD based methods outperform the SIFT based method, but the distinction between SVM and a neural network for the MOD based methods is not obvious. The neural network slightly outperforms the support vector machine approach, especially for high recall values.

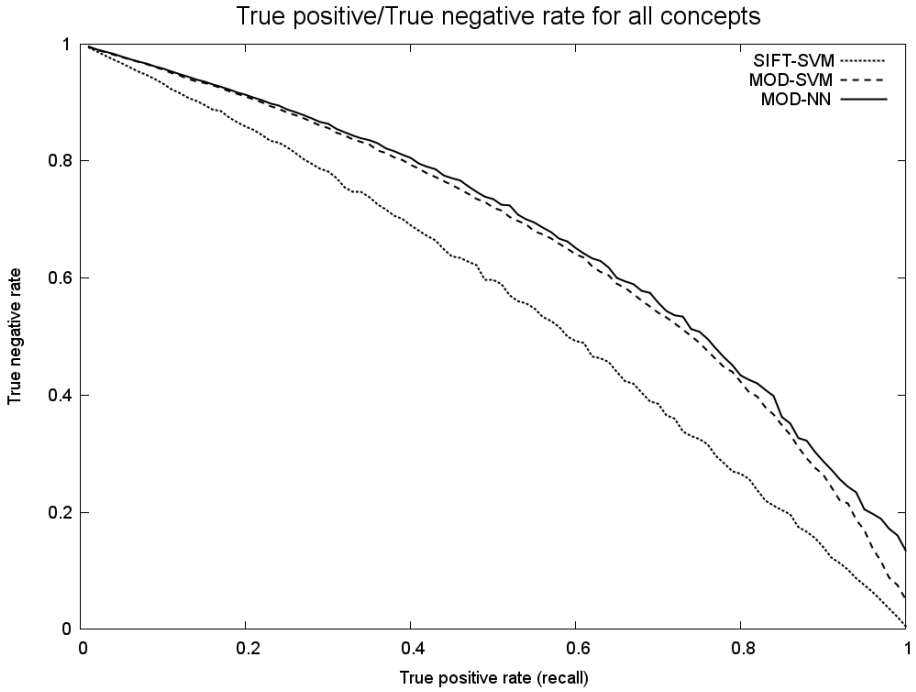


Figure 6.46: Average performance over all concepts.

6.7 Discussion, conclusions and future work

The results from the different concepts and classifiers show the promising results that can be obtained when using the MOD interest points as a hint on where to look for concepts within images. Especially the combination of a more advanced classifier like the support vector machine or a neural network with the MOD regions turns out to give very interesting results.

As an example, detecting the 'tree' concept, as shown in Figure 6.9, with a support vector machine yields a correct classification of a small image region, that we did

not expect to be easily detected.

The other results on the Leiden-Amsterdam database show that the 'building' concept detection results in some errors. The nature of this concept is visually more complex than a 'sky' concept, so we expected more difficulties with detecting the concept, which can be clearly seen in Figure 6.10.

For the MIRFLICKR experiments, our research has focused on a new method of interest point detection for sub-image visual concept detection. Our experiments have shown that the MOD based classifiers with a few standard features outperform the SIFT based classifier with SIFT feature descriptors, a method that has been recently shown to be very effective for visual concept detection.

The results of the experiments also indicate that for the far majority of the concepts, the neural network based classifiers have equal or greater performance than the SVM based classifiers. On average over all the concepts, the neural network has a 7.4% improvement, although the main contribution comes from the highest recall values, where the neural network clearly outperforms SVM.

In our results we compare relative detection rates for the same training and test sets between MOD and SIFT salient points. We expect that the absolute detection rates can be significantly improved by using more training images, which we intend on pursuing in the future.

Also for future research, we would like to improve our MOD based concept detector by using other features. In the current tests, only three basic features were used and we expect that for example a scale-invariant feature would benefit the detection of concepts that vary in size.

Chapter 7

Multi-Dimensional Maximum Likelihood

7.1 Introduction

In retrieval applications, there is a point where the similarity between two documents has to be determined. Documents similar to the query should be ranked higher in the search results than documents that are less similar.

An interesting question to ask is: how do we determine similarity? When are two image features similar? Assuming that these features are represented by vectors of real numbers, standard ways of determining feature similarity exist, such as the sum of absolute distances (SAD). However, these methods assume certain properties of the data that is used. For example, as will be described later in this chapter, using a sum of squared distances (SSD) as a similarity metric, assumes that the feature differences of similar images are normally distributed.

Previous research has shown that this is not always true and that more optimal similarity measures exist if the distribution of feature data of similar images is known. This research extends this work to a multi-dimensional situation where not only the distribution of feature value differences is analyzed, but also the relation of these differences to the feature values themselves. In other words, given a set of similar images, we are not examining the distribution of $(x - y)$ for each pair of features but the distribution of (x, y), which is a 2D distribution. This approach results in the multi-dimensional maximum likelihood (MDML) similarity measure.

7.2 Definitions

A distance is a function D with nonnegative real values, defined on the Cartesian product $X \times X$ of a set X. So, for every $x, y \in X$:

1. $D(x, y) = 0$

A distance is called a metric if the following properties also hold for every $x, y, z \in X$:

2. $D(x, y) = 0$ if and only if $x = y$
3. $D(x, y) = D(y, x)$
4. $D(x, z) = D(x, y) + D(y, z)$

A set X provided with a metric is called a metric space. As an example, every set X has the trivial discrete metric $D(x, y) = 0$ if $x = y$ and $D(x, y) = 1$ otherwise.

If one of the metric conditions is not met, the distance function is referred to as a similarity measure.

7.3 Detailed description

We begin by reviewing the theory of maximum likelihood theory. Given two images X and Y with feature vectors x and y, the probability of these two being similar, is the product of the probabilities of the similarity of each element of the vector.

$$Sim(x, y) = \prod_{i=1}^{n} P_{sim}(x_i, y_i) \tag{7.1}$$

These individual probabilities $P_{sim}(x_i, y_i)$ are directly linked to the distribution of the feature value co-occurrences and are often modeled by a chosen probability density function.

The origin of the widely used L_2 distance metric is the assumption that the differences between feature vector elements are normally distributed [78], with the same parameters for each element. This results in the following similarity metric:

$$Sim(x, y) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(x_i - y_i)^2}{2\sigma^2}\right)} \tag{7.2}$$

If one wants to find the most similar image to an image X, we loop through all images Y to find the image with the highest similarity value resulting from 7.2.

However, since σ and μ are both constants in this formula, we can simplify the maximization problem using:

$$Sim(x,y) = \prod_{i=1}^{n} e^{-(x_i - y_i)^2} \tag{7.3}$$

Applying $ln(ab) = ln(a) + ln(b)$ to this function yields:

$$Sim(x,y) = \sum_{i=1}^{n} ln\left(e^{-(x_i - y_i)^2}\right) \tag{7.4}$$

Which in turn can be simplified to:

$$Sim(x,y) = \sum_{i=1}^{n} -(x_i - y_i)^2 \tag{7.5}$$

Finally, if we convert the maximization problem to a minimization problem, we can use:

$$Sim(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{7.6}$$

This is exactly the sum of squared distances. So now we can conclude that if the differences of all feature values have the same normal distribution, using L_2 as a distance metric is justified.

As mentioned before, the normal distribution does not always represent the true distribution in retrieval experiments. For example in motion detection in video, a representative distribution and the best fit Gaussian is shown in figure 7.1.

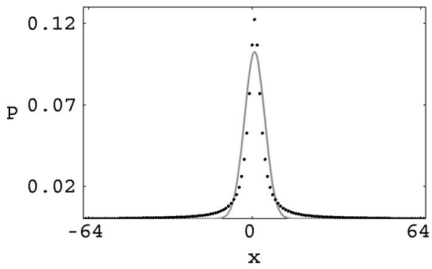


Figure 7.1: An example of the best fit Gaussian to the true distribution in one of our video retrieval experiments.

Especially the more prolonged tails of the true distribution are noticeable. This suggests that other distributions might have a better fit to the true underlying distribution and that other similarity metrics should be used in this case.

There are three assumptions that are made when the L_2 distance metric is used for determining the similarity of two feature vectors:

1. The difference $(x - y)$ between elements of the feature vectors contains all information needed to determine similarity.
2. The differences between the elements of similar feature vectors follow a Gaussian distribution.
3. All differences of the elements in the feature vector have identical independent distributions.

The first assumption is widely used in all types of distance metrics. It is assumed that the differences of feature vectors follow a certain distribution for similar images and that this difference captures it all. However, there are many other possibilities that can be analyzed. Instead of using the differences, looking at the absolute values (x, y) or even $(x + y)$ or $(x * y)$ might result in better distance metrics.

The second assumption states that $(x-y)$ follows a Gaussian distribution, which is explained by the derivation of the L_2 metric in this chapter. The third assumption is based on the fact that the same distance metric is used for each feature value when determining the similarity of feature vectors. The same function is used for each feature vector element.

If the third assumption is not true, then the distribution of each feature vector element of similar images should be determined and a suitable distance should be selected based on that distribution. This will not be addressed in this chapter, but it will be the focus of future research.

If the second assumption is not true (which it is not, as was demonstrated by previous research [76]), a better approximation for the true distribution should be used to select a distance metric that better suits the data. This will be the focus of the rest of this chapter.

If the first assumption is not true, more information should be extracted from the original feature values. In this chapter, we do this by examining the distribution of (x, y).

7.4 Related work

A lot of research has been done in the field of distances and metrics. Below is a review of a few recent papers. Almost all try to improve on the standard L_1 and L_2 norms by looking at their general form, the Minkowski L_p norm:

$$D_{L_p}(x, y) = \left(\sum (x - y)^p\right)^{\frac{1}{p}} \tag{7.7}$$

With SAD (L_1) and SSD (L_2) being two instances of this norm.

Sebe et al. [78] analyze the true noise distributions of similar items from various test sets and the authors conclude that the implied Gaussian or Exponential distributions are often not close enough to the true noise distributions. The presented Cauchy metric has increased performance over both L1 and L2. Further research, also by Sebe et al. [76], shows that using the histogram of feature value differences of similar items, further increases the similarity measure performance. This research addresses assumption 2, the differences of feature values of similar images do not follow a Gaussian distribution.

Yu et al. [97] build on the research by Sebe and introduce several other distance measures based on different probability distributions that are implied by the harmonic mean and the geometric distribution, as opposed to the arithmetic mean and the median that are related to L_2 and L_1. Yu also correctly identifies the problem of the possibility of having different noise distributions for individual feature elements. He shows that using several distance measures for sets of feature elements, by using a boosting algorithm to find these sets, the performance of the distance measure increases. This research addresses assumptions 2 and 3, the differences of feature values of similar images do not follow a Gaussian distribution and also the distributions of differences of feature elements are not the same for each element.

However, the second assumption, stating that $(x - y)$ contains all information needed to determine similarity, is not addressed by these papers. Our research will focus on this assumption to determine if more information should be used.

7.5 Multi-Dimensional Maximum Likelihood similarity (MDML)

As mentioned above, previous research [76] has shown that analyzing the distribution $P(x_i - y_i)$ of feature differences for adjusting the similarity metric results in better retrieval results. The resulting similarity metric was called the maximum likelihood metric. In this research we are directing our focus at the distribution of $P(x_i, y_i)$ and we attempt to create a 2D version of this metric. Using (x, y) seems like the most general approach to analyze the data.

Recall that a similarity measure can be thought of as a histogram of feature value co-occurrences. If given enough training samples, the normalized histogram will give a representation of the true joint probability of the occurrence of each feature value pair for a certain class of images. The similarity of two images, given the class C of one of them, can then be calculated by

$$Sim(x, y) = \prod_{i=1}^{n} H_C(x_i, y_i) \tag{7.8}$$

Where H_C is the probability of the two feature values occurring for similar images, determined by the histogram for class C. To convert this into a minimization problem and to get more numerical stability in the calculations (by converting the product to a sum), we get:

$$Sim(x, y) = -\sum_{i=1}^{n} log\left(H_C(x_i, y_i)\right) \tag{7.9}$$

Determining the similarity of two feature vectors is now reduced to directly using the true 2D distribution of similar feature values.

7.6 Experiments on stereo matching

Stereo matching is the process of finding corresponding image locations in sets of images, such that these locations represent the same real-world location. The difference between the pixel locations in the images is called the disparity.

Stereo matching algorithms come in various forms, but three important categories can be distinguished: local, semi-global and global methods. Local methods try to find the best match to a single location by using information around that location or possibly even only the pixel itself. Global methods try to optimize a global correspondence function, which causes local correspondences to be influenced by neighboring locations. Semi-global methods use the same principle as global methods, a form of functional optimization, however restricted to a subset of the global set of correspondences to improve calculation times.

A well-known dataset for testing stereo matching algorithms is the Middlebury dataset. A total of 28 sets of stereo images and corresponding ground truth were released between 2001 and 2006. In our experiments we have used the datasets released in 2001, 2003 and a few from 2005, together with two other sets (Map and Tsukuba) that were used in the experiments from [70].

7.6.1 Results - template based

For our first experiments, we have selected a template size of 11 and a search range of one line, because the images are already rectified. Table 7.1 shows the accuracy of template based stereo matching for each Middlebury dataset and various similarity measures. The experiments were carried out with 10-fold cross validation.

7.6.2 Results - pyramidal template based

These experiments were based on a variation of the template based stereo matching algorithm: it uses a 5-layer pyramid of subsampled versions of the original

Dataset	L1	L2	ML	MDML
Barn 1	93.88 (0.07)	92.93 (0.05)	94.75 (0.08)	96.66 (0.05)
Barn 2	91.43 (0.09)	91.70 (0.12)	94.46 (0.10)	95.63 (0.09)
Bull	95.27 (0.07)	95.54 (0.08)	97.26 (0.08)	97.66 (0.04)
Poster	84.58 (0.13)	85.29 (0.09)	87.08 (0.11)	94.18 (0.22)
Sawtooth	93.92 (0.09)	92.97 (0.11)	94.90 (0.15)	96.59 (0.13)
Venus	92.46 (0.13)	92.45 (0.14)	93.75 (0.14)	95.53 (0.10)
Map	92.30 (0.09)	91.33 (0.06)	92.57 (0.05)	93.05 (0.09)
Tsukuba	51.27 (1.16)	50.11 (1.18)	51.84 (0.97)	71.05 (0.87)
Cones	81.23 (0.26)	81.77 (0.24)	88.72 (0.21)	92.14 (0.16)
Teddy	77.97 (0.26)	77.70 (0.21)	84.40 (0.22)	88.68 (0.24)
Art	56.01 (0.13)	54.87 (0.16)	68.10 (0.19)	73.01 (0.20)
Books	72.83 (0.36)	75.43 (0.20)	78.01 (0.26)	86.65 (0.26)
Dolls	78.63 (0.12)	76.51 (0.11)	81.69 (0.11)	85.90 (0.16)

Table 7.1: Average accuracy and unbiased standard deviation of found correspondences for the Middlebury datasets and various similarity measures

image. Searching for a match starts at the lowest resolution layer. Each layer has a separately trained ML or MDML classifier. The matching template location on the low resolution image is used as the center of the limited search range for the next higher resolution.

For these experiments, we have selected a template size of 11 and a search range of one line, because the images are already rectified. Table 7.2 shows the accuracy of template based stereo matching for each Middlebury dataset and various similarity measures. The experiments were carried out with 10-fold cross validation.

Dataset	L1	L2	ML	MDML
Barn 1	93.88 (0.07)	92.93 (0.05)	94.75 (0.08)	96.66 (0.05)
Barn 2	90.90 (0.29)	91.20 (0.33)	91.50 (0.22)	95.58 (0.11)
Bull	94.94 (0.06)	95.15 (0.08)	95.13 (0.07)	97.79 (0.04)
Poster	85.48 (0.14)	86.03 (0.09)	85.84 (0.18)	94.41 (0.20)
Sawtooth	93.03 (0.10)	91.98 (0.11)	93.44 (0.13)	96.65 (0.13)
Venus	92.51 (0.10)	92.10 (0.09)	86.61 (0.09)	96.12 (0.01)
Map	90.39 (0.10)	88.69 (0.10)	91.45 (0.08)	94.53 (0.16)
Tsukuba	46.22 (1.01)	41.23 (0.75)	51.64 (0.65)	74.61 (0.09)
Cones	75.91 (0.27)	75.63 (0.22)	80.29 (0.28)	92.14 (0.17)
Teddy	76.24 (0.26)	75.57 (0.16)	83.82 (0.28)	89.64 (0.24)
Art	53.73 (0.23)	51.44 (0.20)	64.53 (0.28)	75.48 (0.10)
Books	68.27 (0.36)	68.49 (0.21)	75.72 (0.15)	87.99 (0.29)
Dolls	74.45 (0.17)	72.00 (0.18)	75.28 (0.14)	87.03 (0.12)

Table 7.2: Average accuracy and unbiased standard deviation of found correspondences for the Middlebury datasets and various similarity measures

7.7 Future work

Given the MDML histogram and assuming enough training samples were available to find the true probability distribution, an optional next step is to find a parametric representation of the distribution. This will reduce the need to keep the histogram data for calculating distances. Several methods can be thought of when converting a histogram to a 2D function:

- PCA
- Surface approximation
- Wavelets

As the research by Sebe has shown, using an approximation to the true distribution will result in suboptimal accuracy, but usually this will be compensated for by processing speed or memory consumption.

In case of the stereo matching algorithm described in this chapter, the 2D histogram uses 64 kilobytes of memory and uses a lookup in the probability distribution array for each pixel in the template matching step. Future research will be focusing on the accuracy of stereo matching with parametric 2D approaches based on the true distribution, taking into account the effect on processing time and memory.

Chapter 8

Texture Classification: What Can Be Done with 1 or 2 Features?

For real-time imaging, pattern classification with short feature-vectors is desirable. One well-known and effective method used for texture classification is the use of statistical information on 3x3 pixel blocks. Our method is an extension of this method to larger and non-square features, which we call 'constructed features'. The standard 3x3 features are very usable for real-time imaging, because the resulting feature space can only contain 512 elements and the vectors can be created very quickly. However, larger or non-square features might give better classification results. Our proposed method searches for small sets of constructed features, with arbitrary size and shape, which will give the best results for a classifying a specific texture, and still keeping the feature vectors as short as possible. In this chapter, we show the selected features and the performance of our method with a minimum distance classifier and with a neural network on a texture classification task.

8.1 Introduction

In many applications such as video analysis, computational efficiency is of critical importance. Thus, in certain contexts, the maximum allowed computational complexity is known beforehand and the algorithm must therefore be designed to accommodate the computational requirements.

Furthermore, the novel aspect of this chapter is in exploring binary pattern oriented features which are not simply 3x3 templates. We attempt to find more

general template shapes which could give better performance and accuracy than the traditional 3x3 features.

Our texture classification method generates features for a specific texture and iteratively selects the best features to use. In this way, a feature set of any size can be created, that will give the best classification results for that specific texture.

8.2 Related work

In 1990, Wang and He have introduced the Texture Unit (TU) [93], which is generally considered as the basis for texture analysis based on the distribution of 3x3 pixel blocks.

A variant of the TU was introduced by Ojala et al. [60], which they named the local binary pattern (LBP). They showed that classifying textures using LBP features turned out to be very effective, especially when combined with a local contrast measure. Later they have improved the LBP feature by creating a rotation invariant version of it [62].

Other texture unit variants include multiscale selected local binary features by Raja [64] or the simplified texture unit by Madrid-Cuevas [45]. Variants of local image features have also been applied to other fields, for example face detection [33] or facial expression recognition [26].

In 1996, Huijsmans et al. [30] showed that using frequencies of 3x3 binary patterns is very helpful in copy-locating in image databases. In their experiments, they showed that using a subset of all features or non-equal weighing yielded better results than using the entire set of features with equal weights. Mäenpää later found the same results for the LBP features [46].

In line with these findings, we expect that non-square features or larger features might also improve better classification results over the use of the entire range of 3x3 features.

8.3 Our method

We define a "Constructed" feature as a binary pattern of NxN in size. An example is shown in Figure 8.1. All these constructed features are tested for classification accuracy.

In general, the method applies to arbitrarily large features. For computational efficiency reasons, we note that the space of all potential features is exponential, which is not feasible for straightforward searching. To make the problem tractable, we record all possible features within the training set up to a certain size and then we select features which appear in all images of the training set. These features form the space for candidate features.

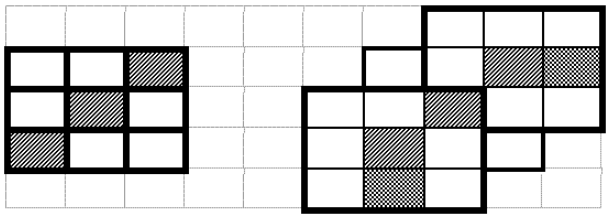


Figure 8.1: Constructing a feature. Starting with a 3x3 feature in a), we can then create a new feature in by joining several 3x3 features together to 'construct' a 5x5 feature as in b).

For our first experiments, we used an even more restricted set of features: each of the features in the candidate set for a specific class was required to be present in every image from that class. We wanted to make sure we were using the most descriptive features, which are probably those features that can be found in every positive example.

To get binary images for our feature construction, we create edge-detected versions of our textures with the Marr-Hildreth algorithm. This method detects intensity changes in an image by convolving it with the $\nabla^2 G$ filter, where ∇^2 is the Laplacian operator

$$\left(\frac{\partial^2 x}{\partial x^2} + \frac{\partial^2 y}{\partial y^2}\right) \tag{8.1}$$

and G stands for the two dimensional Gaussian distribution

$$G(x, y) = e^{-\frac{x^2+y^2}{2\pi\sigma^2}} \tag{8.2}$$

In effect, this is a 'mexican-hat' operator. The zero-crossings in the output of this convolution mark the intensity changes. Pixels with an intensity change, or zero-crossing, are set to 1 in the resulting image and the rest of the pixels are set to 0. Figure 8.2 shows an example of this edge detection method.

The sigma parameter that is used for the Marr-Hildreth algorithm selects the scale the method works on, or the size of the mexican hat. Higher values yield the detection of less detailed structures in the images. In our experiments we used a fixed value of 2.0 for sigma, but by varying this parameter, the search space for the best features could even be extended.

For each feature, we keep track of the number of times it is found within each image. Note that the number of features that can be found for each class is significantly less than the maximum number of possible features. For computational purposes, the features are stored in hash tables for fast lookups.

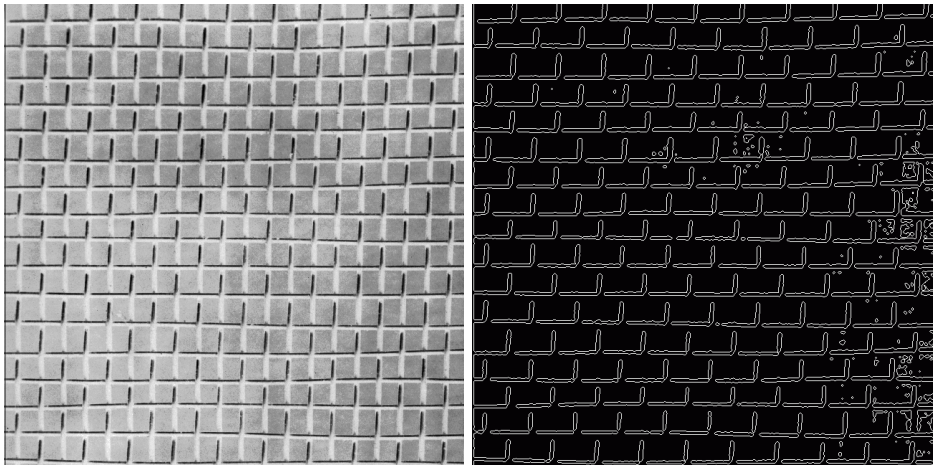


Figure 8.2: Brodatz texture D1 and the result of the Marr-Hildreth edge detection applied to it with sigma = 2.0

Since we are trying to find an optimal feature set, in theory every possible combination of features should be tested for classification accuracy, but the required processing time will increase exponentially. We therefore use a greedy hill-climbing method by iteratively increasing the feature set by adding a feature to the set that minimizes the training error.

The first feature can be selected directly by using the feature that minimizes the training error. After that, in case that x features are already found for a specific class, we test the classification accuracy of all sets of features that consist of these x features plus one of each of the features that are left. By repeatedly following this procedure, a best feature set of any length can be found.

8.4 Results

For testing our method, we used 12 textures from the Brodatz database, which are listed in Figure 8.3. We created a separate classifier for each texture class.

For each of these classes, we have generated random subblocks of size 64x64 and we selected 50 training blocks from the upper half of the image and 50 test blocks from the lower half. For this experiment, we have limited the size of the constructed features to 7x7.

Table 8.1 contains the results from our first experiments. We have determined the best two features for each class, using 3x3 features only and with the constructed features. Also, we show the comparison between minimum distance classifiers and neural network classifiers. The neural network is a fully connected 3-layer

network that has a hidden layer with 10 nodes and one output node. These values were determined by empirical testing. The number of input nodes is equal to the number of features.

Note that for each class, there are 50 positive examples of that class and 550 negative examples in both the training and the test set. The misdetection rates are given over all test images.

Class	Nearest neighbor (1st and 2nd feature)				Neural network (1st and 2nd feature)			
	3x3		constructed		3x3		constructed	
D1								
	0.043	0.002	0.008	0.002	0.035	0.005	0.083	0.005
D3								
	0.095	0.103	0.070	0.077	0.087	0.102	0.065	0.068
D4								
	0.072	0.047	0.072	0.047	0.052	0.040	0.063	0.042
D6								
	0.018	0.023	0.005	0.027	0.027	0.028	0.003	0.032
D9								
	0.180	0.107	0.180	0.107	0.047	0.060	0.047	0.060
D11								
	0.097	0.108	0.080	0.073	0.100	0.098	0.078	0.055
D16								
	0.057	0.102	0.057	0.047	0.050	0.053	0.042	0.042
D17								
	0.060	0.058	0.055	0.058	0.052	0.085	0.047	0.053

Class	Nearest neighbor (1st and 2nd feature)				Neural network (1st and 2nd feature)			
	3x3		constructed		3x3		constructed	
D20								
	0.060	0.008	0.000	0.017	0.060	0.007	0.083	0.007
D21								
	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.020
D24								
	0.088	0.097	0.088	0.097	0.080	0.012	0.083	0.012
D29								
	0.083	0.115	0.097	0.088	0.102	0.090	0.102	0.065

Table 8.1: Misdetection rates and the selected features for 3x3 features only and for constructed features, using a minimum distance classifier or a neural network.

8.5 Discussion, conclusions and future work

The first thing that is noticeable in the results, is that adding a second feature does not always give a better test result. This is probably caused by the small number of images in the training and test sets. In general, the training error was lower with two features.

Also, it is interesting to see that for some textures, the optimal features are still the 3x3 features, even when all constructed features are considered. The reason for this could be the fine scale of the texture. The small details might have are already been captured by the 3x3 features.

For the minimum distance classifiers with one feature, 8 out of 12 selected features were constructed features. For the neural network classifiers with one feature, 9 out of 12 selected features were constructed features. These features were selected based on a lower misdetection rate on the training set.

Table 8.2 shows a comparison between constructed features and 3x3 features. It can be seen that using the constructed features gives an improved classification result on average.

	MDC-1	MDC-2	NN-1	NN-2
Underperforming	1	2	4	3
Equal	5	6	2	4
Outperforming	6	4	6	5

Table 8.2: The performance of the constructed features compared to the 3x3 features.

For us, this justifies the idea of further experimenting with constructed features instead of 3x3 features for texture classification.

Our next experiments will focus on the constructed features with what we call ̈don't care ̈pixels. After constructing a feature, we will test the same feature with one or more pixels left out. We expect these features to outperform the features shown in this chapter.

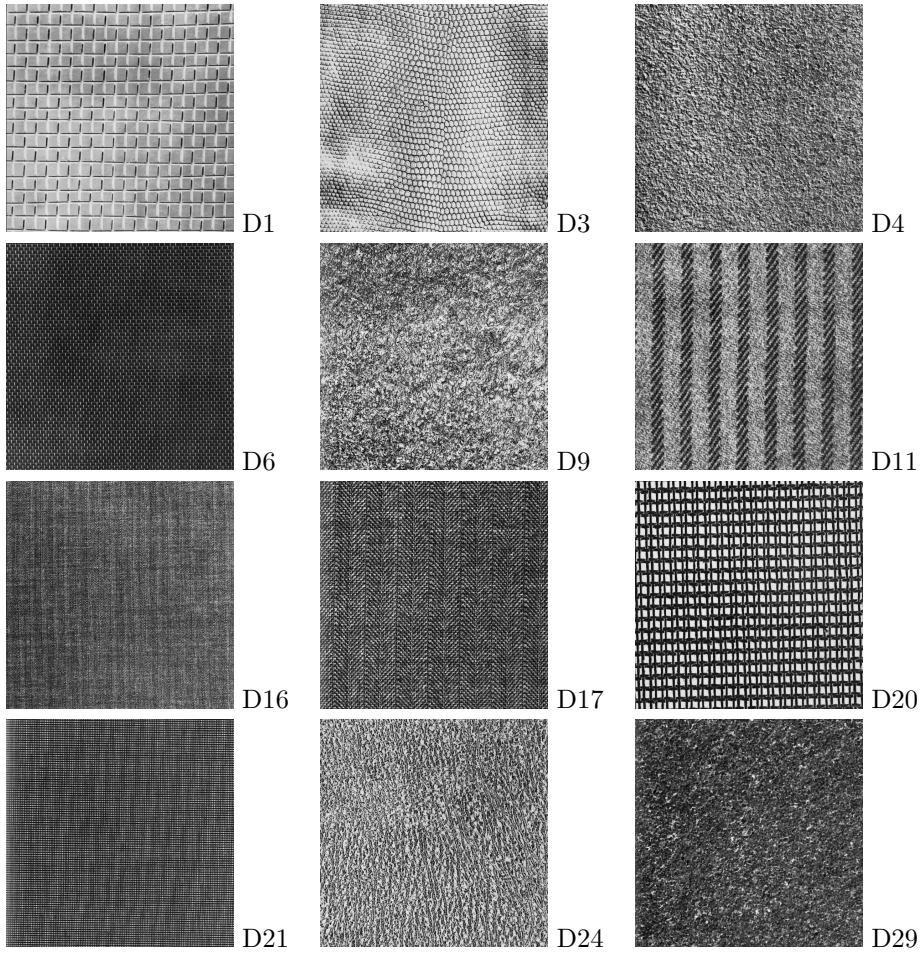


Figure 8.3: Brodatz textures used in our experiments.

Chapter 9

Detecting and Identifying Moving Objects in Real-Time

For surveillance applications or for human-computer interaction, the automated real-time tracking of moving objects in images from a stationary camera can be very useful. In this chapter we describe a novel method for object tracking which works in real-time using standard hardware. The object tracking algorithm uses an improved adaptive version of an existing robust motion detection method and a set of decision-rules is used to handle difficult situations such as tracking multiple objects at the same time, crossing paths and partial occlusion. Experiments on several real complex test sets were performed and discussed.

9.1 Introduction

Tracking moving objects in images from a stationary camera, such as people walking or cars driving, is a very useful technique for surveillance applications. The same technique can also be used in different environments, such as human-computer interaction or the automated counting of objects. A well-known example of a real-time object tracking application used for human-computer interaction is the Sony eye-toy.

We have implemented a new method of object tracking by using decision rules for handling the fairly general problem of unlimited moving objects, occlusions by the environment and other objects, and indoor and outdoor lighting.

9.2 Related work

A lot of work has been done in the field of object tracking. [8] [10] [19] [20] [42] [49] [94] One of the main conferences for this research subject is Performance and Evaluation of Tracking and Surveillance (PETS). The articles clearly show that several common problems such as occlusions, multiple objects and variable illumination are largely unsolved. Many of the systems achieve good accuracy by making special assumptions such as a single person, two persons, only indoors lighting, etc. In fact, the conclusions of some of the advanced systems are simply that they are unsuited for outdoors [82]. In short, the general problem that we have looked at is very difficult.

The first step for every object tracking algorithm is motion detection. This can be done using a standard background-subtraction method based on color or intensity, but other more sophisticated methods like Gaussians [14] or optical flow [50] [47] are widely used. For an overview of background subtraction algorithms, see [5].

For the object tracking itself, a variety of methods is available. The choice of the right method depends on the application. Object tracking can be done with just simple nearest-neighbor matching of regions with motion [53]. An improvement of this method would be to use motion prediction, for example with Kalman filters [17] or Bayesian networks [20].

Other methods for object tracking include the kernel-based object tracker made by Comaniciu et al. [9]. Another interesting real-time object tracking system to look at is W4, by Haritaoglu et al. [21] [22].

9.3 Motion detection

Horprasert, Harwood and Davis have developed a method [27] [28] for robust motion detection using background subtraction with a statistical model of the background. We have chosen to adapt this method for our object tracking algorithm, because of its robustness to changes in lighting and the ability to adapt to deposited objects. The main idea of this method is to decompose the difference between the current color information and the modeled background for every pixel into a chromaticity (color) component and a brightness component.

After the background model is created (see below for details), new frames can be compared to this model to detect motion by comparing the expected RGB value for pixel i, $E_i = [E_R(i), E_G(i), E_B(i)]$, to the RGB value of that pixel in the current frame, $I_i = [I_R(i), I_G(i), I_B(i)]$ in the following way:

When seen in a three-dimensional RGB cube, the line between the origin and E_i is called the expected chromaticity line for pixel i. All colors on this line are perceived as the same color but with a different brightness. When comparing a pixel value from the current frame to the background model, there is a point $\alpha_i E_i$ for which the distance from I_i to $\alpha_i E_i$ is minimal. This distance is called

the chromaticity distortion, the difference in color. The value α_i is called the brightness distortion, the difference in brightness.

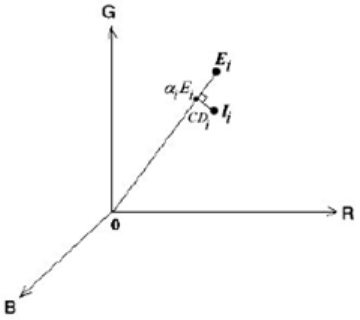


Figure 9.1: The RGB cube with points E and I.

If the chromaticity distortion or the brightness distortion exceeds certain thresholds, the pixel is classified as a motion-pixel.

9.3.1 Building the background model

The background model is created using a set of N frames without motion. We have used $N = 50$ for our experiments, which equals to two seconds of video. First, for each pixel i the mean RGB value over N frames is calculated. The mean R value of pixel i for example, is calculated as follows:

$$\mu_R(i) = \sum_{j=1}^{N} I_R(i_j) \tag{9.1}$$

where i_j is pixel i in frame j. This results in the expected color value

$$E_i = [\mu_R(i), \mu_G(i), \mu_B(i)] \tag{9.2}$$

for each pixel i. Next, the standard deviation

$$s_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)] \tag{9.3}$$

is calculated, where $\sigma_R(i)$ is the standard deviation for the I_R values of pixel i over N frames. The value s_i is used for normalizing the mean RGB values.

As mentioned above, the brightness distortion α_i can be found by minimizing the distance between I_i and the line OE_i:

$$\phi(\alpha_i) = (I_i - \alpha_i E_I)^2 \tag{9.4}$$

In terms of RGB values and taking the normalization into account, the equation for α_i can be written as:

$$\begin{aligned} \alpha_i &= \underset{c}{\operatorname{argmin}} \left[\begin{array}{l} \left(\frac{I_R(i)-c\mu_R(i)}{\sigma_R(i)}\right)^2 + \\ \left(\frac{I_G(i)-c\mu_G(i)}{\sigma_G(i)}\right)^2 + \\ \left(\frac{I_B(i)-c\mu_B(i)}{\sigma_B(i)}\right)^2 \end{array} \right] \\ &= \frac{\left(\frac{I_R(i)\mu_R(i)}{\sigma^2_R(i)} + \frac{I_G(i)\mu_G(i)}{\sigma^2_G(i)} + \frac{I_B(i)\mu_B(i)}{\sigma^2_B(i)}\right)}{\left([\frac{\mu_R(i)}{\sigma_R(i)}]^2 + [\frac{\mu_G(i)}{\sigma_G(i)}]^2 + [\frac{\mu_B(i)}{\sigma_B(i)}]^2\right)} \end{aligned} \tag{9.5}$$

The chromaticity distortion CD for pixel i can be calculated by taking the absolute difference between I_i and $\alpha_i E_i$. In terms of RGB values this becomes:

$$CD_i = \sqrt{\frac{I_R(i) - \alpha_i\mu_R(i)}{\sigma^2_R(i)} + \frac{I_G(i) - \alpha_i\mu_G(i)}{\sigma^2_G(i)} + \frac{I_B(i) - \alpha_i\mu_B(i)}{\sigma^2_B(i)}} \tag{9.6}$$

Two other values are also calculated: the variation of the brightness distortion α_i and the variation of the chromaticity distortion b_i. These are defined as:

$$a_i = \sqrt{\frac{\sum_{i=1}^{N}(\alpha_i - 1)^2}{N}} \tag{9.7}$$

$$b_i = \sqrt{\frac{\sum_{i=1}^{N}(CD)^2}{N}} \tag{9.8}$$

Note that, although the name may suggest otherwise, these values are actually the standard deviations of the brightness and the chromaticity distortion. Using these two values, the normalized brightness distortion and the normalized chromaticity distortion can be calculated:

$$\overline{\alpha_i} = \frac{\alpha_i - 1}{\alpha_i} \tag{9.9}$$

$$\overline{CD_i} = \frac{CD_i}{b_i} \tag{9.10}$$

At this point, each pixel is modeled by a tuple of four values: (E_i, s_i, a_i, b_i). Now, for each new frame, pixel values can be compared to the model to detect motion. For each pixel the normalized chromaticity distortion and the normalized brightness distortion are calculated and compared to certain thresholds. Motion

is detected at a pixel if the normalized chromaticity distortion is higher than τ_{CD} or if the normalized brightness distortion is below τ_{α}.

Thresholds τ_{CD} and τ_{α} are selected by creating a histogram of the normalized chromaticity distortions and a histogram of the normalized brightness distortions of all pixels of all frames in the training set. Below is an example of such histograms:

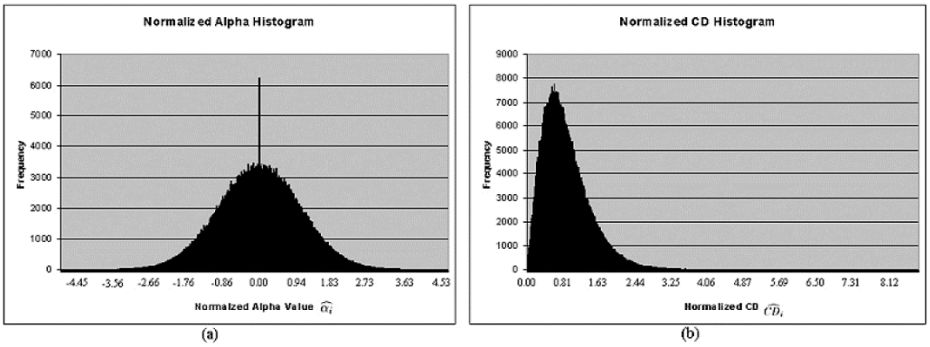


Figure 9.2: Histograms of the (a) normalized Alpha (brightness) and (b) CD (chromaticity) values

By choosing a percentage of all pixels of the training set which should be correctly classified as background pixels, a chromaticity distortion threshold can be selected. Note that in fact all pixels should be classified as background, but this would result in too high a threshold.

9.3.2 Adaptive background model

The model described above is static. Once all values are calculated, the model is not changed, although the scene for which the model is made could be changing due to different lighting conditions or deposited objects. To handle the situation of a changing background, we have altered the algorithm described above to get an adaptive version.

To accomplish this, the entire training set is kept in memory and for each new frame the oldest frame in the training set is replaced by the current frame, but only those pixels where no motion is detected are replaced. Ideally, the background model should be recalculated completely at this point, but for real-time applications this is too time-consuming. For this reason, only one scanline of the model is recalculated.

The global chromaticity distortion threshold cannot be used in this adaptive version of the model, so it is changed into a per-scanline threshold. Each time the

background model for a scanline is updated, the chromaticity distortion threshold for that line is recalculated.

Another difference between the motion detection method by Horprasert et al. and our method is the use of an upper bound for the brightness distortion, where the lower bound is denoted by $\tau_{\alpha 1}$ and the upper bound by $\tau_{\alpha 2}$. We classify a pixel as motion pixel when the brightness distortion is below $\tau_{\alpha 1}$ or above $\tau_{\alpha 2}$.

9.3.3 Post processing

The output of the motion detection algorithm contains noise. This is especially true when used with compressed video input, because the algorithm was designed to be used with uncompressed data. For clearing the noise from the images, a number of image filters has been used, such as removing salt-and-pepper noise, erosion, dilation and median filtering.

For object tracking, the output of the motion detection has to be converted to a list of regions with motion. Each motion-pixel is assigned a region ID, based on 8-connectivity.

9.4 Object tracking

For each input frame, the motion detection algorithm yields a list of regions, or blobs. A blob, short for 'binary large object', is a coherent group of pixels where motion was detected. Every object moving through the scene will create such a blob in a series of successive frames. The task of the object tracking algorithm is linking the positions of these blobs to a single object and to find out the path which has been followed by the object.

Figure 9.3 illustrates this. The object tracking algorithm should report one object for each frame and for each of these frames, the object has to be given the same identifier.

For each frame, the task is to link each blob to an object that was present in the previous frame. One possibility for finding a corresponding object for each blob is comparing the positions of the blobs with all positions of objects in the previous frame. An improved method would be to predict the position each object would have in the current frame and to compare those to the positions of the blobs. This method is of course very dependent on the method of motion prediction used.

For speed reasons, we chose to use yet another method for linking blobs to objects. We decided to use the bounding boxes of the blobs for tracking. Object positions and estimated positions are also stored as a bounding box.



Figure 9.3: Some frames from a sequence where one person walks through the scene.

9.4.1 Data structure

First, we define the data structures used by the object tracking algorithm. The inputs for the algorithm are the blobs detected by the motion detection algorithm. These blobs have the following properties:

- Position of the bounding box
- A list of pixels belonging to the blob
- Color model

The color model contains the average color for 32 parts of the blob. We chose to divide the blobs vertically into 32 equal parts. Figure 9.4 is a schematic view of such color models.



Figure 9.4: Output of the motion detection algorithm: blobs and their color models.

The object tracking algorithm keeps an internal list of tracked objects. For each frame, these objects are compared with the detected blobs and if they correspond, they are updated with information of these blobs. Objects have the following properties:

- Object ID
- A history of a maximum of 25 positions of this object in previous frames
- The predicted position of this object for the next 25 frames
- Color model
- Type

Object positions are stored for the prediction of movement. The object type can be either be simple or compound, which will be explained later.

Before explaining each step of the object tracking algorithm, we show a typical situation in figure 9.5: the motion detection algorithm has detected two blobs and the object tracking algorithm was tracking two objects. The blobs correspond with the predicted positions of the objects, so the objects will be updated with the position and color information of the blobs. The third image shows all previous and predicted positions of the tracked objects.

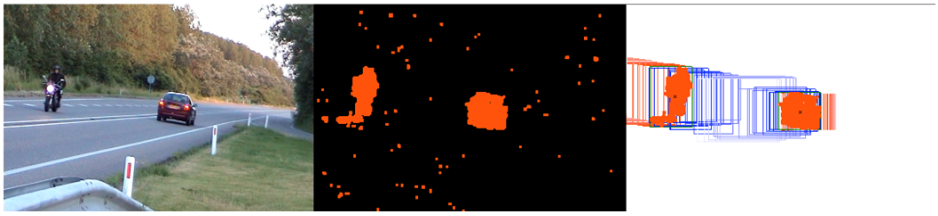


Figure 9.5: An input frame, the detected blobs and the previous and estimated object positions.

9.4.2 Object motion prediction

Using the coordinates of the bounding boxes of the previous positions of an object, a prediction can be made for the position of these bounding boxes in future frames. A bounding box of an object position at a certain time is actually a tuple $(t_k, X_1, Y_1, X_2, Y_2)$. A prediction has to be made for these tuples with values t_{k+1} and higher, where t_k is the value of t for the current frame. We chose to use a quadratic approximation of $(t_k, (X_1 + X_2)/2)$ and $(t_k, (Y_1 + Y_2))/2)$.

With this approach, the size of the predicted bounding box will be same as the current size. (Figure 9.5 illustrates this.)

A quadratic approximation of a series of tuples $(t_1, X_1), \ldots, (t_n, X_n)$ can be calculated by a least-squares fit of a quadratic function. The quadratic function can be described by the function $y = ax^2 + bx + c$. We try to find the values for a, b and c so that the error-value, defined as

$$\epsilon(a, b, c) = \sum_{i=1}^{n} (at_i^2 + bt_i + c - X_i)^2 \tag{9.11}$$

has the smallest value.

It is known that if the partial derivatives of the error function with respect to a, b and c are all equal to zero, the error function has the least value. The partial derivative of the error function with respect to a is

$$\frac{\partial \epsilon}{\partial a} = \sum_{i=1}^{n} 2(at_i^2 + bt_i + c - X_i) \tag{9.12}$$

Setting this equal to zero and rearranging gives the following formula:

$$a[\sum_{i=1}^{n} t_i^2] + b[\sum_{i=1}^{n} t_i] + cn = \sum_{i=1}^{n} X_i \tag{9.13}$$

The same can be done for the partial derivatives with respect to b and c, which yields these formulas:

$$a[\sum_{i=1}^{n} t_i^3] + b[\sum_{i=1}^{n} t_i^2] + c[\sum_{i=1}^{n} t_i] = \sum_{i=1}^{n} X_i \tag{9.14}$$

$$a[\sum_{i=1}^{n} t_i^4] + b[\sum_{i=1}^{n} t_i^3] + c[\sum_{i=1}^{n} t_i^2] = \sum_{i=1}^{n} X_i \tag{9.15}$$

These three formulas can be solved and the resulting a, b and c values can be used for the quadratic prediction function.

9.4.3 Rule-based object tracking

For object tracking, the task is to find corresponding blobs in the current frame for the objects being tracked. We define this correspondence in two different ways:

- A blob belongs to a tracked object, if the blob covers more than 50% of the predicted bounding box of the object
- An object contains a blob, if more than 50% of the area of the blob is within the area of the predicted bounding box of the object

Note that this means that an object can have no more than one blob belonging to it and that a blob can be contained by no more than one object.

To illustrate the concepts of belongs to and contains, a few schematic examples will be given. In these examples, a red circle stands for a blob and a green square stands for the predicted position of an object. Figure 9.6 shows the ideal situation: A blob belongs to an object (it covers more than 50% of the bounding box) and the object also contains the blob (at least 50% of the blob is inside the bounding box).

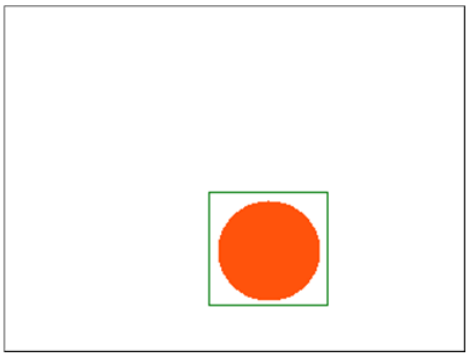


Figure 9.6: A blob belonging to an object, which also contains the blob.

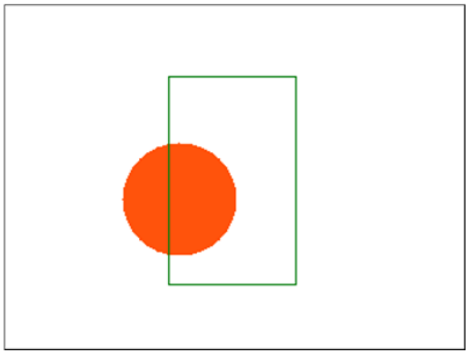


Figure 9.7: An object containing a blob, which does not belong to it.

Figure 9.7 shows that a blob can be contained by an object, but that the blob does not have to belong to it.

We need to find not only one object which a blob belongs to, but every object a blob belongs to. Also, we want to know every blob that is contained by an object. At this point, we introduce the Type property of an object: The Type property of an object can be either simple or compound. Simple objects are all objects which can be tracked by the straightforward version of the object tracking algorithm in which every blob corresponds to no more than one object.

Compound objects are virtual objects which consist of two or more objects which are currently occluding or occluded. The object tracking algorithm will track these objects just as simple objects, but will also track the actual simple objects which are involved.

The object tracking starts with an empty list of tracked objects. For each new input frame the following is done:

- For each blob, all objects it belongs to are searched for. A matrix B is created, where the value at $B_{i,j}$ equals 1 if blob i belongs to object j and it equals 0 otherwise.
- For each tracked object, the blobs it contains are determined. A matrix C is created, where the value at $C_{i,j}$ equals 1 if object i contains blob j, otherwise this value is 0.
- For each blob k, calculate the number of objects the blob belongs to:

$$b_k = \sum_i B_{k,i} \tag{9.16}$$

- For each object l not being part of a compound object, calculate the number of blobs contained by that object:

$$c_l = \sum_j C_{l,j} \tag{9.17}$$

- For each blob k, follow these rules to determine which objects need to be updated:
  - $b_k = 0$
    There is no object which this blob belongs to. This situation will be handled later.
  - $b_k = 1$
    This blob belongs to one object l. Check the c value for that object and do the following:
    - $c_l = 0$
      The object does not contain any blob. Apparently, the predicted position of the object and the actual position are differing. Update the object with information of the blob.
    - $c_l = 1$
      The object contains a blob. We assume that this blob is the blob we are currently checking. Update the object with information of the blob.
    - $c_l > 1$
      The object contains more than one blob. There are two possible cases:
      - The object is a simple object. We assume that the object contains the current blob and several other blobs and that all these blobs actually belong to one object. The object is updated with information of all blobs.
      - The object is a compound object. We assume that the compound object has split up into one or more simple objects and

at most one compound object. For all blobs contained by the object, try to find a corresponding simple object by comparing their predicted positions to the blob positions. In other words, try to find the simple objects which contain the blobs.

For all blobs for which no object is found, try to use the color model to find the corresponding object. The object with the closest matching color model will be updated with the information of a blob.

- $b_k > 1$
  This blob belongs to more than one object. We assume this is the result of two or more objects occluding each other. All objects this blob belongs to are combined into a new compound object. Note that an existing compound object can be one of these objects.

  The c value for each of the objects is not checked at this point. We assume that objects do not split up into several parts because of failing motion detection at the same time that it starts occluding an object.

- For each object l not updated yet, check the c value of the object. We then have the following possibilities:
  - $c = 0$
    The object does not contain a blob. It is possible that this is a simple object which is part of a compound object. This will be handled later.
  - $c = 1$
    The object contains one blob. The object is updated with information of that blob.
  - $c > 1$
    The object contains more than one blob. If the object is a simple object, we assume that all blobs are part of the object. The object will be updated with the combined information of these blobs.

    If the object is a compound object, we assume the object has split into one or more simple objects and at most one compound object. Use the previous rule for the case $b_k = 1$, $c_l > 1$ and with a compound object.
- Each object that is still part of a compound object is updated with its predicted position.
- For each blob that does not belong to an object and that is not contained by an object, create a new object.
- Each object that has not been updated at this point, is deleted from the list of tracked objects.

There are some assumptions made in the above algorithm, which might not always be true for some situations. Consequently, we expect the algorithm to fail for those situations. For example, we assume that objects can only be detected as two separate blobs because of failing motion detection, but we do not take into

account the possibility of an object actually splitting up. This can happen when someone steps out of a car, or when someone leaves a suitcase behind.

9.5 Results

We have tested the output of the object tracking algorithm on a set of six videos for which we created a ground truth. Table 9.1 gives a description for each video (see also Figure 9.8).

Number	Description
1	one person walking
2	a car driving
3	two persons walking
4	two persons walking, crossing paths
5	one person walking, partial occlusion
6	PETS2000 test video

Table 9.1: Test videos used in our experiments.

For our experiments, we define the following: An object is detected if the area of the object in the ground truth covers at least 50% of the area of an object in the program output, or if the area of the object in the program output covers at least 50% of the area of the object in the ground truth.

For each frame of the input video, the following values are calculated:

- G - # of objects in the ground truth for this frame
- P - # of objects in the program output for this frame
- C - # of correctly detected objects for this frame
- I - # of false detections for this frame

Note that $(I + C) = P$.

With these values, we can evaluate the object tracking algorithm. We have used the following benchmarks:

- The detection rate: For all frames, the percentage of objects correctly detected, C/P.
- The misdetection rate: For all frames, the percentage objects detected when no object was present, I/P.
- The mean error for the position of detected objects. The error-value for one object-position is:

$$\epsilon = (G_{x1} - P_{x1})^2 + (G_{y1} - P_{y1})^2 + (G_{x2} - P_{x2})^2 + (G_{x2} - P_{x2})^2 \quad (9.18)$$

where the corners of the bounding box are $(x1, y1)$ and $(x2, y2)$.

- The standard deviation of the error.
- The probability of an object ID change between two frames.

The results that we obtained can be seen in table 9.2.

	Video number					
	1	2	3	4	5	6
Detection	100.0	91.76	100.0	100.0	94.74	44.80
Misdetection	0.00	21.43	0.00	5.88	0.00	2.74
Mean error	2.75	4.94	9.48	3.85	4.80	7.84
Error std.dev.	3.47	4.29	4.99	3.21	4.15	4.32
P(id switch)	0.06	0.00	0.09	0.00	0.00	0.03

Table 9.2: Results from our object tracking experiments.

Note that for all videos 1-5, the detection rates are high and the misdetection rates are low. For the PETS2000 video which we chose for its complexity, it clearly shows an area for future improvement in our decision rules. In the current implementation, no rules exist in our algorithm for a situation which occurs in this video: A car being parked and someone stepping out of it, which results in 1 object becoming 2 objects. This problem will be addressed with an update to our rules.

9.6 Conclusions and future work

We have shown that using a rule-based algorithm for object tracking, we can reliably track multiple objects under several complex situations such as crossing paths, occlusion, and variable lighting. Our current system needs to be refined toward handling the situation where a single object becomes multiple objects.

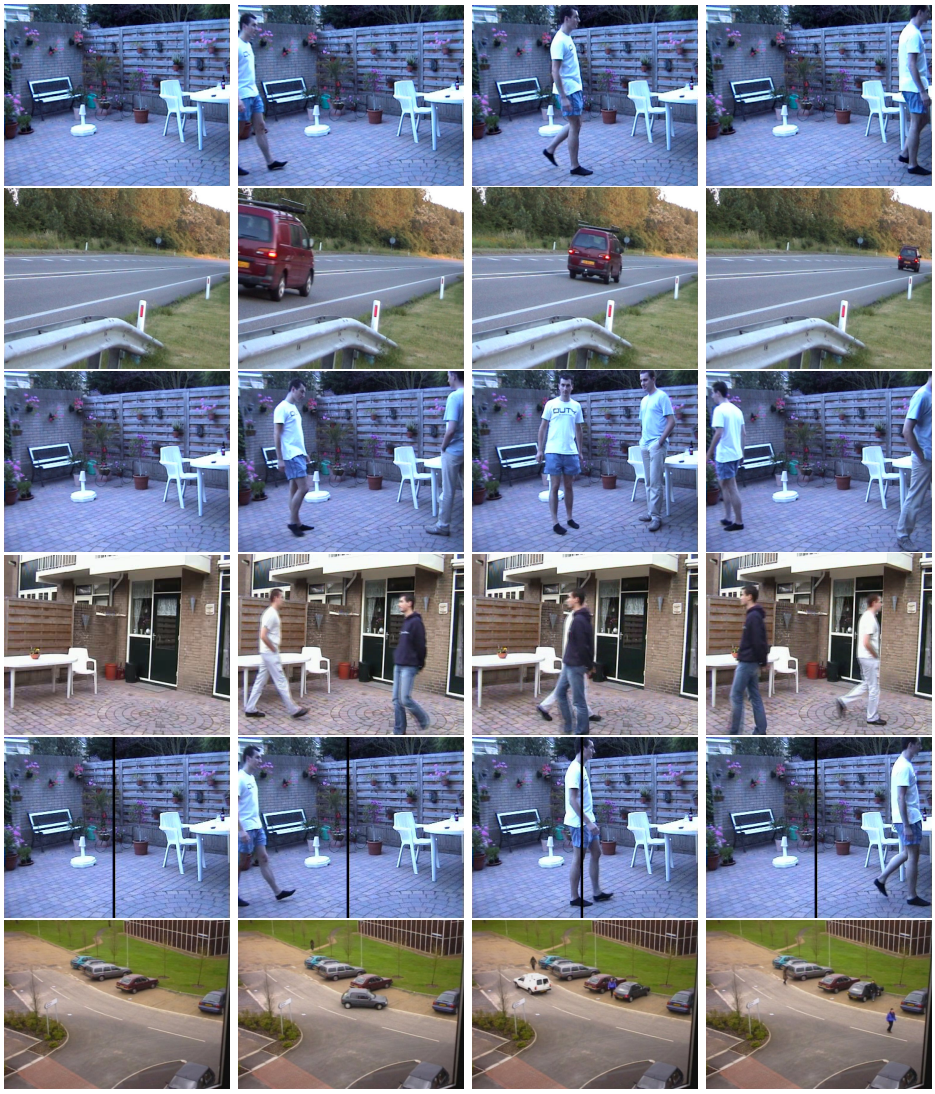


Figure 9.8: Some frames from the six video sequences used for testing the object tracking algorithm.

Chapter 10

Hybrid Maximum Likelihood Similarity

In this chapter we present an object tracking system which allows interactive user feedback to improve the accuracy of the tracking process in real-time video. In addition, we describe the hybrid maximum likelihood similarity, which integrates traditional metrics with the maximum likelihood estimated metric. The hybrid similarity measure is used to improve the dynamic relevance feedback process between the human user and the objects detected by our system.

10.1 Introduction

Human Computer Interaction (HCI) will require the computer to have similar sensory capabilities as humans including face [7] [36] [77] [79] and body [11] [27] [35] [40] [54] [79] [81] [96] understanding.

This chapter presents an interactive video tracking system which includes real-time user feedback in both motion detection and object tracking. The feedback from the user is applied in real-time, so the change in the tracking results is immediately visible.

Tracking and identifying moving objects in images from a stationary camera, such as people walking by or cars driving through a scene, has gained much attention in the recent years. It can be a very useful technique for human-computer interaction, the next-generation games, and for surveillance applications [96].

We developed an object tracking system [58] that can analyze live input streams from any video input device and which is able to output the locations, unique identifiers and pictorial information of the moving objects in the scene. All components are plug-ins, so in theory any method for object segmentation, tracking

and user feedback can be inserted.

Object detection and identification however is a topic that has its unique set of problems that still are not fully addressed. Multiple object tracking is complicated by the fact that objects can touch or interact with each other, can occlude another, even leave the scene and come back again. And the usual problems with single object tracking, like illumination changes, partial occlusion or object deformation, still apply as well.

To get improved object tracking results, we investigate methods to include user feedback for detecting moving regions and to ignore or focus on specific tracked objects. Our object tracking framework includes color-based relevance feedback [67] functionality at both the segmentation and tracking level.

However, we have seen from earlier experiments [78] that matching the users feedback to new input from the segmentation and tracking algorithms using standard visual similarity metrics, performs poorly in some situations. Similarity metrics that adjust to the true visual similarity are needed. Especially for the constantly slightly changing object appearances, differences in color-feature values do not always have the same visual difference, so we are investigating new similarity metrics that are applied to the object tracking framework, but are applicable in general visual similarity matching as well.

10.2 Related work

There has been significant research on motion tracking - an extensive review has been written by Yilmaz et al. [96], that gives a clear overview of object tracking and the challenges and limitations that current object tracking systems face. Notable scientific meetings on object tracking and related research include the Performance and Evaluation of Tracking and Surveillance PETS) and Video Surveillance & Sensor Networks (VSSN).

Relevance feedback [67] is an interactive method that has been successfully introduced into text and image queries. It is an interactive query process where a computers internal representation of the object that a user is interested in is continually updated by providing the system with information about the relevancy of the query outcome.

10.3 Visual similarity

10.3.1 The maximum likelihood training problem

In chapter 7 of this thesis, the Multi-Dimensional Maximum Likelihood (MDML) paradigm was presented as a method for determining similarity of feature values using the 2D distribution of a training set of these feature values. It was shown

that the multi-dimensional approach outperforms the 1D maximum likelihood similarity measure.

In general it is difficult to find sufficient training examples to arrive at a statistically representative model of the underlying probability density function. This fundamental training problem motivates the Hybrid Maximum Likelihood Similarity Measure described next.

10.3.2 Hybrid maximum likelihood similarity

In practice, the L_2 distance measure is typically a rough but not perfect fit to the underlying similarity measure. Therefore, we propose the Hybrid Maximum Likelihood Similarity (HMLS) measure which interpolates between the L_2 distance and the maximum likelihood distance to both obtain a better similarity measure and address the training problem from Section 10.3.1. At both the pixel and object-level feedback, the general algorithm for using the hybrid maximum likelihood similarity measure for a color feature is:

- For each feature vector element x:
  - Initialize a histogram H_x to $H_x[i][j] = (i - j) * (i - j)$
  - Normalize H_x so that the sum of all $H_x[i][j]$ is 1
- When calculating the similarity between to elements at position x with values i and j:
  - Use $H_x[i][j]$
- After feedback from the user:
  - Create a new histogram $H_{temp}[i][j]$ for each feature vector element
  - Fill H_{temp} with the feature values from the examples from the user
  - Normalize H_{temp}
  - Set $H_x = w * H_x + (1 - w) * H_{temp}$

In this algorithm, i and j range over the possible values of the feature vector element, in our case $[0 \dots 255]$. The last step in the algorithm generates a new version of the histogram that will converge to the true similarity distribution if enough training samples are given.

10.4 Relevance feedback in object tracking

Figure 10.1 gives an overview of our object tracking system and the location of the relevance feedback module in it.

For our first experiments, we decided to use color-based relevance feedback, so we have used our own adaptation of a color-based motion detection method developed by Horprasert et al. [27]. The main idea of their method is to decompose the

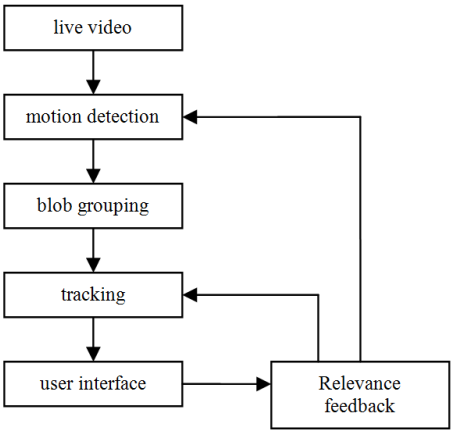


Figure 10.1: The components of the object tracking system with relevance feedback.

difference between the current color information and the modeled background for every pixel into a chromaticity (color) component and a brightness component.

As mentioned in chapter 9, one assumption that the authors of this motion detection method made, was that the lighting would stay roughly constant. In real world applications however, the light can change gradually. Thus, we implemented an adaptive version of their model to compensate for dynamic real world lighting. Small parts of the background model are continuously updated with new data, to compensate for these gradual changes in lighting. Another effect of this adaptation is that deposited objects can be added to the background model if they do not move for a given period of time. For further details, please refer to section 9.3.2 of chapter 9.

We use bounding boxes as a method for object tracking. Objects are considered either simple or compound. Simple objects are objects that can be tracked by the straightforward version of the object tracker, in which every blob corresponds to no more than one object. In the case of object interactions, or overlapping objects, there is ambiguity as to which blob belongs to which object. We define compound objects as virtual objects which consist of two or more objects which are currently interacting in some fashion. The object tracker will track these objects just as simple objects, but will also track the actual simple objects which are involved.

10.4.1 Pixel-level feedback

Our object tracking system continuously compares pixel values to the modeled background to decide whether a pixel should be considered as part of a moving

object. The relevance feedback component can change this decision by learning from user input. An example is given in figure 10.2. In this case, the user indicates that pixels that look like the selected part of the image (the brick wall) should never be considered to be an object, even if the background model indicates that it should, which could happen in case of fast lighting changes.



Figure 10.2: User feedback for the object segmentation algorithm: selecting a negative example.

The user can supply feedback while the tracking system is running. The selected positive and negative examples are immediately included in the decision process, so the effect on the object tracking results is instantly visible.

The HMLS is trained using all pairs of pixels in the area that the user has selected.

10.4.2 Object-level feedback

We will now present an example of using feedback for the tracking algorithm. Figure 10.3 shows a frame from a sequence in which a person is leaving an object behind. In figure 10.4, the user selects a positive example for the object tracking algorithm. In this case, objects with similar color will always remain marked as foreground and they will not be added to the background model, which is the normal behavior for adaptive object tracking algorithms. Figure 10.5 shows the object being classified as a foreground object based on the user feedback.

Negative examples for the object tracking algorithm are useful for marking objects that are not interesting to the user. The tracking algorithm will ignore objects that are similar to the examples supplied by the user.

The HMLS metric is trained using information on the tracked object from each frame in which it is still tracked.

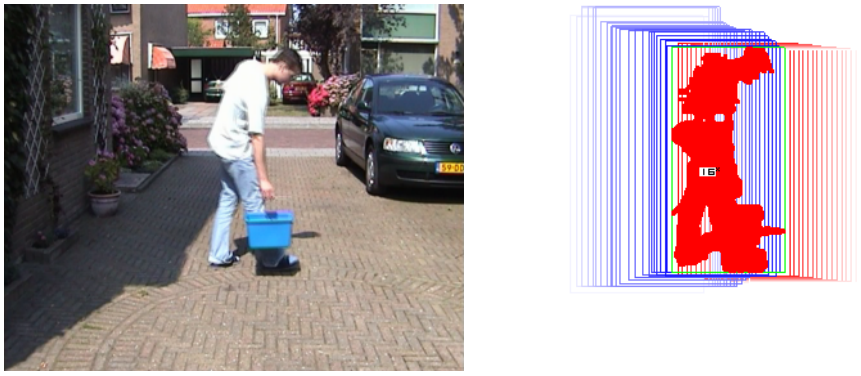


Figure 10.3: One frame from a sequence where someone leaves an object behind, together with the object tracking results.

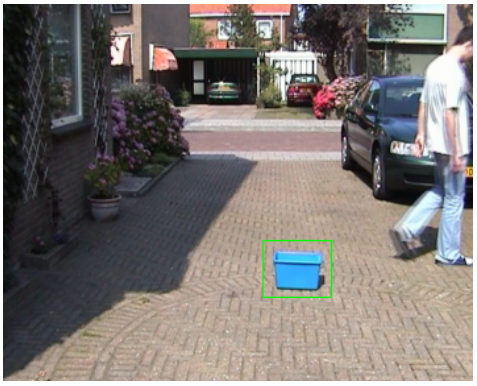


Figure 10.4: The user selects a positive example for the object tracking algorithm.

10.5 Conclusions and future work

Based on user surveys, our interactive video tracking system has shown that including relevance feedback in the motion detection and object tracking process is intuitive and promising.

The strong point of the HMLS is that it gives the benefits of maximum likelihood similarity estimation while also addressing the limited training set problem.

In future work, we are interested in treating the temporal space as a wavelet based texture [73], learning optimal features [38], and performing more extensive quantitative evaluation including comparing different similarity measures.



Figure 10.5: The object tracking using the positive example. The object will not be added to the background model and stays visible as a tracked object.

Appendix A

RetrievalLab

A.1 Introduction

RetrievalLab is a tool for illuminating content based retrieval. It can be used in research and in educational workshops to explore, compare, and demonstrate the use of features, databases, images and evaluation methods in content based retrieval tasks. Regarding education, the intention is that students will be able to learn about content based retrieval without spending numerous months creating a custom system. In addition, RetrievalLab has a plugin architecture that makes it possible for users to add new functionality.

A.2 Related work

There has been significant prior work in content based retrieval (CBR) systems. See Datta et al. [12] for a recent survey. A few examples of such systems include the the GNU Image Finder and imgSeek.

RetrievalLab was inspired by Matlab [15], except that the native data structures and functionality are specifically designed toward facilitating content based retrieval (CBR) research. For example, in retrieval contexts, we have the notion of a database of multimedia objects which is common to most CBR systems. The typical usage in Matlab would be to load the pictorial, feature, and tag information into separate arrays. In RetrievalLab, there is the fundamental notion of a database object so the user can issue a command like

```
MyDatabaseObject = loaddatabase("MyImageDirectory")
```

More information will be given in Section A.3, but the fundamental notion is that databases are now native objects which can be updated, copied, manipulated, and analyzed or used as sets of positive and negative examples in machine learning.

RetrievalLab provides the following:

- Programming interface - Matlab-like algebraic/functional.
- Database data structure is a fundamental and native object.
- Free (unlike Matlab or Mathematica)
- User extendable plug-in architecture with sample plug-ins.
- Basic functionality for all research stages: loading databases, feature extraction, machine learning, visualization of results, and quantitative benchmarking and evaluation.

While there are many research systems which provide some of the items above, we are currently not aware of any other system which provides all of them.

A.3 Example usage

This section will demonstrate a few uses of the RetrievalLab program, by showing the commands that are needed to accomplish certain tasks. A more extensive description is available at http://press.liacs.nl/researchdownloads/retrievallab/.

A.3.1 Image retrieval

In a typical image retrieval task, a database is loaded into a variable and both ground truth tags and features are loaded into memory. We added support for loading ground truth for the MIRFLICKR [31] [32] and IMAGECLEF [55] datasets.

```
> db=loaddatabase("D:/MIR-Flickr/")
> loadmirflickrtags(db, "D:/tree_r1.txt")
> loadfeature(db, "hsv")
```

After that, an image is loaded and the same feature as was used for the database is calculated.

```
> im=loadimage("D:/Trees/tree1.jpg")
> updatefeature(im, "hsv")
```

With the database and the image, a search query can be executed that results in an index.

```
> index=searchimage(db, im)
> displayindex(index)
```

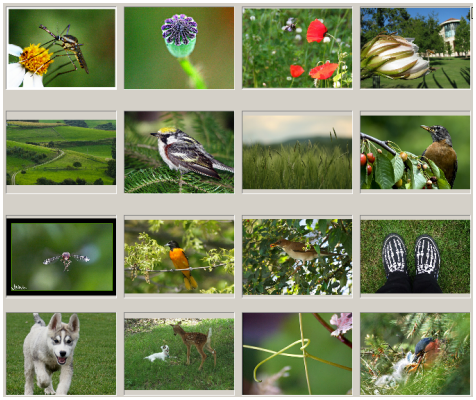


Figure A.1: Result of the 'displayindex' function.

Results can be displayed in the form of a standard grid, or a 2D view with results centered around the best matching image, where the feature distance is used to determine the distance in the image.

```
> displayindexmap(index)
```



Figure A.2: Result of the 'displayindexmap' function.

Finally, we evaluate the results. In this case, we evaluate the tree_r1 tag using:

```
> evaluateindex(index, "tree_r1")
```

which returns the MAP (mean average precision) value with respect to the ground truth.

A.3.2 Visual concept detection

In a typical visual concept detection query, two databases with positive and negative examples are loaded into variables and after that, the images are segmented and features are calculated. With these two databases, a visual concept can be created based on the selected classifier.

First, the positive database is loaded and the enhanced wavelet representation feature [57] is added to the image segments.

```
> dbpos=loaddatabase("D:/Trees/")
> segmentimage(dbpos, "sift")
> updatefeature(dbpos, "hsv")
> updatefeature(dbpos, "lbp")
> updatefeature(dbpos, "wavelet")
```

Note that each image segment now has three different features attached to it. All three features will be used. For the negative examples, we do the same. (Segmenting and feature extraction is omitted for brevity.)

```
> dbneg=loaddatabase("D:/NotTrees/")
```

The concept can then be learned with a selected classifier. Currently, nearest neighbor, SVM and neural network classifiers are supported.

```
> concept=learnvisualconcept(dbpos, dbneg, "svm")
```

After this an image is loaded and segments and features are added. We can now apply our visual concept to this image.

```
> findvisualconceptlocations(im, concept, "tree")
> displayimage(im)
```

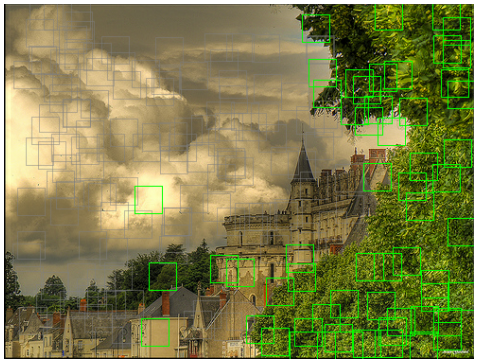


Figure A.3: Result of detecting a concept.

A.4 Discussion, conclusions and future work

The current system gives a programming interface to content based retrieval functionality, which is both user extendable and focuses on the typical CBR components including diverse features [12] [74] [80], distance measures [76], and classifiers [76]. We think that having a database as a native object facilitates many typical database operations. In addition, RetrievalLab has built-in support for the MIRFLICKR [31] [32], IMAGECLEF [55] test sets.

In future work, we will add plug-ins for video database analysis. Note that our framework was designed for generality regarding media types and should be able to accommodate most kinds of media.

Bibliography

[1] H. Bay, Tuytelaars T., and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (LNCS vol. 3951)*, pages 404–417. Springer, 2006.

[2] R.E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, USA, 1995.

[3] M. Blighe and N.E. O'Connor. Myplaces: detecting important settings in a visual diary. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 195–204, 2008.

[4] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[5] T.H. Chalidabhongse, K. Kim, D. Harwood, and L.S. Davis. A perturbation method for evaluating background subtraction algorithms. In *Proceeding of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 110–116, 2003.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[7] I. Cohen, N. Sebe, A. Garg, M.S. Lew, and T.S. Huang. Facial expression recognition from video sequences. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'02), vol. II*, pages 121–124, 2002.

[8] A. Colombari, A. Fusiello, and V. Murino. Segmentation and tracking of multiple video objects. *Pattern Recognition*, 40:1307–1317, 2007.

[9] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[10] M. Cristani, M. Bicego, and V. Murino. Integrated region- and pixel-based approach to background modelling. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 3–8, 2002.

[11] R. Cucchiara. Multimedia surveillance systems. In *Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 3–10, 2005.

[12] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(1), April 2008.

[13] M. Egmont-Petersen, D. De Ridder, and H. Handels. Image processing with neural networks - a review. *Pattern Recognition*, 35:2279–2301, 2002.

[14] T. Ellis and M. Xu. Object detection and tracking in an open and dynamic world. In *Proceedings of Workshop on Performance Evaluation of Tracking and Surveillance*, page unnumbered, 2001.

[15] D.M. Etter. *Introduction to Matlab (2nd Edition)*. Prentice Hall, 2004.

[16] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, et al. Query by image and video content: the qbic system. *Computer*, 28:23–32, 1995.

[17] N. Funk. A study of the kalman filter applied to visual tracking. Technical report, University of Alberta, 2003.

[18] G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19:699–707, 2001.

[19] B. Gloyer, H.K. Aghajan, K.-Y. Siu, and T. Kailath. Video-based freeway-monitoring system using recursive vehicle tracking. In *Proceedings of the SPIE Symposium on Electronic Imaging: Image and Video Processing*, pages 173–180, 1995.

[20] B. Han, Y. Zhu, D. Comaniciu, and L.S. Davis. Kernel-based bayesian filtering for object tracking. pages 227–234, 2005.

[21] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: A real time system for detecting and tracking people. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 222–227, 1998.

[22] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:809–830, 2000.

[23] C. Harris and M. Stephens. A combined edge and corner detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[24] D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman., and L.S. Davis. *CAR-TR-678* - texture classification by center-symmetric auto-correlation, using kullback discrimination of distributions. Technical report, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland, 1993.

[25] D.-C. He, L. Wang, and J. Guibert. Texture discrimination based on an optimal utilization of texture features. *Pattern Recognition*, 21:141–146, 1988.

[26] L. He, C. Zou, L. Zhao, and D. Hu. An enhanced lbp feature based on facial expression recognition. In *Proceedings of the 27th Annual International*

Conference of the IEEE Engineering in Medicine and Biology Society, pages 3300–3303, 2005.

[27] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of the IEEE Frame-Rate Applications Workshop*, pages 1–19, 1999.

[28] T. Horprasert, D. Harwood, and L.S. Davis. A robust background subtraction and shadow detection. In *Proceedings of the Asian Conference on Computer Vision*, pages 983–988, 2000.

[29] T.S. Huang, S. Mehrotra, and Ramchandran K. Multimedia analysis and retrieval system (mars) project. In *Proceedings of the 33rd Annual Clinic on Library Application of Data Processing - Digital Image Access and Retrieval*, 1996.

[30] D.P. Huijsmans, S. Poles, and M.S. Lew. 2d pixel trigrams for content-based image retieval. In *Proceedings of the 1st International workshop on Image databases and Multi-Media search*, pages 139–145, 1996.

[31] M.J. Huiskes and M.S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, pages 39–43, 2008.

[32] M.J. Huiskes, B. Thomee, and M.S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, 2010.

[33] H. Jin, Q. Liu, H. Lu, and X. Tong. Face detection using improved lbp under bayesian framework. In *Proceedings of the Third International Conference on Image and Graphics*, pages 306–309. IEEE computer society press, 2004.

[34] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 41–56. MIT Press, 1999.

[35] N. Lazarevic-McManus, J. Renno, and G. A. Jones. Performance evaluation in visual surveillance using the f-measure. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 45–52, 2006.

[36] M.S. Lew. Information theoretic view-based and modular face detection. In *Proceedings of the 2nd. International Conference on Automatic Face and Gesture Recognition*, pages 198–203, 1996.

[37] M.S. Lew. Next generation web searches for visual content. *IEEE Computer*, 33:46–53, 2000.

[38] M.S. Lew, T.S. Huang, and K. Wong. Learning and feature selection in stereo matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:869–881, 1994.

[39] M.S. Lew and N. Huijsmans. Information theory and face detection. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 601–605, 1996.

[40] M.S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2:1–19, February 2006.

[41] J. Li and J. Wang. Real-time computerized annotation of pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:985–1002, 2008.

[42] L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10, 2003.

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

[44] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[45] F.J. Madrid-Cuevas, R. Medina Carnicer, M. Prieto Villegas, N.L. Fernández García, and Carmona Poyato. Simplified texture unit: A new descriptor of the local texture in gray-level images. In *Proceedings of the first Iberian conference on Pattern recognition and image analyis (LNCS2652)*, pages 470–477. Springer, 2003.

[46] T. Mäenpää, T. Ojala, M. Pietikäinen, and Soriano M. Robust texture classification by subsets of local binary patterns. In *Proceedings of the 15th international conference on pattern recognition, volume 3*, pages 3947–3950, 2000.

[47] J. Malo, J. Guttierrez, I. Epifanio, and F.J. Ferri. Perceptually weighted optical flow for motion-based segmentation in mpeg-4 paradigm. *Electronics Letters*, 36:1693–1694, 2000.

[48] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[49] S.J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80:42–56, 2000.

[50] M. Middendorf and H. Nagel. Vehicle tracking using adaptive optical flow estimation. In *Proceeding of the Workshop on Performance Evaluation of Tracking and Surveillance*, pages 42–56, 2000.

[51] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, 2005.

[52] H. Moravec. Visual mapping by a robot rover. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 598–600, 1979.

[53] C. Motamed. Motion detection and tracking using belief indicators for video surveillance applications. In *Proceedings of the 1st IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS2000)*, pages 58–63, 2000.

[54] W. Nam and J. Han. Motion-based background modeling for foreground segmentation. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 35–44, 2006.

[55] S. Nowak and M.J. Huiskes. New strategies for image annotation: Overview of the photo annotation task at imageclef 2010. In *CLEF (Notebook Papers/LABs/Workshops)'10*, 2010.

[56] A. Oerlemans and M.S. Lew. Interest points based on maximization of distinctiveness. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 202–207, 2008.

[57] A. Oerlemans and M.S. Lew. Minimum explanation complexity for mod based visual concept detection. In *Proceedings of the international conference on Multimedia information retrieval*, pages 567–576, 2010.

[58] A. Oerlemans, M.S. Lew, and E.M. Bakker. Detecting and identifying moving objects in real-time. In *Proceedings of the Conference of the Advanced School for Computing and Imaging*, pages 358–365, 2005.

[59] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28:40–48, 1995.

[60] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996.

[61] T. Ojala, M. Pietikäinen, and T. Mäenpää. Gray scale and rotation invariant texture classification with local binary patterns. In *Proceedings of the Sixth European Conference on Computer Vision*, pages 404–420, 2000.

[62] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:971–987, 2002.

[63] S. Philipp-Foliguet, J. GONYA, and P.-H. Gosselina. Frebir: An image retrieval system based on fuzzy region matching. *Computer Vision and Image Understanding*, 113:693–707, 2009.

[64] Y. Raja and S. Gong. Sparse multiscale local binary patterns. In *Proceedings of the 17th British machine vision conference, volume II*, pages 799–808, 2006.

[65] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[66] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–28, 1998.

[67] Y. Rui and T.S. Huang. *Relevance feedback techniques in image retrieval*, pages 219–258. Springer-Verlag, London, UK, 2001.

[68] Y. Rui, T.S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Proceedings of the International Conference on Image Processing, volume 2*, pages 815–818, 1997.

[69] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.

[70] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[71] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37:151–172, 2000.

[72] N. Sebe, T. Gevers, J. van de Weijer, and S. Dijkstra. Corner detectors for affine invariant salient regions: Is colour important? In *Proceedings of the International Conference on Image and Video Retrieval*, pages 61–71, 2006.

[73] N. Sebe and M.S. Lew. Wavelet based texture classification. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR), vol III*, pages 959–962, 2000.

[74] N. Sebe and M.S. Lew. Color-based retrieval. *Pattern Recognition Letters*, 22:223–230, February 2001.

[75] N. Sebe and M.S. Lew. *Texture features for content-based retrieval*, pages 51–85. Springer-Verlag, 2001.

[76] N. Sebe and M.S. Lew. *Robust Computer Vision: Theory and Applications*. Kluwer Academic Publishers, 2003.

[77] N. Sebe, M.S. Lew, I. Cohen, Y. Sun, T. Gevers, and T.S. Huang. Authentic facial expression analysis. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG)*, pages 517–522, 2004.

[78] N. Sebe, M.S. Lew, and D.P. Huijsmans. Toward improved ranking metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1132–1143, October 2000.

[79] N. Sebe, M.S. Lew, X. Zhou, T.S. Huang, and E.M. Bakker. The state of the art in image and video retrieval. In *Proceedings of the 2nd international conference on Image and video retrieval*, pages 1–8, 2003.

[80] N. Sebe, Q. Tian, E. Loupias, M.S. Lew, and T. Huang. Evaluation of salient point techniques. *Image and Vision Computing*, 21:367–377, 2003.

[81] M. Siddiqui and G. Medioni. Robust real-time upper body limb detection and tracking. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 53–60, 2006.

[82] N.T. Siebel and S. Maybank. Real-time tracking of pedestrians and vehicles. In *Proceedings of Performance Evaluation of Tracking and Surveillance PETS*, page unnumbered, 2001.

[83] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.

[84] M. Srikanth, J. Varner, M. Bowden, and D. Moldovan. Exploiting ontologies for automatic image annotation. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 552–558, 2005.

[85] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 15:76–84, 1995.

[86] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part 2. *IEEE Computer Graphics and Applications*, 15:75–85, 1995.

[87] M. Stricker and M. Orengo. Similarity of color images. In *Proceedings of SPIE - Storage and Retrieval of Image and Video Databases III, vol. 2*, pages 381–392, 1995.

[88] Q. Tian, N. Sebe, E. Loupias, M.S. Lew, and T.S. Huang. Content-based image retrieval using wavelet-based salient points. In *Proceedings of SPIE - Storage and Retrieval for Media Databases*, pages 425–436, 2001.

[89] L. Trujillo and G. Olague. Using evolution to learn how to perform interest point detection. In *Proceeding of the 18th International Conference on Pattern Recognition*, pages 211–214, 2006.

[90] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[91] R.C. Veltkamp and M. Tanase. A survey of content-based retrieval systems. In O. Marques and B. Furht, editors, *Content-Based Image and Video Retrieval*, pages 47–101. Kluwer, 2002.

[92] J.S. Walker. *A Primer on Wavelets and their Scientific Applications, Second Edition*. Chapman & Hall, London, 2008.

[93] L. Wang and D.-C. He. Texture classification using texture spectrum. *Pattern recognition*, 23:905–910, 1990.

[94] Q. Xiong and C. Jaynes. Multi-resolution background modeling of dynamic scenes using weighted match filters. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 88–96, 2004.

[95] M.-S. Yang, D.J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[96] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38, December 2006.

[97] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian. Distance learning for similarity estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:451–462, 2008.

[98] G.P. Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man and Cybernetics*, 30:451–462, 2000.

Nederlandse Samenvatting

In het huidige digitale tijdperk zijn zeer veel gegevens beschikbaar in de vorm van bijvoorbeeld foto's, video's en geluid. Deze hoeveelheid gegevens neemt dagelijks met onvoorstelbare snelheid toe. Een voorbeeld hiervan is YouTube, waar elke dag voor ongeveer zeven jaar aan video bijgeplaatst wordt. Ook op websites als Flickr zijn al miljarden foto's geüpload. Maar wat is het nut van al deze digitale informatie, als er niet op een handige manier in teruggezocht kan worden?

Dit proefschrift beschrijft een aantal onderzoeken die als doel hadden het terugzoeken van digitale afbeeldingen te vergemakkelijken. De technieken staan bekend als Content-Based Image Retrieval (CBIR) methodes, wat betekent dat de inhoud van afbeeldingen gebruikt wordt om de gebruiker te ondersteunen in zijn zoekproces.

De gangbare manier van zoeken op internet vindt plaats door middel van het invoeren van tekst. Als er gezocht wordt naar een afbeelding, dan is men hierbij afhankelijk van de bij de afbeeldingen geplaatste omschrijvingen. Helaas is het vaak zo dat niet de juiste omschrijving bij een afbeelding staat, of dat er zelfs helemaal geen omschrijving bij een afbeelding gegeven wordt. Denk hierbij bijvoorbeeld aan een serie vakantiefoto's die op internet gezet wordt. Er zal waarschijnlijk wel een omschrijving voor de gehele serie foto's zijn, zoals 'Vakantie in Frankrijk 2011', maar niet elke losse foto zal een specifieke beschrijving hebben, zoals bijvoorbeeld 'De Eiffeltoren'. In dit geval zullen de gangbare methoden van zoeken met tekst niet de gewenste resultaten kunnen geven, want er zijn immers geen tekstuele omschrijvingen beschikbaar aan de hand waarvan die ene specifieke afbeelding van de Eiffeltoren uit de serie gevonden kan worden.

Zoals hierboven al vermeld, onderzoeken de technieken uit dit proefschrift de inhoud van afbeeldingen en proberen hieruit informatie af te leiden die gebruikt kan worden voor zoekacties. Voor de gebruiker betekent dit dat hij of zij ook kan zoeken naar een afbeelding die lijkt op een afbeelding die hij of zij zelf al heeft, of dat er gezocht kan worden naar omschrijvingen die door de computer automatisch van afbeeldingen afgeleid zijn.

Een voorbeeld van de eerste manier, is het zoeken naar een afbeelding van een specifiek type auto. Als een gebruiker al een afbeelding heeft van het type auto, maar hij of zij wil er graag nog meer bekijken, dan kunnen CBIR technieken

uitkomst bieden. Deze technieken kunnen afbeeldingen met hetzelfde type auto terugvinden, zelfs als er verder geen informatie beschikbaar is bij de afbeelding. Er wordt dan gezocht op overeenkomsten tussen de inhoud van beide foto's. De computer kan zien of twee foto's op elkaar lijken.

Een voorbeeld van de tweede manier van zoeken op basis van beeldinhoud, is het zogenaamde 'visual concept detection', het automatisch herkennen van bepaalde visuele concepten of ideen op een afbeelding. Hierbij wordt een foto geänalyseerd door de computer en zullen er automatisch bepaalde woorden bij een foto geplaatst worden die uit deze automatisch analyse volgen. Hierbij kan gedacht worden aan 'gebouw', 'berg', 'zee', maar ook aan meer specifieke woorden als 'winkel', 'Mount Everest' of 'strand bij Noordwijk'. Uiteraard zullen deze concepten een keer door de computer geleerd moeten worden aan de hand van voorbeelden, maar hierna zal het geleerde concept gebruikt kunnen worden en zal er zonder tussenkomst van een mens bij elke afbeelding een lijst met woorden gemaakt kunnen worden. Hierna kan de gebruiker weer zoeken met tekst, zoals hij of zij gewend is. De foto van de Eiffeltoren uit de eerder genoemde serie vakantiefoto's kan dan toch gevonden worden.

In dit proefschrift worden drie wetenschappelijke bijdragen beschreven op het gebied van content-based image retrieval: het MOD paradigma, geconstruëerde textuur-patronen en de zogenaamde multi-dimensional maximum likelihood measure, een meerdimensionale aanpak voor het vergelijken van eigenschappen van afbeeldingen. Elk van deze bijdragen zal hieronder besproken worden.

Op het gebied van het terugzoeken van visuele informatie is een zeer uitdagend probleem van de laatste jaren het detecteren van visuele concepten, waarbij de computer gevraagd wordt om automatisch een beeld te voorzien van relevante steekwoorden. Op een fundamenteel niveau betekent dit dat de computer een vorm van begrip voor afbeeldingen heeft gekregen. Als de computer een strand, gebouw, gezicht of zonsondergang ziet, worden deze concepten herkend op basis van het beeld, net zoals een mens zou doen. De kleuren, vormen en andere eigenschappen (ook wel 'features' genoemd) worden hiervoor gebruikt. Het is tot zeer recent haast onmogelijk gebleken om dit probleem op te lossen.

De eerste bijdrage van dit proefschrift, te vinden in hoofdstuk 6, is een algoritme voor visuele concept detectie dat gebruik maakt van 'salient points', automatisch bepaalde punten in een afbeelding die interessant of opvallend zijn. In dit proefschrift wordt een nieuw paradigma voorgesteld dat MOD heet, wat staat voor Maximization Of Distinctiveness. Hierbij worden deze interessante punten geselecteerd op basis van hun onderscheidend vermogen. De MOD aanpak is getest op de meest uitdagende internationale wetenschappelijke test set en bleek een significante verbetering te geven ten opzichte van de beste methode uit de literatuur van visuele concept detectie. In tegenstelling tot de andere onderzoeken die salient points gebruiken voor het analyseren van beelden, generaliseert deze methode tot elk type beeldinhoud en elk type afbeeldingen.

In hoofdstuk 8 is de tweede bijdrage van dit proefschrift te vinden, op het gebied

van computationeel efficiënte textuur beschrijvingen. Eenvoudig omschreven, zegt een textuur beschrijving iets over het materiaal waar naar gekeken wordt, zoals bijvoorbeeld bakstenen of grind. Afbeeldingen die dezelfde textuur bevatten, kunnen teruggevonden worden als de door de computer bepaalde beschrijving van de textuur maar goed genoeg is. Textuur features zijn waarschijnlijk de meest gebruikte visuele eigenschappen in de computer vision. Op dit moment is de meest voorkomende textuur feature in de wetenschappelijke literatuur de 'local binary patterns' (LBP), een 256 dimensionale beschrijving van 3x3 patronen. Er is herhaaldelijk vastgesteld dat deze feature een hoge mate van accuraatheid heeft, maar dat het gebruik ervan ook een significante computationele rekenkracht vereist. In dit proefschrift wordt beschreven dat het mogelijk is om met grotere patronen dan 3x3 een vergelijkbare nauwkeurigheid te behalen als met LBP, maar dan met slechts 2 dimensies in plaats van 256. Dit verhoogt de computationale efficiëntie met een factor honderd en reduceert de hoeveelheid benodigd geheugen met een vergelijkbare factor.

De derde bijdrage van dit proefschrift, terug te vinden in hoofdstuk 7, is een beschrijving van de 'multi-dimensional maximum likelihood' (MDML) methode voor het vergelijken van eigenschappen van afbeeldingen. Op dit moment is op het gebied van computer vision de meest voorkomende manier van het vergelijken van eigenschappen van afbeeldingen de 'sum of squared differences' (SSD). Gebruikmakend van de theorie van meest aannemelijke schatters wordt in dit proefschrift beschreven dat het gebruik van SSD alleen optimaal is bij bepaalde aannames over de onderliggende kansverdeling van de ruis, in het bijzonder wanneer deze Gaussisch is. In dit proefschrift wordt het bekende computer vision probleem beschreven van het zoeken van overeenkomsten in afbeeldingen die uit twee zichtpunten gemaakt zijn, eenvoudig gezegd: een stereo-paar. Hierbij wordt voor elk punt uit de ene afbeelding een overeenkomend punt gezocht in de andere afbeelding. Met deze overeenkomsten kan dan de driedimensionale structuur van de beeldinhoud berekend worden of de beweging van de camera.

Dit proefschrift laat zien dat de optredende ruis in de analyse van stereo-paren niet Gaussisch is, dus dat in dit geval het gebruik van SSD niet optimaal is. Daarnaast worden er diverse methoden onderzocht om de werkelijke ruisdistributie te schatten en hierbij wordt gevonden dat de ééndimensionale aanpak een tweede fundamentele aanname heeft: het verschil tussen eigenschappen bevat voldoende informatie om de gelijkheidsdistributie te modelleren. Dit proefschrift toont aan dat dat niet zo is.

Op het gebied van computer vision is het probleem van het vinden van overeenkomsten tussen afbeeldingen van twee zichtpunten een zeer belangrijke en uitdagende geweest in de afgelopen twintig jaar. In dit proefschrift wordt beschreven dat de meerdimensionale aanpak de eerder genoemde tweede aanname overwint en significant betere resultaten geeft bij de meest geloofwaardige en gerespecteerde internationale test set in het vinden van overeenkomsten in stereo-paren. In het algemeen verbetert dit werk de theorie van computationale gelijkheid op een fun-

damentele manier, die mogelijk op alle gebieden van patroonherkenning en computer vision verbetering kan brengen.

De laatste hoofdstukken van dit proefschrift, 9 en 10, beschrijven onderzoeken die enigszins losstaan van de content-based image retrieval. Deze onderzoeken hebben te maken hebben met automatische video-analyse. De technieken die beschreven worden, zijn ontworpen voor videobeelden van stilstaande camera's, wat veelal neerkomt op beveiligingscamera's. Het doel van de technieken is om bewegende personen of object in beeld te onderscheiden en deze te volgen. Hierbij kan de gebruiker terugkoppeling geven aan de processen die objecten proberen te volgen, zodat bepaalde objecten altijd zichtbaar blijven als bewegend object, zelfs als ze lang stil staan. Ook is het mogelijk om bepaalde delen van het beeld juist aan te merken als achtergrond, waarmee het nooit als een bewegend object gezien zal worden.

Als laatste wordt er in dit proefschrift nog een programma beschreven dat voor zowel onderzoek als onderwijs gebruikt kan worden. RetrievalLab stelt de gebruiker in staat om complexe bewerkingen uit te voeren op databases van afbeeldingen, zonder dat daar iets voor geprogrammeerd hoeft te worden. Het geeft hiermee snel inzicht in de processen die schuil gaan achter content-based image retrieval en het detecteren van visuele concepten.

Acknowledgements

First, I would like to thank my parents, who, each in their own way, have given me so much valuable advice and shared their experiences with life, that I would not have been able to get to the point in life where I am now. Although we have our own way of looking at things, I have learned a lot from them, probably a lot more than they realize.

I also would like to thank Gerrie Oerlemans for making me more of a family guy and for putting my feet back on the ground sometimes. And thanks to Peter van der Salm for challenging me with questions and discussions to make me think more deeply about my own research. Wim and Lenie Faber have also given me a lot of support in the last few years.

Someone who has also given me very valuable advice and who was always able to set things in perspective, is Jan Kool. Although he is not around anymore to share in my happiness of receiving my degree, I am sure he has always known that I would be able to finish my thesis. He always pointed me in the right direction and told me about his own experiences to make all of my problems a lot easier to handle. He deserves a lot of credit for that.

Many thanks to Robin Hermann, who started out as a colleague when I started with my PhD program, but became a very valuable friend along the way. We had many discussions about every subject imaginable and I have learned a lot from him. He always had some good advice for me, no matter what the subject was. Maarten van der Heijden has also become a very good friend in these years and someone who I could always talk to when things were not going as planned.

At LIACS I have also had a lot of interesting discussions with Mark Huiskes and Bart Thomée about our research and we even travelled together to conferences a few times. I had a great time with you.

I would also like to thank my cousins Pieter van Maasdam, Bas van Goozen, Joost Kortekaas and Thijs van Maasdam for their support and mostly for keeping my mind off of things when needed. The sunday evening tradition has been an important part of my life and although I was not always able to be there, I will make it up to you guys.

My colleagues and friends at VDG Security also deserve a word of thanks, for

giving me the time I needed to finish research papers or my thesis and for the challenging discussions we had about all kinds of things. Also the not-so-challenging discussions have been great fun. Marijn Loomans, Timme Grijpink and all others, thanks.

A lot of other people have also been very supportive in the last few years, such as Jelle Bosma, Elise Zandstra, Casper Deurloo, Sander Eijkelhof and Mariëlle Linting.

And last but certainly not least, I wish to express my most deepest gratitude to my girlfriend Rolien. She has always supported me in attempting to reach the goals I had set for myself, even if that meant that I could not give her all the attention she deserves. I realize that I can be unpredictable and maybe even a little difficult to live with sometimes, especially when my mind is set to something, but I am sure that we will have a great future together. I could not have done this without you.

Curriculum Vitae

Ard Oerlemans graduated from Leiden University with a M.Sc. degree in 2004. During his studies, Ard also worked as a software engineer for VDG Security BV, a company that develops software for recording and analyzing surveillance video. He was also interested in image and video analysis so he started his PhD research at Leiden University in the domain of content based image and video retrieval.

Since then he has published his work in the leading venues in multimedia retrieval and also contributed to both teaching and research activities at LIACS. He believes in the synergistic interaction between academia and private industry. In particular, academia provides a fertile environment for doing frontier, next-generation research and private industry provides strong societal relevance. This thesis can be seen as both the culmination of his work and the beginning of the next stage of his career.