# Computer Vision for Timber Harvesting

**Dahl, Anders Bjorholm; Ersbøll, Bjarne Kjær; Aanæs, Henrik**

*Publication date:*
2009

*Document Version*
Publisher's PDF, also known as Version of record

# Computer Vision for Timber Harvesting

Anders Bjorholm Dahl

Kongens Lyngby 2009
IMM-PHD-2009-214

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

# Summary

The goal of this thesis is to investigate computer vision methods for timber harvesting operations. The background for developing computer vision for timber harvesting is to document origin of timber and to collect qualitative and quantitative parameters concerning the timber for efficient harvest planning. The investigations in this thesis is done as initial work on a planning and logistic system for timber harvesting called *logTracker*[1].

In this thesis we have focused on three methods for the logTracker project, which includes image segmentation, image classification, and image retrieval. Segmentation is to partition an image based on image characteristics and in our study we have focused on image texture. Our segmentation method is inspired by iterative function systems and contractive maps, which makes the basis for both our texture characterization and our method for obtaining the image segments. The purpose of image segmentation is to make the basis for more advanced computer vision methods like object recognition and classification. Our second method concerns image classification and we present a method where we classify small timber samples to tree species based on Active Appearance Models and texture characteristics. The last method is image retrieval based on the so called "bag of visual words" procedure. An image is characterized as a distribution of local image descriptors, which is the basis for effective image search.

These methods are described and discussed in relation to the logTracker project and ideas for further development of the system is provided. Building a complete logTracker system is a very demanding task and the conclusion is that it is

[1]The logTracker project is owned by the company Dralle A/S who has hosted this industrial PhD project.

important to focus on the elements that can bring most value to timber harvest planning. Besides contributing to the development of the logTracker system the described methods have a general applicability making them useful for many other computer vision problems.

# Resumé

Målet med denne afhandling er at undersøge computer vision metoder til brug i forbindelse med tømmerhugst. Baggrunden for at udvikle computer vision for tømmerhugst er dels at tilvejebringe dokumentation for træets oprindelse, og dels at indsamle kvalitative og kvantitative parametre om tømmeret til brug for effektiv hugstplanlægning. De undersøgelser, som er foretaget i forbindelse med denne afhandling, er indledende arbejde på et planlægnings- og logistiksystem til brug ved tømmerhugst kaldet *logTracker*[2].

Vi har i denne afhandling fokuseret på tre metoder, som har betydning for logTracker-projektet. Det drejer sig om billedsegmentering, billedklassifikation og objektgenkendelse. Segmentering er at opdele et billede ud fra billedkarakteristika, og vores fokus har været på billedtekstur. Vores segmenteringsmetode er inspireret af iterative funktionssystemer og kontraktive funktioner, som danner grundlag for teksturkarakteriseringen og vores metode til identifikationen af billedsegmenterne. Billedsegmentering har til formål at danne grundlag for mere avancerede computer vision metoder, som eksempelvis objektgenkendelse og klassifikation. Vores anden metode omhandler billedklassifikation, og vi præsenterer en metode til klassifikation tømmer som ved hjælp af *active appearance models* og teksturkarakteristika klassificeres til træart. Den sidste metode omhandler objektgenkendelse og er baseret på det såkaldte *bag of visual words procedure*. Her bliver et billede bliver karakteriseret som en fordeling af lokale billed-*descriptors*, hvilket danner grundlag for en effektiv metode til billedsøgning og objektgenkendelse.

Disse metoder er beskrevet og diskuteret i relation til logTracker-projektet, hvorved der skabes fokus på den videre udvikling af systemet. At bygge et

[2]logTracker-projektet ejes af Dralle A/S, som har været vært for dette Ph.d.-studie.

fuldstændigt logTracker-system vil være en meget krævende opgave, så konklusionen er, at indsatsen bør rettes mod de elementer, som kan bidrage med mest værdi til hugstplanlægningen. Ud over at bidrage til udviklingen af logTraker har de beskrevne metoder generel karakter, der gør dem anvendelige til løsning af mange computer vision problemer.

# Preface

This thesis was prepared at DTU Informatics, the Technical University of Denmark and at Dralle A/S, in partial fulfillment of the requirements for acquiring the degree of Doctor of Philosophy, PhD, in applied mathematics. The project was conducted as an Industrial PhD with Dralle A/S as host company who partly financed the project together with The Danish Agency for Science, Technology, and Innovation.

The thesis is based on research carried out in the PhD period and contains a public and a confidential part, and the public part is based on three scientific papers.

The project is carried out in close collaboration with Associate Professor Ali Shokoufandeh and PhD student Peter Bogunovich, both at Drexel University, Philadelphia, PA, USA.

The project was partly supervised by Associate Professor Bjarne Kjær Ersbøll and Associate Professor Henrik Aanæs, both at DTU Informatics, and partly by industrial supervisor PhD Mads Jeppe Tarp-Johansen at Dralle A/S.

Kgs. Lyngby, February 2009,

Anders Bjorholm Dahl

# Acknowledgements

This thesis is the result of a collaboration that has involved many people who deserve a great appreciation. With my background in forestry it has been a challenge to study image analysis, but also a very interesting and enlightening experience and I would like to acknowledge the people that believed in my project. First of all I would like to thank my advisors Henrik Aanæs, Mads Jeppe Tarp-Johansen, and Bjarne Kjær Ersbøll who have helped me stay in focus and get me safely through the process of this PhD study. I would also like to thank Kim Dralle for introducing me to this interesting field of research and making my studies possible by providing a company to facilitate this industrial PhD.

The person who deserves my deepest gratitude is Dr. Ali Shokoufandeh who hosted my foreign stay in the winter of 2007. His critical advice, guidance, and inspiration have been a tremendous help for my knowledge and understanding of the scientific discipline of image processing and computer vision, but also through his engagement and support on a personal level. My "buddy" Peter Bogunovich also deserves a great appreciation for his effort in the work of my PhD, for critical and constructive discussions, for being a great friend, and for hosting me in his home when I have visited Philadelphia. I also want to thank my first host in Philadelphia Jeff Abrahamson who is a great friend and who was a great support when I first came to the USA. Walt Mankowski also deserves recognition for his help in improving my English and for being a good friend. I would also like to thank the other people that I have met and made friends with at my foreign stay, and who have made my several visits in Philadelphia enjoyable.

Also many thanks to all my other colleagues, both present and former at DTU Informatics and at Dralle A/S, who have made it great fun to perform this PhD study. I also want to thank Eina Boeck who has been helpful with all practical issues and ready to answer any question and to Tove Jacobsen for keeping track of my budget.

Finally, I would like to than my fiancé Lotte and our daughter Rigmor Emilia for being supportive and understating throughout the PhD, but especially in the thesis writing period.

# Contents

Chapter 1

# Introduction

This thesis aims at computer vision methods for improving the effectiveness and quality of timber harvesting operations. The studies that we will present in this thesis are very preliminary in relation to an operational computer vision system for timber harvesting, but the development of such a system has not been the goal. Instead, we have investigated different elements and identified relevant problems of computer vision in relation to timber harvesting in particular, but also in relation to more general computer vision problems for example the problem of classification and image segmentation. This thesis shows that computer vision is a valuable tool for solving important problems in the timber harvesting process. These problems concern tracking the wood for documentation of origin and obtaining information about volume and quality in an efficient manner. Furthermore, the methods developed in the thesis has general applicability for problems other than timber harvesting, and hopefully this can act as a small contribution to the research within the field of computer vision. First we will discuss the motivation of the thesis concerning potential forestry applications, then we will give a short outline of the basic elements of computer vision, and finally we will give a outline of the contributions of this thesis.

## 1.1 Project background

The goal of this thesis is to investigate computer vision methods for visual tracking of wood logs, and for obtaining information about the logs concerning volume and quality. The desired application is the *logTracker system*[1], which is a logistic system to be used in the process of timber harvesting, with the purpose of tracking wood logs in the production chain from the harvesting in the forest to the logs enter a wood processing industry. The idea is to equip each machine that handles the wood with cameras to acquire images of each individual log, and also a GPS for positioning where the images were acquired. These images should be analyzed to obtain log specific information about wood quality and quantity, but also to get a log "fingerprint", which can be used for recognition. By recognizing the log each time it is handled and coupling it with a GPS position, it will be possible to document the origin of the wood. The motivating background for the logTracker system is politically founded in an increasing demand for the sustainable timber production methods. The environmental issues regarding forest management were made a public concern in the 1980's especially in relation to deforestation of tropical rainforests and illegal logging of tropical hardwoods. This concern enhanced the need for systems to ensure good management practices, and certification was seen as a solution. Certification was introduced as methods to benefit all stakeholders in the forest production. The lumber produces would gain the benefit of higher timber prices from a market willing to pay for sustainably produced timber, and the consumers would benefit from the option of purhacing products from legally logged forests. Different certification systems like the FSC[2], PEFC[3], SFI[4], etc., was introduced, and are currently used for marketing products from sustainable managed forests, and this sustainability covers environmental and socioeconomic aspects. Despite the initial concern towards tropical forests was soon extended to the rest of the world [52]. In certified forests the land owners have to fulfill a number of rules and regulations in their production methods to meet these political goals. Today, organizations like the FSC enforce this through consulting and inspection, but a weak link in the system is the transport of wood from the forest to the industry, the so called *chain of custody* [43, 52, 93].

Different labeling technologies have been employed to document the chain of custody. The traditional way is to paint or engrave the label into the log end face, which is a simple way to mark individual logs, but the amount of information is limited and it is in risk of fraud. Traditional barcode labels are also used for tracking wood logs, where the information can be referenced in

[1] http://www.dralle.dk
[2] http://www.fsc.org/
[3] http://www.pefc.org/internet/html/
[4] http://www.sfiprogram.org/

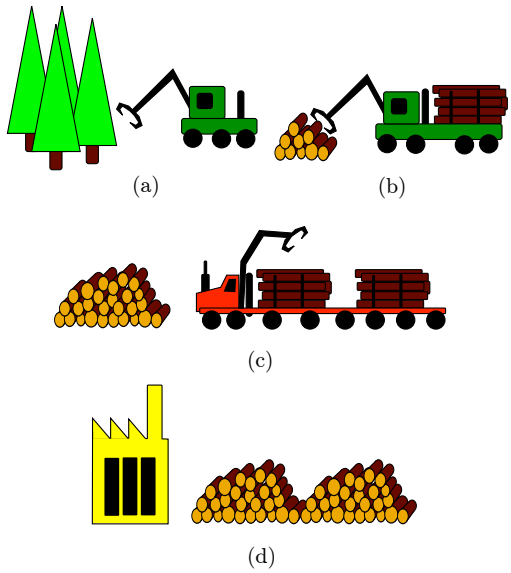



Figure 1.1: Wood harvesting operation. Figure (a) illustrates the harvester, (b) is the forwarder, (c) is the road side truck, and (d) is the industry.

a database or directly encoded in the barcode. In general the two-dimensional barcodes can encode sufficient information. The most promising technology is the Radio-Frequency Identification (RFID) labels, which can both receive and send data without need of physical contact. RFID can store large amounts of data concerning the wood log, and the data can be stored and accessed in a secure manner. The primary problem of this method is the price of the RFID which is still too costly to implement as a standard for wood tracking [43]. The logTracker system is potentially an alternative solution to these problems, where the idea is to obtain a wood log "fingerprint" from digital images of the wood logs, which can be used for documenting the chain of custody problem.

The timber transport chain is illustrated in Figure 1.1. The chain begins with a harvesting operation (a) where the trees are cut down and debranched. The next step is a forwarding operation where the individual logs are collected from the forest floor and transported to the road side (b). At the road side a truck picks up the wood logs (c) for transportation to the timber industry (d). All the machines handling the wood can be equipped with cameras to gather information about the wood, both in relation to documentation and logistical issues. This will enable an objective system for documenting the origin of the wood and through this increase the reliability of a labeling system.

The support of decisions in relation to logistic issues is another important element, and solving the computer vision problems related to the logTracker system is a demanding task and far from accomplished. This thesis contains preliminary studies in methods related to logTracker, which addresses some fundamental computer vision problems.

## 1.2 Computer vision

The human vision is an important sense that enables us to identify and relate objects, and to react to what wee see without any physical contact. Transferring these abilities to computers is the fundamental task of computer vision, which has a wide variety of useful applications. It can for example be used to help navigate cars in crowded traffic, guide a precise automated harvesting of crops, or improve the quality of timber harvesting, which is the goal of this thesis. Computer vision is to provide machines with the benefit of vision making them capable of acting and reacting to the surrounding environment on the basis of visual input.

A computer vision system typically consists of an image acquisition step, a pre-processing step, a feature extraction step where the elements of interest are detected, and a high level vision step. The implementation of these steps is dependent on the task to be solved. Computer vision is based on a range of image types including gray level images with two spatial dimensions and one intensity dimension. Gray level images can be acquired from light sensitive sensors capturing emitted or reflected light, for example in the visible or infrared spectrum, where the intensity value reflects the received amount of light. Other gray level images can be X-rays or ultrasound images where the image intensities reflect absorption of X-rays or reflectance of ultrasound waves. Color images typically include three intensity dimensions and the normal representation is the red, green, and blue color band containing information about the color of the received light. Multispectral images can contain a number of spectral bands where each band normally covers a limited range of wavelengths. 3D information can be obtained from range sensors, which consists of a depth map in addition to a gray level image. This depth map just covers the surface of the depicted objects, whereas 3D information from for example MRI or CT scanners give voxel based information making all the 3D information available.

Some computer vision procedures require a preprocessing of the acquired images. Images can be resampled to account for lens distortion, which is necessary in for example structure from motion systems where multiple camera views are used for reconstructing the 3D surface of a scene [66]. Other preprocessing

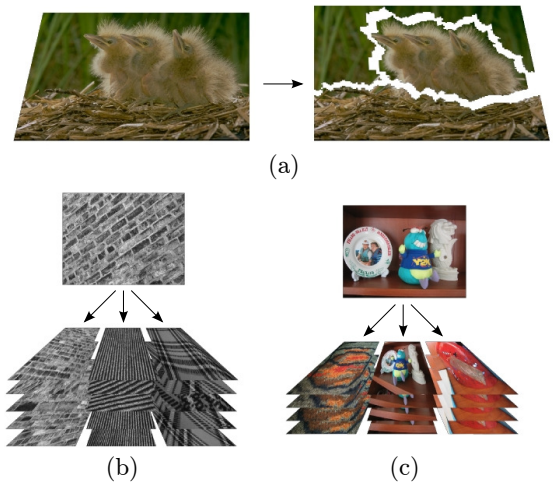



Figure 1.2: Segmentation, classification, and image retrieval in computer vision. Figure (a) shows an example of image segmentation where the image is decomposed into regions. Figure (b) illustrates the problem of classification where the goal is to find a specific image class, in this case matching the image at the top to one of the three classes below. In Figure (c) the problem of image retrieval is illustrated. Here the goal is to find the actual depicted scene.

steps can be noise reduction, if for example the images are acquired in low light conditions, or enhancing the contrast of the image. Building a scale space is another element that can be computed as a preprocessing step. This is done to enable invariance to scale change in the feature extraction step.

Image features are the characteristic image elements that make up the basis for obtaining higher level information. Features include lines, edges, corners, regions, etc., which contain information about for example object boundaries and shape. Texture is another important feature for object characterization. The image features are useful for image segmentation, for detecting regions of interest, or for finding interest points for object characterization, etc. Object recognition and classification are typical high level tasks, which are based on the feature characterization. A large amount of research effort is put into high level vision tasks and there is a wide variety of methods. Typically, these methods are developed to solve specific vision problems, but the research evolves in the direction of generalizing these procedures to solve more demanding problems. Examples of high level vision tasks are object recognition via model verification, feature based image scene classification, and tracking and pose estimation of people useful for surveillance. These examples are just a few of the many useful

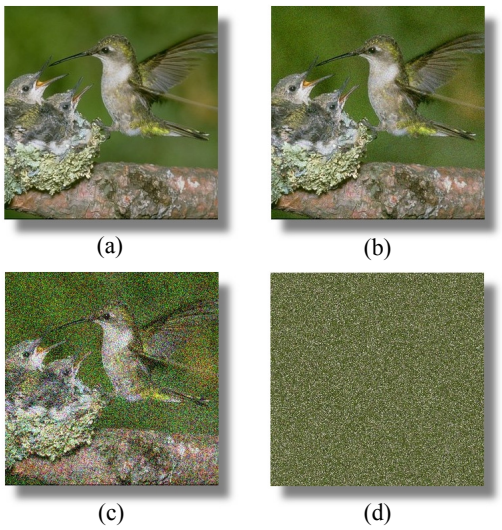



Figure 1.3: Image change due to noise. Figure (a) is the original image, Figure (b) is added Gaussian noise with a standard deviation of 4 % of the image range. Figure (c) has added 20 % noise and Figure (d) shows a random permutation of the image, but retaining the original pixel intensities. It remains quite easy to see what is depicted in the image, even with relatively high amounts of noise (Figure (b) and (c)), whereas the original image values in random order contains very little usable information.

applications of high level vision systems.

Solving computer vision problems is a demanding task because the spatial resolution of an image is a limiting element for any computer vision system. This implies a limit to the level of detail at which it is possible to analyze the image and obtain information. There is also a limit to the temporal resolution, because time is a limiting factor in most computer vision applications. Often, the limited temporal resolution result in low quality image data that are blurred and contain noise. These limitations in the spatial and temporal resolution make the image processing elements of computer vision hard, both concerning the low and high level vision problems. In this thesis we focus on 2D image processing, and we have not investigated the initial steps of computer vision concerning image acquisition or preprocessing. Our work has concerned the low level computer vision tasks of feature extraction and segmentation, where we have suggested a solution for identifying homogenous regions in highly textured and diverse images. We have also worked with the higher level tasks of classification and image retrieval, where we have made a system for wood species identification

and a image retrieval application. The basic elements of these computer vision problems are illustrated in Figure 1.2.

2D digital images, as the investigations in this thesis build on, are characterized by pixels, i.e. a set of numbers ordered in a grid structure, but far from all grid-ordered sets of numbers will make much sense as an image, for example an image of random numbers. On the other hand images of scenes familiar to the viewer contain much information and even different or noisy versions of the same scene type might be easy to recognize. But in the pixel domain these images can be very different, which illustrates a fundamental problem of computer vision, see Figure 1.3. An important observation is that images are not random or chaotic, but highly structured elements with much fewer degrees of freedom than the pixel distribution might suggest. This structure will often be reflected in that a few parameters can control highly complicated image structures. An example is the iso-metric feature embedding of Tenenbaum *et al.* [141] who reduces the parameter characterizing an image of a head from 4096 dimensions of $64 \times 64$ input images to two dimensions reflecting the two directions of head pose change. In computer vision we are faced with the problem of identifying unique visual characteristics that bridge the perception of what we see to the pixel-representation of images.

The characteristics of digital images make computer vision non-trivial because small changes of visual appearance can dramatically change the pixel representation. This is especially relevant considering the motivation for this thesis, which concerns computer vision applications for outdoor forest scenes. Under these conditions computer vision is very demanding, because the methods have to be robust to large changes in object appearance due to change in light, viewing angle, scale, and also physical change of the scene elements. We will begin by explaining the practical background of the thesis.

## 1.3 Layout of the thesis

This thesis contains a public and a confidential part. The public part covers the three main topics, i.e. image segmentation, image classification, and image retrieval, which is covered in one chapter each. The confidential part contains contributions with particular value for the host company.

The three topics of the thesis are related, and they have many elements in common. A theoretic background for some of these topics is discussed in Chapter 2. Chapter 3 describes image segmentation based on contractive functions and IFS, which is related to fractal image compression, see for example [53, 76].

Classification is described in chapter 4 and an approach based on an *Active Appearance Model (AAM)* [31] and second order textural statistics [64, 25] is shown. Finally, image retrieval is presented in Chapter 5 and an example is given building on dimensionality reduced SIFT features [95]. The public part of the thesis is concluded in Chapter 6. The contribution of this thesis is described in three papers listed below. The segmentation method in Chapter 3 is based on our paper in [35], but the method has been modified, so to give a complete description of the method this Chapter is written independently of the paper. In Chapters 4 and 5 the original papers are included with only minor clarifying changes. The topic of all the chapters are related to the problem of the thesis and the practical use of these methods is discussed.

Publications from this thesis:

A. B. Dahl, H. Aanæs, R. Larsen, and B. K. Ersbøll. Classification of Biological Objects using Active Appearance Modeling and Color Co-Occurrence Matrices. Scandinavian Conference on Image Analysis, Aalborg 2007, Denmark.

A. B. Dahl and H. Aanæs. Effective Image Database Search via Dimensionality Reduction. IEEE Conference on Computer Vision and Pattern Recognition, Anchorage 2008, Alaska, USA.

A. B. Dahl, P. Bogunovich, A. Shokoufandeh, and H. Aanæs. Texture Segmentation from Context and Contractive Maps. Submitted to IEEE Conference on Computer Vision and Pattern Recognition, Miami 2009, Florida, USA.

Chapter 2

# Theoretic background

Generally scientific concepts and terms are dependent on the context in which they are used, and the goal of this chapter is to introduce and discuss some basic theoretic concepts related to the methods introduced in the later chapters. We will start by discussing image characteristics and some general approaches for image processing. Following this general discussion we will introduce the theoretic background of Iterative Function Systems (IFS), which is a central element of the segmentation method that is presented in Chapter 3. Both our segmentation procedure in Chapter 3 and the classification procedure in Chapter 4 make use of image texture characteristics. Therefore, we will end this chapter by discussing the basic concepts and characteristics of image texture.

## 2.1 Image processing

The objective of image processing is to obtain information from digital images, which is highly dependent on the tasks to be solved. Some computer vision systems are constructed to make the image processing tasks very easy to solve. This is done by constructing the image acquisition step to give high constrast images and the image processing can be performed by simple classification methods. This is typical for some industrial or medical applications. For many other


(a) (b) (c)

Figure 2.1: Visual perception. Figure (a) and (b) depicts the American president Barack Obama with his mouth turned upside down. With the entire image turned upside down it appears reasonably normal, but with the image in the right position, this mouth misplacement becomes very apparent. In Figure (c) a poster from the American presidential campaign is shown. It is quite easy to see that it is a picture of Barack Obama despite the difference from his natural appearance in both color and texture.

applications it is not possible to control the environment for image acquisition making the image processing harder. Computer vision has long been a very active field of research and a wide range of methods have been developed based on mathematical and statistical tools. Despite remarkable results and solutions to many demanding problems, there are still many open problems and the capabilities of computer vision systems are inferior the human vision.

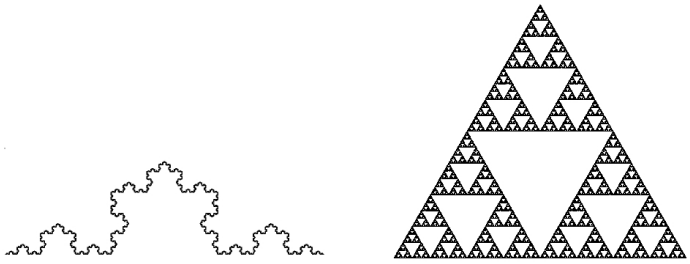
**Image information** Many physical properties of vision are well understood and examples include the anatomy of the eye and which wavelengths are visible [50, 51]. But there is only little understanding of the visual perception, i.e. how the visual stimuli of the receptor cells are interpreted by the brain. Modern digital cameras are good at accurately capturing the visual input, but the task is to couple this visual input to perception. Because of the lacking knowledge of how we transform what we see to what we understand, it is impossible to build a computer model that exactly copies the human vision. Despite this little understanding there are some characteristics that perhaps can act as a guide for the direction of artificially modeling vision. People usually find it hard to precisely reconstruct what we have seen, for example making an exact drawing of an object. This indicates that we do not remember visual input as it was received from our eyes. On the other hand we are very good at generalizing and we can easily recognize who is depicted in a cartoon even though the person

might be highly changed and simplified. Our visual perception can also be confused if we see something different from what we expect. This indicates that we somehow have an expected model of objects, especially for familiar things like peoples faces. Some examples are given in Figure 2.1. The question is how the visual perception of humans can be used in developing computer vision methods.

There are many ways to obtain information about the content of an image. One way to approach this is based on the idea that objects and scenes have a unique visual appearance which can be used for retrieval. Given a very large collection of labeled images the goal is to find the most similar image and transfer the labels to the unknown image. An example of this approach is shown in Torralba *et al.* [142] with high recognition rates and capacities of more than 10 million images. Another related example is the Video Google of Sivic and Zisserman [133], where they built an image representation as an independent set of appearance features. Utilizing techniques from text retrieval enables them to do very fast object recognition on a large set of data. These appearance based image processing models clearly differ from human vision in the capability of generalization. It would be problematic for example to recognize the similarity between Figure 2.1 (b) and (c). A way to get around this would be to have both examples labeled as the American president, but this does not capture the actual similar features.

Another group of computer vision methods is related to the geometric information in objects. If we describe for example a tree, we would say that it has a trunk with branches attached to it and leaves attached to the branches. This kind of geometric information and rules about object relations can be used in a computer vision system. One way to model objects is as a set of geometric primitives and an example is Dickinson *et al.* [40], where they match 3D primitives, including cylinders, cones, blocks, etc., to the image and use the geometric configuration to find the similarity to model objects. Modeling objects this way is related to how we describe objects, but generalizing the approach is problematic because it requires a geometric description of each object type in the data set. Furthermore, it can be hard to fit the geometric primitives to the visual information in images. This can be the explanation of why the geometric based methods lost their popularity to appearance based methods from the late 1990's, see for example [127].

In Chapter 4 we investigate object classification based on an Active Appearance Model (AAM) which is an object specific model related to modeling geometric elements of objects, and in Chapter 5 we do image retrieval based on appearance features. Both these methods have elements of high level vision, whereas the segmentation procedure that we describe in Chapter 3 is a low level task, that provides minor information about the depicted objects, but can act as a very



(a) (b)

Figure 2.2: Examples of fractals, i.e. the fixed point of an iterated function system. The Koch curve (a) and the Sierpinski gasket (b).

important step in higher level tasks. Low level methods like this segmentation procedure is a way of combining ideas from both the geometric and appearance based methods for improving existing computer vision methods.

## 2.2 Iterative Function Systems

The segmentation procedure described in Chapter 3 builds on Iterative Function Systems (IFS) and we will in this section describe some of the important mathematical elements for developing this procedure.

**IFS and contractive maps** An observation made by Mandelbrot [99] was that many natural objects resembles properties related to fractal sets. He discusses the problem of determining the length of certain curves, and it turns out that the length greatly depends on the measuring unit used. The measuring unit should be seen as the minimum distance between two points on the curve, so everything smaller than the measuring unit is neglected. When the length of measuring unit approaches zero the curve length approaches infinity. This is because such curves have infinitely many small "turns" or features and they can only be measured if the length of the measuring unit is smaller than feature itself. An example is the Koch curve [115], see Figure 2.2. Any two points on this curve have an infinite distance between them. Coast lines are examples of natural objects that have nearly the same property. Mandelbrot claimed that these natural curves exhibit a property known as statistical self-similarity, i.e. that each portion of the curve is a scaled down version of the curve.

We will now explain some of the details of the IFS theory, but a more elaborate

discussion of fractals can be seen in for example [9, 47, 115]. We begin this discussion by some definitions

**Definition 2.1 (Metric space)** A set $X$ together with a function $d : X \times X \rightarrow \mathbb{R}$ is called a *metric space* [11] and is denoted by $(X, d)$ iff $d$ has the following properties

1. $d(x, y) \geq 0,\ \forall x, y \in X,$
2. $d(x, y) = 0 \iff x = y,$
3. $d(x, y) = d(y, x),\ \forall x, y \in X,$
4. $d(x, y) \leq d(x, z) + d(z, y),\ \forall x, y, z \in X.$

**Definition 2.2 (Cauchy Sequence)** Given a metric space $(X, d)$, a sequence $Y = (y_1, y_2, \dots)$ with $y_i \in X$ is called a *Cauchy Sequence* [123] if for any $\eta > 0$ there exists $N \in \mathbb{N}$ such that for all $n, m \geq N$ we have $d(y_n, y_m) < \eta$.

**Definition 2.3 (Complete Metric Space)** A metric space $(X, d)$ is called *complete* if all Cauchy Sequences in $X$ converge to a point in $X$ [123].

**Definition 2.4 (Hausdorff Space)** For the complete metric space $(X, d)$ the so called *Hausdorff Space* $\mathcal{H}(X)$ denotes the set of all non-empty compact subsets of $X$.

**Definition 2.5 (Hausdorff Metric)** Given a complete metric space $(X, d)$ with $x \in X$ and $B \in \mathcal{H}(X)$, let $d(x, B) = \inf\{d(x, y) : y \in B\}$; i.e. the distance to $x$ from the point $y \in B$ which is closest to $x$. If $A \in \mathcal{H}(X)$ then let $d(A, B) = \sup\{d(x, B) : x \in A\}$; i.e. the distance to set $B$ from the point $x \in A$ which is furthest from $B$. The *Hausdorff metric* $h : \mathcal{H}(X) \times \mathcal{H}(X) \rightarrow \mathbb{R}$ [9] is defined as $h(A, B) = \max\{d(A, B), d(B, A)\}$.

**Definition 2.6 (Attractor Set)** Given an IFS, $\{X, \mathbf{w}\}$ and a set $S \subset X$, let $w_i(S) = \{w_i(x) : x \in S\}$ and denote the set operator associated with $\mathbf{w}$ as

$$F_{\mathbf{w}}(S) = \cup_{i=1}^{n} w_i(S). \tag{2.1}$$

An *attractor set* or sometimes called a *fractal set* [10] is defined as a set $A \subset X$ with the property that $F_{\mathbf{w}}(A) = A$. In other words, an attractor is a set that is invariant to $F_{\mathbf{w}}$.

IFS can be defined based on contractive transformations.

**Definition 2.7 (Contractive Transformation)** Given a metric space, $(X, d)$, a transformation, $f : X \to X$, is called *contractive* or a *contraction* with *contractivity factor* $s$ [47] if the exists a constant $s \in [0, 1)$ so that

$$d(f(x), f(y)) \leq s \cdot d(x, y) \qquad \forall x, y \in X. \tag{2.2}$$

**Definition 2.8 (Iterated Function System)** A family of contractions $\mathbf{w} = \{w_1, w_2, \ldots, w_n\}$ with respective contractivity factors $\{s_1, s_2, \ldots, s_n\}$ in a complete metric space $(X, d)$ is called an *iterated function system* or *IFS* [47]. Such a family of contractions is also sometimes also called a *hyperbolic iterated function system* [10].

Applying an IFS to a point $x \in X$ will converge to a fixed point. A fixed point is defined as

**Definition 2.9 (Fixed Point)** Given a transformation $f : X \to X$, then a point $x^* \in X$ where $f(x^*) = x^*$ is called a fixed point of $f$ [9].

Contraction is the property that guarantees that the IFS applied to any point $x \in X$ will become a fixed point $x^*$. The following theorem is very important in this regard.

**Theorem 2.10 (Contractive Mapping Fixed Point Theorem)** *Let $(X, d)$ be a complete metric space, with points $x \in X$ and distance metric $d$. If $f : X \to X$ is a contractive mapping on $X$, then there exists a unique point $x_f \in X$ such that for any point $x \in X$*

$$x_f = f(x_f) = \lim_{n \to \infty} f^{\circ n}(x), \tag{2.3}$$

*$x_f$ is called a* fixed point *or* attractor *of $f$.*

$\lim_{n\to\infty} f^{\circ n}(x)$ is limit of reapplying the function $f$ $(f(f(...f(x))))$ infinitely many times. See for example [53, 107] for a proof of the theorem. It has been shown that given an IFS, i.e. is a set of functions, there is one unique attractor [107], and the attractor is obtained by iteratively applying the set of contractor functions. The inverse problem of IFS is to find the set of contractor functions that will result in an attractor set that is close to a given set. Formally the inverse problem of IFS is stated as

> Let $(X, h)$ be a complete metric space with $A \in \mathcal{H}(X)$ and $\eta > 0$. Does a contractive function $f : \mathcal{H}(X) \to \mathcal{H}(X)$ exists such that $h(A, A_f) < \eta$?

where $A_f$ is the attractor of $f$ and $h(\cdot)$ is the Hausdorff metric. It can be hard to find the set of contractive functions with an attractor fulfilling $h(A, A_f) < \eta$. The collage theorem helps in solving that problem

**Theorem 2.11 (Collage Theorem)** *Given an IFS:* $\{X, \mathbf{f}\}$, $f_i \in \mathbf{f}, i = \{1, ..., n\}$ *in the complete metric space* $(X, d)$ *with the contractive factor* $0 \leq c < 1$. *Let* $\eta > 0$ *and suppose that there exists an* $x \in \mathcal{H}(X)$ *we obtain*

$$d(x, x_f) \leq \frac{1}{1-c} d(x, f(x)), \tag{2.4}$$

*where* $x_f$ *is the fixed point of the function* $f$.

Proof. We use the triangle inequality to obtain

$$d(x, x_f) \leq d(x, f(x)) + d(f(x), x_f) \tag{2.5}$$

We know that the fixed point maps onto itself: $f(x_f) = x_f$, so we get

$$\begin{aligned} d(x, f(x)) + d(f(x), x_f) &= d(x, f(x)) + d(f(x), f(x_f)) && (2.6) \\ &\leq d(x, f(x)) + c\, d(x, x_f). && (2.7) \end{aligned}$$

From this we obtain

$$d(x, x_f) \leq \frac{1}{1-c} d(x, f(x)). \tag{2.8}$$

The collage theorem gives a limit to the difference between a given point $x$ and it's fixed point $x_f$ with the contractive map $f$. This is very useful because we get an indication of how close we are to the attractor of a give function. This leads to the notion of collage error, which can be used to find a good solution

of the inverse problem. Let $\epsilon$ denote the collage error, then we can estimate the collage error of a point $x$ and the contractive function $f$ as

$$\epsilon = d(x, f(x)). \tag{2.9}$$

If we obtain a low collage error for a given $f$ and $x$, we know that $x$ is close to the fixed point of $f$. This is the important observation enabling us to handle the inverse problem. The inverse problem can be reformulated as

> Let $(X, h)$ be a complete metric space with $A \in \mathcal{H}(X)$ and $\eta > 0$. Does a contractive function $f : \mathcal{H}(X) \to \mathcal{H}(X)$ exists such that the collage error $h(A, f(A)) < \eta$?

Finding a map that has a small collage error is computationally less expensive than finding maps that has a close fixed point. Using the collage error for identifying maps with fixed points close to a given point is not guaranteed to be optimal, but the result will have an upper bound given by equation (2.4), see for example [4] for further discussion.

The Sierpinski gasket and the Koch curve shown in Figure 2.2 are examples of fractals obtained from IFS. Fractals can be generated very easily using an algorithm based on the so called chaos game. Given the IFS consisting of the functions $f_i$, $i = \{1, ..., n\}$ in the metric space $(X, d)$, we can find the fractal using Algorithm 1 shown below. Note that there is a probability associated with each function in the chaos game algorithm. Another way of constructing a fractal is to pick a set $C \in X$ and iteratively apply each map to this set until convergence.

**Algorithm 1** Chaos game

```
Pick an initial point x_0 ∈ X
Choose number of iteration M
for m = 1 to M do
    pick a random i with probability p_i, i = {1, ..., n}
    x_m ← f_i(x_{m−1})
end for
```

It can seem quite surprising that a very ordered image is obtained by having a set of functions and randomly selecting and reapplying one of these functions. This is caused by the contractive nature of the IFS, which is a very useful property and the key element for fractal image compression, which we will discuss in Chapter 3.

## 2.3 Texture characterization

Image texture is an important element in our segmentation procedure described in Chapter 3 and our classification procedure in Chapter 4. A good texture characterization is important for both applications and we will give an introduction to some aspects of texture. Texture has been a very active field of research and many computer vision applications are based on textural image information. It is not the goal of this section to give a complete overview of ways to characterize texture, but merely to give some examples of how texture can be described. We will focus on explaining texture characterizations that have been used or have inspired the development of our methods.

There is no clear definition of visual texture, but some features are important for characterizing it. In Figure 2.3 we show some examples of natural textures with varying degree of homogeneity. Some authors have grouped textures in relation to their homogeneity from periodic to random patterns [73, 120]. Haralick [63] describes texture in relation to pixel intensity and a spatial organization of texture primitives, which is the motivation for his texture features. Texture primitives are also the observation that Julesz [79] uses for characterizing texture in the form of textons. Additional features like smoothness, roughness, orientation, regularity, repetition, etc. are also important for texture description [17, 71, 79, 103]. Texture characterization is only useful if there is some degree of structure and repetition in the patterns. So, the periodic element of many textures is very important in texture characterization and has been the basis for some recent computer vision applications, see for example [8, 69, 71, 157]. The frequency of these periodic elements relates to the scale of the texture, which is also an important issue, because it indicates at which scale descriptors should be calculated, but also because it is an important feature itself [71, 75, 98].

### 2.3.1 Texture descriptors

Identifying the characteristic elements of texture has been done in multiple ways, and we will now describe some of these characterizations and discuss how they have be used.

**Statistical methods** Statistical methods for texture characterization are typically grouped as first, second, and higher order statistics. First order statistics concern the distribution of pixel intensities, second order concerns the joint distribution of intensities, and higher order concerns the joint distribution of more than two pixels [17, 25, 103]. We will give a short introduction to these texture

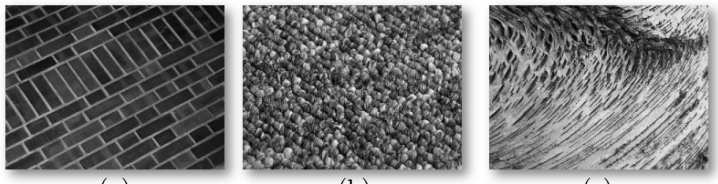



Figure 2.3: Examples of natural textures. Figure (a) shows a brick wall where each brick is a relatively regular pattern repeated over the image. Figure (b) is a carpet with varying intensity in the repetitive patterns, but with relatively uniform intensity and scale of the patterns. Figure (c) is a bark texture with large variation in intensity, orientation, and scale of the image patterns. Images are from the UIUCTex database [89].

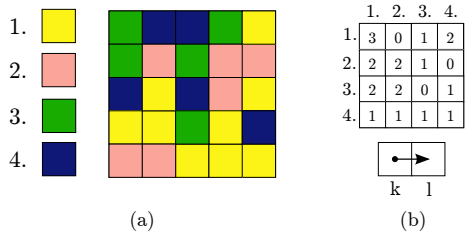



Figure 2.4: Gray level co-occurrence matrix (GLCM). Figure (a) illustrates a simple image with four intensity values. The associated GLCM is shown in Figure (b) for the displacement vector $\mathbf{h} = (0, 1)$ shown in the bottom part of the figure. To obtain the actual probabilities of the GLCM, it is normalized with the total sum, which is 20 in this example.

statistics and their associated features. The pixel distribution or the gray level histogram can be used for estimating features as the mean gray level, variance, skewness, entropy, etc. [25]. Second order statistics can be calculated as Gray Level Co-Occurrence Matrices (GLCM) introduced by Haralick *et al.* [64], and higher order statistics include for example gray level run length matrices and neighboring gray level dependence matrix [25].

We have used GLCM calculated from color images in our work on tree species classification presented in Chapter 4. We will now give a short discussion of the GLCM and how descriptors can be obtained. The GCLM expresses the probability of a given intensity change between two relative positions in the image. Let $\mathbf{c}$ be a GLCM with a displacement $\mathbf{h}$ and the intensity elements

$(k, l)$. Then the GLCM is defined as $\mathbf{c} = P((k, l)|\mathbf{h})$. An example of a GLCM is shown in Figure 2.4. It is possible to calculate the GLCM from an infinite number of displacements, so to make this manageable the typical choice is to use a fixed number of angles and radii. Even with a limited number of displacements, for example 8 angles and 3 radii, the result will be 24 GLCM for one image. To reduce this number and obtain invariance to rotation the GLCMs are averaged for the rotations resulting in an isotropic GLCM. It should be noted that this averaging removes information about the relative GLCM orientation which turns out to be an important characteristics in our experiment, see Section 4.2.5.

Palm [112] investigates the use of color in co-occurrence matrices, both as single channel (SCM) and multi-channel (MCM). In the single channel approach the isotropic GLCM is calculated in each color channel resulting in independent texture statistics for the color bands. In the multi-channel approach this is extended to include co-occurrence between texture channels, i.e. $P((k, l)|\mathbf{h})$ for $k \in K$ and $l \in L$ where $K$ and $L$ are color bands and $K \neq L$. Palm uses different combinations of the SCM and MCM together with a subset of the Haralick features [64] to form a multidimensional feature vector. He obtains the best classification results by combining both SCM and MCM in the LUV color space. In our paper we use the combination of SCM and MCM, but only for the RGB color space, see Section 4.2.

Another example is the *Local Binary Patterns* (LBP) of Ojala *et al.* [140]. The texture descriptor is obtained by measuring the intensity difference between a point and it's neighbors. If this difference is negative the descriptor is assigned the value 0 and if it is positive or equal to zero it is assigned 1. This results in a binary string with a size given by the number of neighbors that it is compared to. The obtained string is made invariant to rotation by grouping all rotated versions of the same pattern. Furthermore, Ojala *et al.* observed that only some patterns were occurring frequently enough to give reliable information, so the final characterization is a distribution of what they call "uniform" binary patterns. The LBS are fast to estimate and they obtain good classification rates. Singular value decomposition has also been used for statistical texture characterization [96], where the largest normalized singular values are used as a feature vector for texture description.

**Fractal dimension** Fractal dimension comes from fractal geometry and is a measure of how much of the space is occupied around a point on a fractal, which is typically measured as the Hausdorff dimension [47]. This is typically approximated by the box counting dimension [153]. Varma and Grag [149] observe that the sum of intensities within a disk follows a power law as a function of the radius of the disk. Formally this is $\log \mu = D \log r + L$ where $\mu$ is the

pixel sum within a disk of radius $r$. $D$ is the fractal dimension, also known as the Hölder exponent, and $L$ is the intercept, and these measures turns out to be independent of the texture scale. They measure the distribution of $D$ and $L$, and the obtained distributions are good at characterizing texture, which is demonstrated with impressive classification results.

**Wavelet features** Wavelets have successfully been used for texture characterization. The principle of the discrete wavelet transform is to represent an image as a set of multiscale wavelet features. This is obtained from applying a vertical and horizontal convolution and a down sampling, where high and low frequency signals are separated into combined high and low frequency elements, a high frequency element, and a low frequency element. The low frequency element can then be further convolved creating the multiscale wavelet encoding [103]. Different basis functions are used for wavelet transforms, for example the Haar basis or Gabor filters. Gabor filters are combined Gaussian and harmonic oscillating functions, which have also been used for texture characterization [54]. The wavelet transform is used for building texture descriptors based on the wavelet coefficients. Examples are the channel energy [103], co-occurrence features [5], and the wavelet coefficients directly [48].

**Texture scale** Texture is only characteristic within a certain scale range. If viewed from a far enough distance a textured object will seem uniform, and as you get closer the texture will appear, and zooming very close to the texture it will disappear again, because the view would be below the scale of the texture. An example is a brick wall and viewed from far away it will seem uniform, but at some distance individual bricks will appear as a texture. A very close view will only reveal a part of one brick, and the view will be below the scale of the texture. Malik *et al.* [98] found the characterization of texture scale to be a fundamental issue for texture analysis and many texture descriptors will change if scale changes. Scale is also an important factor in itself, which is shown by Hong *et al.* [71] who characterizes texture by local scale estimation which they use for texture segmentation. Their texture scale model is based on local intensity distribution and includes dissimilarity in pixel distribution, estimated as the Wasserstein or Kullback-Leibler divergence, and the entropy of the pixel distribution. Estimating the model parameters is done by comparing a small image region to neighboring regions. The texture scale is found by optimizing a function that favors small patch size, high pixel distribution similarity, and high entropy (uniform distribution). Huang *et al.* [75] also use the texture scale for segmentation. They identify the minimum scale by identifying the size of image textons based on the Bhattachayya distance. These experiments demonstrate the importance of scale.

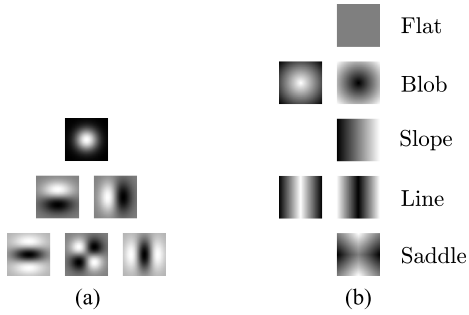



Figure 2.5: Gaussian derivatives (a) and basic image features (b). The top image in (a) is the zeroth order Gaussian derivative, the next two are the first order derivative, and the last three are the second order derivatives. The seven basic image features shown in (b) are obtained by convolving with these Gaussian derivatives.

**Algorithm 2** Basic image features

Measure filter responses: $\rho_{ij}$
Compute:
$\lambda = \sigma^2(\rho_{20} + \rho_{02})$
$\gamma = \sigma^2\sqrt{(\rho_{20} - \rho_{02}) + 4\rho_{11}^2}$
Classify according to the largest of the following features:
Flat: $\eta\rho_{00}$
Slope: $2\sigma\sqrt{\rho_{10}^2 + \rho_{01}^2}$
Blob: $\overset{+}{-}\lambda$
Line: $2^{-\frac{1}{2}}(\gamma \overset{+}{-} \lambda)$
Saddle: $\gamma$

**Basic Image Features** *Basic Image Features* (BIF) is a set of scale-space features obtained from Gaussian convolution of the image [32, 92]. BIF is based on the six Gaussian derivative filters up to the second order which are illustrated in Figure 2.5. This figure also illustrates the seven basic image features which are calculated according to Algorithm 2. $\rho_{ij}$ is the filter response from the Gaussian derivatives, $\eta$ is a parameter controlling the assignment of features as flat, and $\sigma$ is the standard deviation used in the Gaussians. Each pixel is assigned to the highest feature response, so every pixel in the image is represented by one BIF. Changing the standard deviation will result in a scale change of the feature response, so these features can easily be adapted to different scales. Orientation can also be included in the feature descriptors. The slope, line, and saddle features have principal directions, which has been used in for example [92] for orientated BIF, the so called oBIF's.

Crosier and Griffin [32] use the BIF for texture classification, where they obtain an image characterization as a histogram of the BIF distribution. To improve the discriminative power of the features, each BIF is calculated over four scales, resulting in 1296 -dimensional ($6^4$) normalized descriptor histogram, which shows to have very good classification performance. We will later show that BIF also performs well for texture segmentation, see Section 3.4.

**Textons** Textons is a set of local image features, which was first described by Julesz [79]. Julesz used the term texton to describe a set of discriminative features of first and second order statistics. Later more operational approaches for the modeling of textons have been given. Examples include a generative approach described by Olshausen and Field [110] and a discriminative procedure described by Leung and Malik [90] and Malik *et al.* [98], and these methods are compared by Zhu *et al.* [159]. We will give a short explanation of the fundamental elements of the two approaches. The generative method for modeling textons is based on a over-complete set of basis functions, for example Gabor basis or Laplacian of Gaussians. Let us denote the set of basis functions $\psi$. An image can then be modeled as

$$I = \sum_{i=1}^{n} a_i \psi_i + \epsilon. \tag{2.10}$$

$\epsilon$ is an image noise element and $a_i$ are the contributions of the basis functions. 144 basis functions are used in [110], but the modeling is done sparsely so most of the $a_i$'s are set to zero. The basis functions $\psi_i$ are treated as latent variables and they are inferred in a probabilistic model base on an EM-learning algorithm [159].

The discriminative approach for modeling is based on a set of filter responses. Each basis is convolved with the image giving a response in each pixel. These responses are placed in a vector with the same dimensionality as the number of filters. Let us denote the filters $\{\phi_1, ..., \phi_n\}$ and we denote the image $I$. From this we obtain the filter response vector

$$\mathbf{b} = [\phi_1 * I, ..., \phi_n * I]^T. \tag{2.11}$$

The hypothesis is that these features is well represented as a discrete set of image patterns. The image patterns can be found as clusters in the feature space and in [98] this is done using $k$-means. The cluster centers are the image

textons and the rest of the feature vectors are viewed as noisy versions of these patterns.

Both the generative and discriminative texton modeling are good at identifying characteristic texture features in the images that they are trained from, making useful as basic image elements for solving different computer vision tasks including image segmentation and classification.

Chapter 3

# Image segmentation

The work in this chapter is done in collaboration with Ali Shokoufandeh and Peter Bogunovich from Drexel University, Philadelphia, USA, and Henrik Aanæs from DTU Informatics, Lyngby, Denmark.

Image segmentation is the problem of partitioning an image into a set of coherent parts. This is useful for many computer vision applications like image retrieval, object recognition, object classification, image registration, etc. Typically these applications rely on matching the visual patterns of different objects. Most natural images contain several objects with occluding boundaries and the objects of interest may also be occluded by other objects. Segmentation can be a pre-processing step that allows objects to be matched with minimal influence of other scene elements. Segmentation is also very important tool for the logTracker system, where many of the visual features relevant for estimating wood quality and volume will benefit from image segmentation. Examples of applications are finding wood logs in the forest or estimating the shape of a log. Our segmentation procedure includes texture analysis which is important for the highly textured wood logs.

In this chapter a novel image segmentation method is introduced. The method is partly described in [35], and a segmentation example is shown in Figure 3.1 illustrating the potential of the procedure. Our segmentation method is based on *Iterative Function Systems* (IFS) and the principle of contraction which is

(a) (b) (c)

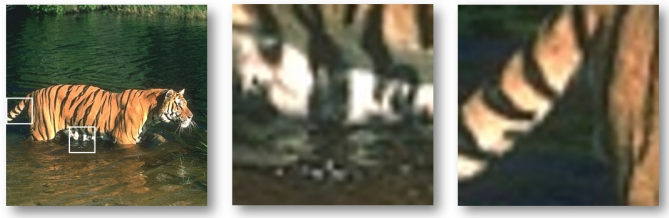
Figure 3.1: Texture segmentation from contextual contractive maps. Figure (a) shows a heterogeneous image created by composing a bark texture with itself rotated $90^o$ in a masked out area. The mask is obtained from the zebra as seen in figure (b) and the resulting segmentation is shown in figure (c).

also the foundation of our method. Our work on segmentation is inspired by the use of IFS for image compression, know as *Partition Iterative Function Systems* (PIFS) [4, 53, 76].

We will begin the chapter by discussing the problem of texture segmentation and how it relates to other work. We will then introduce PIFS and show how it is used for image compression. We use this theory to develop a novel set of image features that we call *kernel*-PIFS (*k*PIFS). The segmentation procedure is based on these features, and we will explain this procedure in the following. Finally, experimental results are presented which shows the high potential and flexibility of the procedure.

## 3.1 Related methods for segmentation

In segmentation we want to group pixels that share some property [23, 48]. The simplest segmentation problem is intensity based segmentation, where image regions with uniform color or intensity are identified. But many images have a degree of complexity that limits the usefulness of these methods, for example the problem of textured image regions. Texture segmentation is usually harder because identifying uniformly textured regions requires a texture analysis included in the segmentation procedure. Probably the hardest segmentation problems are identifying semantically meaningful regions that can have high variation in both texture and color within the same region. An example is people that often resembles large visual variation. In this chapter we have focused our work on texture segmentation.



(a) (b) (c)

Figure 3.2: Problem of boundary detection of textured objects. Figure (a) shows the original tiger image and Figure (b) and (c) are small image patches obtained from the white squares in Figure (a). Looking at the tiger as a whole it is quite easy to identify the boundary, but the black stripes merge into the background at a local level.

**Approaches for segmentation** Segmentation can be approached in many ways, but an important element is the degree of prior knowledge that is used for guiding the partitioning. Methods building on prior knowledge are called supervised segmentation, and this can be done by giving initial seed points to the segmentation algorithm [8, 75], by predefining the number of segments [7, 114], by searching for known image features [7, 98] or image classes [23], or by having a predefined shape model that is fitted to the image [12, 60]. Segmentation based on no prior assumptions is called unsupervised segmentation [48].

The typical way of segmenting an image is by image similarities. Supervised methods are typically guided by known image elements or features, whereas unsupervised methods are limited to the similarities within the image. This makes the choice of image characterization an important element for a segmentation method. We will now discuss some procedures for obtaining the image segments.

A common way to perform segmentation of textured images is to extract a set of texture descriptors and use the similarity of the descriptors to find homogeneous image regions. Typically, the segmentation should be found close to pixel accuracy, but the problem is that the scale of the texture is much larger than the pixels making it hard to find accurate boundaries. Malik *et al.* [98] point at this scale and boundary issue as the two major problems for texture segmentation. The issue of scale relates to the distribution of the repeatable image patterns making up the texture. The repetitions may vary over scale, for example the stripes of the zebra in Figure 3.1 (b). The stripes on the back are much wider than the ones on the leg, but both should be categorized as the same texture. Figure 3.2 illustrates the issue of detecting boundaries of textured objects. At a large scale the boundaries are quite easy to find, see Figure 3.2 (a), but at a local level the stripes of the tiger merge into the background, see Figure 3.2 (b)

and (c), which makes it impossible to segment the object based only on local pixel information.

**Splitting vs. merging** Segmentation procedures can roughly be categorized as either bottom-up or top-down. The top-down segmentation is performed starting with the image as a whole and splitting it until a desired segmentation level is reached. One of the simplest ways to perform top-down segmentation is through thresholding of intensity values. The basic approach is to find a number of thresholds for grouping the pixels which has been done in a number of ways. The simplest thresholding is binary segmentation, where the pixels are grouped in two. One criteria for a binary segmentation is formulated in [111] where a threshold value is selected to minimize the within class variance. Many other criteria for selecting thresholds have been developed and a survey can be seen in [128]. Despite the robustness and simplicity of the technique, making it good for example character segmentation, it cannot handle many natural images, especially images containing texture. A more advanced example of top-down segmentation is the normalized cut algorithm of Shi and Malik [132]. In this procedure the image is modeled as a graph, with each pixel being a node, and the segmentation is performed as graph-cuts.

In the bottom-up approach small image elements, for example pixels, are treated as initial segments, and the process is to merge these basic image elements. An example of a bottom-up approach is the region growing algorithm of Adams [2] where a number of seed regions is found. Image segments are obtained by adding neighboring pixels to the regions according to some similarity measure. Another bottom-up approach for segmentation is the mean-shift algorithm of Comaniciu and Meer [30]. Mean-shift is a clustering algorithm and the segmentation is performed by grouping the image pixels. Each pixel is treated as an image descriptor based on pixel intensity and the spatial location make up the descriptor. An iterative procedure is applied to each image descriptor, and in each iteration the mean value of a local neighborhood is found. The descriptor is replaced by this mean value until convergence. The neighborhood is found as all points within a given distance from the descriptor, and this distance is given as a parameter to the procedure. Mean-shift has later been used for other segmentation procedures such as based on wavelet features [48]. $k$-means is another cluster-algorithm that has been used for segmentation [72].

**Class-specific supervised segmentation** Knowing what you are looking for can be a great help for precise segmentation which has been shown in Borenstein and Ullman [23]. They perform object specific segmentation by matching a set of image patches with regions labeled as background and object. To handle

scale variation they obtain and match the templates at a range of scales, and the segmentation is done by selecting the most probable matching templates. In [22, 130] this method is extended to encoding the image in a tree structure and performing the segmentation using a normalized cut like criteria. Each node in the tree is assigned to be either foreground or background by combining the information obtained from matching the labeled templates and information from local similarity including intensity, texture, and boundary properties. In [85, 91] this was further extended to include neighbor information by the use of markov-random-field. These methods have been shown to perform well with a limited number of known classes, but extending them to a large number of classes will be hard, because class specific labeled image patches have to be available for each class.

**Active contours** The difficulty of texture boundary segmentation can be handled by modeling them as active contours. Active contours, also known as snakes, was introduced by Kass *et al.* [81] and has been used in a large number of texture segmentation procedures, see for example [6, 26, 70, 71, 74, 78, 114, 121, 124, 160]. The basic active contour as introduced by Kass *et al.* is an energy minimizing spline which contains an internal energy term, external energy term, and an energy term related to the image information. The internal energy affects the bending of the spline and consists of a first and second order element. The external energy is model-defined and acts like springs on the active contour. Features like edges and corners affect the image dependent term. These three energy terms are modeled as a weighted sum and the goal of the active contour is to minimize this sum which makes the snake follow the contours of the objects. The original formulation of the active contour model is problematic for texture segmentation, because it depends on image features like edges, and textured images typically contain many internal boundaries [75]. To overcome this issue the image energy term is modified also to be based on region characterization of the image by a set of image descriptors. In Houhou *et al.* [74] the descriptors are based on the principal curvatures of the image surface, in Paragios and Deriche [114] the descriptors are modeled as a mixtures of Gaussians for filter response distributions, and in Zhu *et al.* [160] as a mixture of Gaussians for intensity distributions. Finding the place to initialize an active contour can be problematic, but when this is solved the active contours give impressive results for unsupervised texture segmentation, which is also why they have become very popular.

**Watershed** Segmentation has been inspired by watersheds from topography, which are the phenomena of water flowing to the lowest parts in the landscape. Viewing the image as a landscape with the intensity being the altitude, seg-

mentation is performed by simulating flow of water. Vincent and Soille [150] presented an algorithm based on these ideas. The principle of the algorithm is to initially find the lowest parts of the image "landscape" and start flooding the image from here. This is effectively done by sorting the pixels and adding them to the image with increasing intensity, which leads to connected components making up the segments. Boundaries between segments are found when connected components meet. A similar procedure based on color utilizing the LUV color space is presented in Shafarenko *et al.* [129], where they segment natural images based on color gradients. A very fast version of the basic watershed algorithm is presented in Sun *et al.* [138] based on chain codes. They obtain high computational performance while retaining the precision. In Haris *et al.* [65] a watershed segmentation method is presented. In their procedure image is initially smoothed while preserving the edges and a set of segments is obtained using the watershed algorithm. The final segmentation is done by grouping neighboring segments based on an adjacency graph utilizing the edge response. The ideas of noise reduction and adjacency graph matching have similarities to our procedure.

**Repetitive patterns and image self-similarity** An important element of texture characterization is repetition of small image patterns [120], which has been used as the basis for several segmentation procedures. This use of repetitive elements relates both to texture and larger image structures. The segmentation procedure described in Zeng and Van Gool [157] is based on identifying repetitive patterns in images. $k$-means clustering of pixel color is used for initializing the procedure resulting in an over-segmentation. Based on these segments they find point wise repetitive patterns. The comparison of patterns is based on mutual information. This makes the method capable of handling imperfect repetitions of image elements, which is often seen in natural images.

Repetitive elements are also an important aspect of the segmentation procedure of Bagon *et al.* [8]. Their method is based on the principle of segmentation by composition described in [20]. They introduce what they call "a good image segment" as one that is easy to compose from parts of the segment but hard to compose from other parts. In their procedure image elements are compared by composing, in a puzzle-like manner, one part of the image from other image parts. If it is possible to compose large image elements form other large image elements, then these elements are likely to belong to the same segment [19, 21]. Salient image regions can also be detected, because these regions are hard to compose from other parts of the image. Their segmentation procedure is based on maximizing the sum of three measures including a foreground to foreground similarity measure, a foreground to background dissimilarity measure, and a boundary measure. The boundary measure favors large gradients. Similarity

is found within a transformation map, and the procedure builds on local evidence, i.e. similarity within a window, and global evidence, which is across the entire image. The similarities found in [8] are based on SIFT [95], color histograms, texton histograms [98], shape context [15, 16], and self-similarity descriptors [131]. Self-similarity is the property that one image element is similar to other image elements. An example is an image of a face where one half of the face is similar to the other half through a mirror transformation. Self-similarity is a common property in images especially for small image patterns. The self-similarity descriptors of Shechtman and Irani [131] capture local geometric self-similarities which enables matching of similar geometric shapes despite significant texture differences. This idea of encoding images in relation to self-similarities characterizes the geometry of objects, which is important in many applications besides segmentation. These applications include image registration, object recognition, object classification, etc. The segmentation results obtained from "segmentation by composition" [8] are very impressive, for example accurate segmentations of people.

**The suggested procedure** The procedure that we suggest for image segmentation is unsupervised, so the number of image segments and their localization is automatically found by the method. It combines the top-down and bottom-up approach, by initially decomposing the image into a set of sub-images structured in a quadtree. The decomposition is guided by IFS and contractive functions, and it is only done on boundary regions. We apply a bottom-up merging step based on planar graph merging for identifying the final image segments. Image elements are compared based on a set of descriptors that we developed from the ideas of PIFS, BIF, and textons.

## 3.2 PIFS for image compression

Our segmentation procedure builds on IFS and the principle of contraction. We have developed a novel set of image features called $k$PIFS which is based on the ideas of PIFS for image compression. We will first introduce PIFS and then describe the $k$PIFS image characterization.

**Partition Iterative Function Systems** Storing images as raw pixel intensities can take up much memory, and often images contain redundant information, which enables a more effective data representation. Image compression is the problem of transforming an image to a representation, which takes up less memory than the original pixel representation. Compression involves an encoding

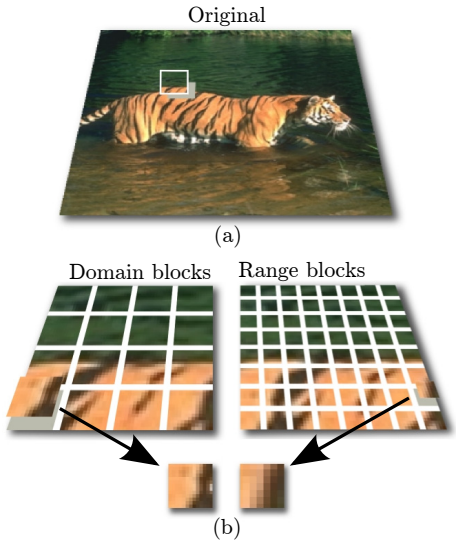



Figure 3.3: PIFS maps for image compression. This is a simple illustration of how the PIFS maps can be obtained. The original image shown in (a) is divided into a set of domain and range blocks, and a small sample of these blocks is shown in (b). The contractive PIFS maps are obtained by identifying the best matching domain block for each range block. In this process the domain blocks are down-sampled to have the same size as the range blocks. Furthermore, the domain blocks are allowed to be rotated and flipped, and the intensity can be changed within the limits of a linear transformation.

step, where the image is transformed, and a decoding step, where the image is transformed back to original pixel intensities. Compression can be exact, where the image can be reconstructed so every pixel value is the same as the original, or it can be lossy, which is when the reconstructed image looks like the original but not all pixel values are the same as the original [53]. For systems with limited memory or networks with limited transmission capacity, image compression can be a great advantage. We will now present the PIFS for image compression.

PIFS have been developed for image compression, and the fundamental observation behind this approach is that an arbitrary image can be encoded by a set of contractive functions. The decoding process is to iteratively apply these functions to an arbitrary image, making this image converge to a fixed point approximation of the original image. The contractive properties of the PIFS make the decoding independent of the starting image, so the typical choice is to start with an empty image of all zeros. The only information necessary for reconstructing the original image is the set of image maps, so the PIFS procedure

transforms the image representation from an array of pixel intensities to a set of contractive maps [53, 76]. Several authors have shown that the effective encoding and stability properties of PIFS make it suitable for image compression, see for example [4, 53, 56, 76, 151]. There have also been suggestions for using the PIFS encoding for other image processing tasks, see for example [4, 156, 158], which is in line with our technique.

The main problem in PIFS based image compression is to obtain a set of contractive maps with a fixed point image close to the original. Jacquin [76] was one of the first authors to suggest partitioning an image into a set of range blocks and larger domain blocks to obtain these maps, and the mapping is found by comparing domain and range blocks as illustrated in Figure 3.3. The image is encoded by self-mappings, i.e. contractive maps from a part of the image onto itself. We will now give a detailed description of the image compression technique.

**The compression technique** Details of the PIFS codes, how they are obtained, and how the image is decoded will now be presented. Image compression based on PIFS is lossy, but has potential for high compression rates. The method is based on a set of contractive self-mappings, which is mapping a part of the image onto itself. The only restriction is that these maps have to be contractive, but for simplicity reasons these maps are normally constrained to be affine transformations. If we let $f_i$ be an affine contractive self-mapping for the $i$'th element in the PIFS then we can write $f_i$ as:

$$f_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & \alpha_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ \beta_i \end{bmatrix}. \tag{3.1}$$

These affine self-mappings are obtained by comparing parts of the image to itself. A simple way of doing this, is by partitioning the image into a set of non-overlapping square sub-images of size $n_r \times n_r$. They are the range blocks and denoted $r_k$. Furthermore, we partition the image in to a set of square sub-images of size $n_d \times n_d$, which are called domain blocks and denoted $d_j$, and the domain blocks are allowed to be overlapping. In this example the size relation between the range and domain blocks are: $n_d = 2n_r$. The domain blocks are allowed to be overlapping. The affine transformations described in equation (3.1) should ideally fulfill the criterion

$$r_k = f_k(d_j), \tag{3.2}$$

but normally it is not possible to find exact matches between range and domain blocks. This block is found by estimating the affine transformation making the domain block as similar to the range block as possible. This map contains a geometric and an intensity element. Let us denote the intensity map $\phi$ and the geometric map $\gamma$, then we get the total map

$$f(d_j) = \phi(\gamma(d_j)), \tag{3.3}$$

where $d_j$ is the domain block. The first step of the geometric map is to downsample the domain blocks to the size of the range blocks. To reduce the computational complexity of the geometric map, the domain blocks are restricted to eight configurations, i.e. four 90 $^o$ rotations and the mirrored versions of these. Let us denote the geometrically transformed domain block $\bar{d_j}$. The intensity map is found by the following minimization

$$\min_{(\alpha,\beta)} ||\alpha\bar{d_j} + \beta - r_k||. \tag{3.4}$$

As shown in for example Fisher [53] this is simple to solve using least squares

$$\min_{(\alpha,\beta)} (\alpha\bar{d_j} + \beta - r_k)^2. \tag{3.5}$$

Given one geometrically transformed domain block $\bar{d_j}$ containing pixels $a_i, i = \{1, ..., n\}$ and a range block $r_k$ containing pixels $b_i, i = \{1, ..., n\}$, we want to minimize the following equation with regard to $\alpha$ and $\beta$

$$\psi = \sum_{i=1}^{n} (\alpha a_i + \beta - b_i)^2. \tag{3.6}$$

Finding the minimum squared distance of the pixel intensities in the two image blocks is done by estimating where the partial derivatives with respect to $\alpha$ and $\beta$ are equal to zero. The solution is

$$\alpha = \frac{\left[ n \sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i \right]}{\left[ n \sum_{i=1}^{n} a_i^2 - \left( \sum_{i=1}^{n} a_i \right)^2 \right]} \tag{3.7}$$

$$\beta = \frac{1}{n} \left[ \sum_{i=1}^{n} b_i - \alpha \sum_{i=1}^{n} a_i \right]. \tag{3.8}$$

The image is encoded by finding the domain block that minimizes $\psi$ from equation (3.6) and storing that map, i.e. the geometric map (origin, rotation, and mirror) and the intensity map ($\alpha$ and $\beta$). Both the intensity map and the geometric map have to be contractive for obtaining a fixed point image. The contraction of the geometric map is done by scaling down the domain block, which helps in obtaining the individual pixel values. If for example no geometric contraction was involved, we could have two image blocks acting as domain blocks for each other. These blocks would have an attractor with only one value for all pixels in the blocks, removing the detailed pattern of these blocks. Even though we will find a unique attractor without geometric contraction, there would be a risk of a poor reconstruction result. This makes the contractive property of the geometric map important. The intensity map is made contractive by ensuring that $|\alpha < 1|$, but in practice this has shown not to be strictly necessary [53]. It turns out that some intensity maps can be non-contractive as long as the PIFS encoding is contractive as a whole.

The decoding process is to apply these maps to an arbitrary image. Given a contractive PIFS encoding we know that it has one unique fixed point, so we will obtain the same fixed point independent of the initial image. The PIFS decoding is usually fast, as the example in Figure 3.4 shows.

The described procedure is relatively simple and can be extended in several ways. Decomposing the image into range and domain blocks does not have to be squares, but can be shaped as rectangles or triangles. Furthermore, the scale relation between the domain blocks and range blocks does not have to be two, as long as the domain blocks are larger than the range blocks resulting in a contractive geometric map. Changing the image domain from intensity to wavelet has also shown to be effective. In the intensity domain the compression often results in images with blocky artifacts, which can be avoided using the wavelet domain. Many other issues have been investigated for improving the quality and speed of the compression procedure, see for example [4, 53, 56, 152] for a discussion of these issues.

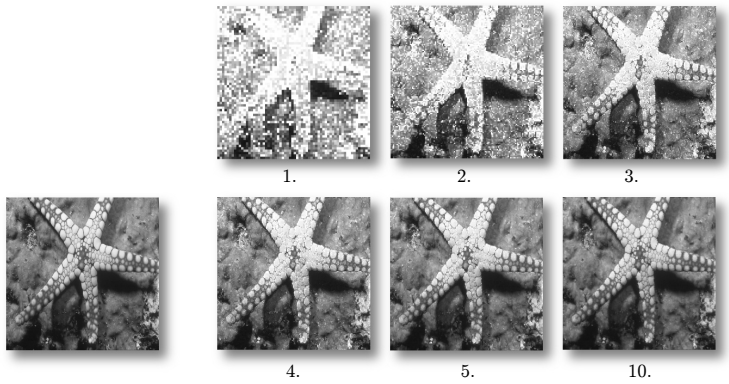



Figure 3.4: Reconstruction of an image using PIFS. Figure (a) shows the original and Figure (b) shows the iterations from one to five and iteration number 10 of the reconstruction. The shape of the see star is apparent already after the first iteration and it is almost fully reconstructed after 5 iterations. There is almost no change after 10 iterations.

**Related image processing based on PIFS** By far the most work based on PIFS is for image compression, but the method has also been used for other image processing tasks. Alexander [4] suggested an interesting extension of PIFS for edge detection. He found that edges are more likely to occur if the average collage error of the domain blocks is high, which he used for edge detection. Related to this procedure, Xu and Wang [154] suggested to use the collage error as an indexing technique for image retrieval. They found an image descriptor by building a histogram of collage error for the best matching domain block in each range block, and they used this descriptor for image retrieval of textured images.

Zhang *et al.* [158] used the PIFS encoding for a texture based retrieval system. The system is based on measuring the similarity between images by comparing PIFS image maps, i.e. both the geometry and intensity maps. The hypothesis is that similar images will have similar PIFS encoding. For identical images this will work perfect, but it will be problematic for images covering different parts of a scene because the set of domain blocks change. To overcome this problem they suggest to use a so called nona-tree which is built from a hierarchical splitting of the images into sub-images. Each sub-image contains a set of domain blocks which is used for encoding. This procedure will ensure that two images parts depicting the same scene element will have the same PIFS code.

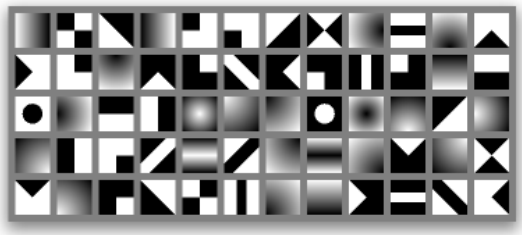


Figure 3.5: Domain kernels for $k$PIFS image characterization. Domain kernels are mapped to image range blocks, and from this we obtain an image descriptor as a normalized histogram of domain kernel distributions.

The common theme for several image retrieval methods are based on variations of of PIFS feature histograms [41, 108, 117]. These methods either encode the PIFS parameters as an index or bin the parameters into a histogram. The PIFS parameters used in the image descriptors include both the parameters in the intensity map, parameters in the geometric map, and the mean intensity value of the range blocks. These image encodings are used as image signatures for retrieval applications. Many suitable distance measures exists for solving this problem.

Our segmentation procedure is similar to the above mentioned techniques because it is based on a set of histogram descriptors obtained from the PIFS method presented by Jacquin [76]. But there are some significant differences in how we obtain our $k$PIFS features that differentiates our image characterization from the methods described above. Using $k$PIFS features is only one part of the method, and we will later describe how we use IFS and contractive maps for the actual segmentation. First we will explain our $k$PIFS features before we give an explanation of the actual segmentation procedure.

### 3.2.1 Image characterization from kernel-PIFS

Instead of the contractive self-mappings, which is the basis of traditional PIFS, we make an accurate characterization of image textures by replacing the domain blocks with an over-complete basis set of image blocks which we call domain kernels. An example of a set of domain kernels is illustrated in Figure 3.5. Our image descriptors is a distribution of domain kernels found by mapping the domain kernels to the image range blocks. We call this modification kernel PIFS ($k$PIFS).

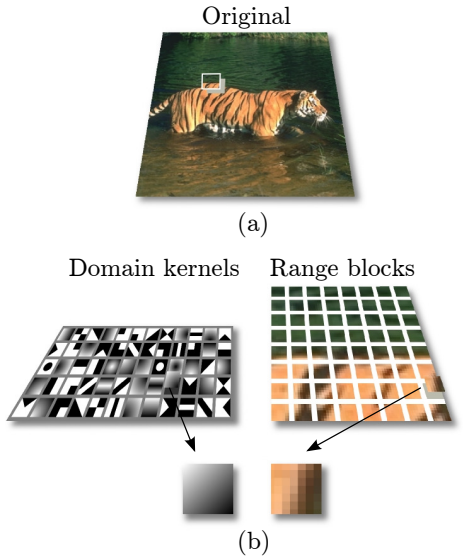



Figure 3.6: Mapping of domain kernels to image range blocks. The range blocks are exemplified from a small sample from the original (a), but in the actual application the whole image is divided into range blocks. Each range block is matched to the best mapping domain kernels illustrated in (b). Just the best mapping domain kernel is chosen here for illustrative purposes, but in the actual matching a number of best mapping domain kernels are chosen as explained in the text.

Figure 3.6 illustrates the mapping of domain kernels to the image range blocks. The size of the domain kernels and the range blocks are the same, so there is no scaling involved in these maps. We also avoid the reorientation of the kernels, because this will remove important information from the image characterizations. The procedure of the mapping is to decompose the image into a set of image range blocks just like in the traditional PIFS mapping. All the domain kernels are mapped to each range block, and the best mapping domain kernels will characterize the range block, i.e. the domain kernels resulting in the lowest collage error. The $k$PIFS descriptors are inspired by the textons [98, 110] and BIF [32]. Both textons and BIF densely describe local image patterns over the image, and this way characterize the basic image structure, which is similar to how we use $k$PIFS. An advantage of the $k$PIFS procedure is that the descriptors are localized precisely because there is no smoothing effect from a convolution, which is the case for both BIF [32] and textons [90]. We will now give a detailed description of the $k$PIFS descriptors.

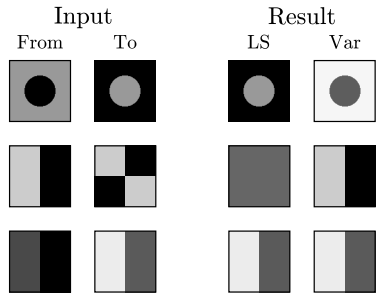



Figure 3.7: Mapping image blocks using Least Squares (LS) and our variance approach (Var). Each row displays a map from the image patches in the first column to the patches in the second column. The resulting maps are shown in column three (LS) and four (Var). The first image patch shows a blob that is mapped to a blob with inverted colors. LS maps perfect by inverting the color whereas the variance approach keeps the color relations and make a poor match. The variance approach does a better job in relation to a discriminative objective because this method reveals the difference between a light and a dark blob. For a block with half of the block being similar as shown in row two the LS approach results in a flat block, whereas the variance approach keeps the original intensities revealing the difference between the blocks. The last example shows a linear transformed block, and both methods are capable of finding the exact map.

***k*PIFS maps** The first element in the $k$PIFS image characterization is to map the domain kernels to the image range blocks, and we will now describe the details for obtaining these maps. It should be noted that the $k$PIFS maps are not contractive. The usual way of mapping domain blocks to range blocks is by least squares, see equation (3.5), but we have found it better to find a map where the transformed domain kernel and the range block have the same standard deviation of the pixel intensities. Figure 3.7 illustrates some examples where this approach is superior to the normal least squares approach. Let us denote the standard deviation of the range block $\sigma_{r_k}$, the domain blocks $\sigma_{d_\ell}$, and the transformed domain block $\sigma_{f(d_\ell)}$. We have the linear transformation relating the domain and range blocks and we restrict the standard deviations of the range block and the domain block to be the same

$$r_k = f(d_\ell) = \alpha d_\ell + \beta \tag{3.9}$$

$$\sigma_{r_k} = \sigma_{f(d_\ell)}. \tag{3.10}$$

From this we get the standard deviation of the transformed domain block

$$\sigma_{f(d_\ell)} = \sqrt{\text{var}(\alpha d_\ell + \beta)} = \sqrt{\alpha^2 \text{var}(d_\ell)} = \alpha \sigma_{d_\ell}. \tag{3.11}$$

From this we can estimate the parameters

$$\alpha = \frac{\sigma_{r_k}}{\sigma_{d_\ell}}, \tag{3.12}$$

$$\beta = r_k - \alpha d_\ell. \tag{3.13}$$

This restriction enables us to normalize the image blocks, so they can be processed as data points. Observe that the ranking of the domain kernels relative to a given range block will stay the same independent of an affine transformation of the range block. This is under the constraint that both range blocks and domain kernels have the same $\sigma$. This observation enables us to normalize the blocks to have zero mean and standard deviation of one. Then the matching problem is simplified to the absolute difference between the range blocks and domain kernels

$$d'_\ell = \frac{d_\ell - \mu_{d_\ell}}{\sigma_{d_\ell}}, \tag{3.14}$$

$$r'_k = \frac{r_k - \mu_{r_k}}{\sigma_{r_k}}, \tag{3.15}$$

$$\epsilon'_{[d_\ell, r_k]} = ||d'_\ell - r'_k||_1. \tag{3.16}$$

The normalization will be highly influenced by noise if $\sigma_{r_k}$ is small, and it is not possible to estimate it if $\sigma_{r_k}$ is zero. To avoid this situation we use a measure of flatness for the range blocks which is done relative to the mean intensity value. Let us denote flatness of the range blocks $b_f$ and let the flatness be estimated as

$$b_f = \frac{\sigma_{r_k}}{\sqrt{\mu_{r_k}}}. \tag{3.17}$$

We categorize the range block as flat if $b_f < t_f$ where $t_f$ is a threshold. Measuring the flatness relative to the mean value allows for higher variance at high intensities.

The domain kernel maps are used for calculating the $k$PIFS image descriptors and we will give a detailed explanation of the elements involved in this characterization.

**$k$PIFS descriptors** The $k$PIFS descriptors are based on all elements involved in the $k$PIFS maps. This includes the best mapping domain kernels and the $\alpha$, $\beta$, flatness, and collage error parameters. All of these elements are represented as a histogram, and in the following we will explain how to obtain this histogram. The image descriptors are built by combining domain kernel distributions and map parameters. The map parameters are continuous, so their distribution is found using a discrete set of bins within an interval.

The set of parameters is treated as independent random variables, which might cause loss of valuable information, but we found it to be a good compromise for limiting the dimensionality of the descriptor vectors. The combinatorial consequence of treating a set of parameters as dependent variables leads to high dimensional descriptors. An example is the BIF features in Lillholm and Griffin [92], where each pixel has six possible values over four scales resulting in a 1296 dimensional descriptor. The size of the descriptors will grow exponentially with the number of scales for BIF descriptors and sample points for the LBP. One extra scale in the BIF feature will give 7776 dimensional descriptors and six scales will result in 46656 dimensions. The problem of high dimensional descriptors is that we will need many observations to get a reliable distribution. There are ways to reduce the descriptor dimensionality, for example principal component analysis (PCA) [67], but we have chosen the simpler solution, i.e. to treat the parameters independently. We will first explain how the descriptor containing the domain kernel is built and then explain the other parameters and how they are combined.

A subset of the domain kernels is used for characterizing each range block, so the histogram characterizing the range block will contain a contribution from a number of domain kernels. The contribution to the histogram is found by weighing the domain blocks relative to the similarity of the range block. Let us denote the dissimilarity between a domain kernel and a given range block by $\delta$, the standard deviation of all the domain kernels and this range block $\sigma_\delta$ and the minimum dissimilarity by $m_\delta$, then we obtain the weight as

$$t_w = \gamma\sigma_\delta + m_\delta, \tag{3.18}$$

$$w_{[r_k,d_\ell]} = \max\{t_w - \delta, 0\}, \tag{3.19}$$

$t_w$ is the parameter for the distance weight. $\gamma$ is a parameter controlling the contribution of the closest domain kernels. If $\gamma$ is close to zero, only the closest domain kernels contribute, whereas larger values will increase the number of non-zero domain kernels. We found a value of $\gamma = 1$ to be appropriate. The normalized descriptor vector is obtained as

$$\mathbf{w}_{r_k} = \left( \sum_{d_\ell \in D_I} w_{[r_k,d_\ell]} \right)^{-1} \{w_{[r_k,d_1]}, ..., w_{[r_k,d_n]}\}. \tag{3.20}$$

For each of the parameters $\alpha$, $\beta$, flatness ($b_f$), and collage error ($\epsilon$) we define an interval which we divide into $n_b$ equally spaced bins. The interval for $\alpha$ is $(0.005, 0.15)$, for $\beta$ it is $(-25, 0)$, for $b_f$ it is $(0.1, 4.5)$, and for $\epsilon$ it is $(20, 60)$, and if any parameter value is above or below the defined interval it will be placed in the first or last bin. We found the descriptors to be good with a low number of bins and $n_b = 3$ to be good choice. The different descriptor parts is weighed with a weight of 0.5 for the kernel part and 0.125 for the other parameters. These parameters are all obtained empirically. The resulting descriptor is of 89 dimensions.

Color can also be added to the image descriptors. This is done in a very simple manner by first histogram equalizing the image. Then the frequencies of R, G, and B values within a descriptor patch are counted in $n_c$ equal sized intervals resulting in a $3 \times n_c$ normalized histogram which is concatenated to the $k$PIFS texture descriptor. This placement of colors in bins is similar to the binned texture parameters, i.e. $\alpha$, $\beta$, flatness and collage error. Equal weights are given to the texture and color parts of the descriptor. We found the best performance with a relative low number of bins and we chose tree bins for each color channel. The resulting color added descriptor has a dimensionality of 98 dimensions.

In contrast to the PIFS image compression method we obtain the range blocks as overlapping image parts which gives a denser sampling of the image compared to traditional non-overlapping blocks. From these range block descriptors we can obtain an image descriptor from an arbitrary part of the image by combining range block characterizations and normalizing the obtained histogram.

## 3.3 Segmentation from contractive maps

Having described the $k$PIFS image characterization we will now describe the actual segmentation procedure. The principles of the procedure is shown in

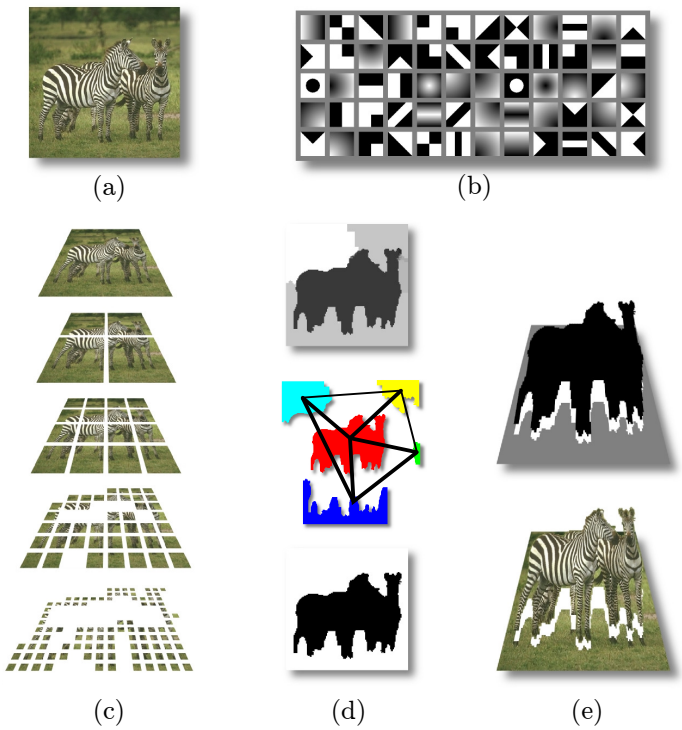



Figure 3.8: Elements of the suggested segmentation procedure. Figure (a) is the initial image. Figure (b) is the PIFS inspired features that we map to the original image to obtain a feature space representation. Figure (c) shows the decomposition of the image into small image samples. The samples in border regions are decomposed to deeper levels than samples in non-border regions. The top image of Figure (d) is the over-segmented image obtained from the top-down procedure in (c). These segments are merged in relation to the planar graph shown in the middle image resulting in the final segmentation in the bottom image. The final segmentation is shown in Figure (e).

figure 3.8 and it involves a top-down decomposition of the image and a bottom-up merging of the decomposed regions. From the decomposition we obtain a quadtree graph where each sub-image results in a descriptor forming the nodes in the quadtree. The edges in the tree connect the sub-image to its parent and children. The root node is a descriptor covering the entire image, the next four nodes cover a quarter of the image each, etc., see Figure 3.8 (c). The bottom up grouping of the decomposed regions is based on image discrepancy. We will start by explaining the top-down procedure.

**Top-down decomposition** In the first step of the top-down procedure we begin the construction of the quadtree by decomposing the image to some start level $l_{\text{start}}$, where level 1 is the root covering the entire image, by splitting the region nodes at each level into 4 child sub region nodes. Once we are at level $l_{\text{start}}$ we calculate a descriptor histogram for each of the $2^{2(l_{\text{start}}-1)}$ region nodes by averaging the $k$PIFS descriptors making up each region and normalizing. From this point onward iterative transformations for each node at the current level are constructed based on the local spatial neighborhoods and are applied to each of the nodes until an approximate convergence is reached. At this point stable regions are identified and the next level of the quadtree is constructed from the children of the nodes based on some stability (or discrepancy) measure.

In practice the choice of $l_{\text{start}}$ is important in determining the resulting segments. If $l_{\text{start}}$ is a small number then there is a risk that the region nodes identified as stable will still contain much heterogeneity while a larger $l_{\text{start}}$ can result in an over-segmentation. We have experimentally found that $l_{\text{start}} = 6$ is a good choice as a start level, i.e. at $32 \times 32$ sub-image nodes.

An IFS consists of a set of contractive maps as described in section 2.2. Our segmentation method has similarities to the PIFS for image compression. Our method is based on finding a set of contractive maps $\mathbf{f}$ and where we obtain one map for each sub-image. The maps are applied in the feature space, i.e. on the $k$PIFS characterization of the sub-image. In the iterative mapping procedure the image descriptors are combined with their spatial neighbors. The idea is to make similar neighboring descriptors become more similar and let dissimilar descriptors keep their dissimilarity. We will now explain the details of obtaining the set of contractive maps.

The convergence of the transformation rely on properties of contractive transformations in a metric space which is described in Theorem 2.10 [122]. The importance of this theorem is that if we can show a transformation to be contractive in a defined metric space, then we are sure that some fixed point will be reached by applying the transformation iteratively. The metric space is defined as the set of $k$PIFS image region descriptor histograms which can be thought of as lying in the space $\mathbb{R}^d$. It follows that any metric on $\mathbb{R}^d$ can be chosen, but in practice however we have just used the $L_1$ distance metric, denoted by $\delta_{L_1}$ and defined as $\delta_{L_1}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$.

Specifically, given some descriptor $\mathbf{w}_i$ at the current level of the quadtree, let $\mathcal{N}_i$ denote the set of $\mathfrak{n} \times \mathfrak{n}$ spatially local neighbor descriptors around $\mathbf{w}_i$ but not including $\mathbf{w}_i$, and let $\mu_{\mathcal{N}_i}$ be the average $L_1$ distance from $\mathbf{w}_i$ to all of the

other descriptors in $\mathcal{N}_i$. We then denote a weighted average distance

$$t_{\mathcal{N}_i} = \psi \mu_{\mathcal{N}_i}, \tag{3.21}$$

where $\psi$ is some weighting constant. Let $\mathcal{N}_i' = \{\mathbf{w}_i\} \cup \mathcal{N}_i$. We now define a set of scalar weights for every descriptor in $\mathcal{N}_i'$ such that $s_{(i,j)}$ represents a measure of similarity between $\mathbf{w}_i$ and $\mathbf{w}_j$ for $\mathbf{w}_j \in \mathcal{N}_i'$. The weights are defined as

$$s_{(i,j)} = \max\{(t_{\mathcal{N}_i} - \delta_{L_1}(\mathbf{w}_i, \mathbf{w}_j))/c_i, 0\}, \tag{3.22}$$

where $c_i$ is a normalization constant so that $\sum_{j=1}^{|\mathcal{N}_i'|} s_{(i,j)} = 1$. In this way all $s_{(i,j)} \leq 1$, and each descriptor $\mathbf{w}_j \in \mathcal{N}_i$ has an associated similarity weight $s_{(i,j)}$ with the special scalar $s_{(i,i)}$ being the weight for $\mathbf{w}_i$. Now define a new descriptor $\mathbf{v}_i$ to be a linear combination of the descriptors in $\mathcal{N}_i$ as

$$\mathbf{v}_i = \sum_{\mathbf{w}_j \in \mathcal{N}_i} s_{(i,j)} \mathbf{w}_j, \tag{3.23}$$

and our affine transformation $G_i$ for descriptor $\mathbf{w}_i$ is given by

$$G_i(\mathbf{w}) = s_{(i,i)} \mathbf{w} + \mathbf{v}_i. \tag{3.24}$$

We iteratively apply $G_i$ to $\mathbf{w}_i$ obtaining $\mathbf{w}_i^n = G_i^{\circ n}(\mathbf{w}_i)$ until convergence, but due to the simple affine form of $G_i$ it is particularly easy to demonstrate the contractivity of the transformation. For arbitrary descriptors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we have

$$\delta_{L_1}(G_i(\mathbf{x}), G_i(\mathbf{y})) = \sum_{j=1}^{d} |(s_{(i,i)} x_j + \mathbf{v}_{i_j}) - (s_{(i,i)} y_j + \mathbf{v}_{i_j})|. \tag{3.25}$$

Notice that the $\mathbf{v}_{i_j}$'s all cancel out and the $s_{(i,i)}$ can be factored out, simplifying to

$$\delta_{L_1}(G_i(\mathbf{x}), G_i(\mathbf{y})) = s_{(i,i)} \sum_{j=1}^{d} |x_j - y_j| = s_{(i,i)} \delta_{L_1}(\mathbf{x}, \mathbf{y}), \tag{3.26}$$

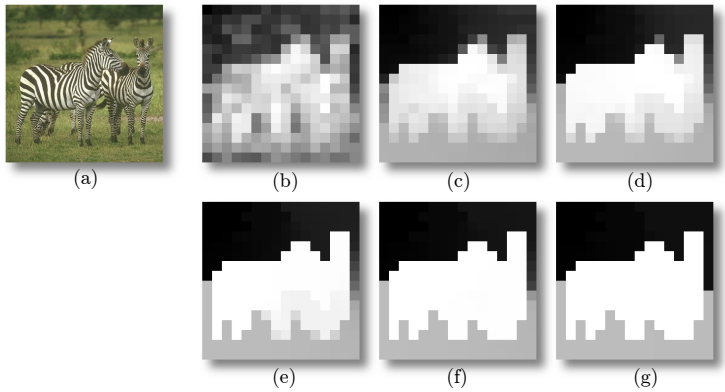



Figure 3.9: The contractive step with lowering $\psi$. The image is decomposed into $16 \times 16$ sub-images each characterized by a $k$PIFS descriptor. Each image shows the $L_1$ distance between the descriptor in the upper left corner and the other descriptors. The original image is shown in Figure (a). Figure (b) is without contraction, (c) the first step, (d) is the second, (e) is the fourth, (f) the seventh, and (g) the tenth step. In each step $\psi$ is lowered with 0.1 and the start value of $\psi = 1$. In each step the contractive function is recalculated.

and since $s_{(i,i)} \leq 1$, we have that $G_i$ is either contractive or it does not move $\mathbf{w}_i$ at all. If $s_{(i,i)} < 1$ we are guaranteed by Theorem 2.10 to reach a fixed point descriptor which we can denote by $\overline{\mathbf{w}}_i$. In the case of $s_{(i,i)} = 1$ the descriptor will not move and the final descriptor will be the same as the one we start out with. But we always choose to apply the function set on the same image that we used for obtaining the function set, so this situation will have no practical effect. In practice the convergence is quite fast and we generally need less than 10 iterations for $\epsilon = 0.01$.

The neighborhood weighing factor $\psi$ affects the segmentation result. Choosing a large $\psi$ results in relatively large but rather non-uniform image segments, whereas a small $\psi$ will give many small image segments. To ensure that the image is not over-segmented, but still contain uniform regions we reapply the contractive step at the same level in the quadtree with decreasing $\psi$. This is illustrated in Figure 3.9.

When the fixed point descriptors $\overline{\mathbf{w}}_i$ are reached for all regions at the current level, we identify the stability of each region based on the discrepancy of its fixed point to the fixed point of its neighbors. Since $G_i$ average each $\mathbf{w}_i$ with its similar neighbors, there is a strong possibility that sub-images in the regions with high local discrepancy after the iterative procedure will cover different textures. To

avoid misclassifications we split and repeat the contractive mappings on these regions at the next level of the quadtree, as illustrated in Figure 3.8(c). The discrepancy of a node is measured by comparing $\overline{\mathbf{w}}_i$ to the fixed points of its four spatially nearest neighbors which we denote by the set $\overline{\mathcal{N}}_i$. Let $\mu_{\overline{\mathcal{N}}_i}$ denote the average $L_1$ distance from $\overline{\mathbf{w}}_i$ to the descriptors in $\overline{\mathcal{N}}_i$ and let $m_{\overline{\mathcal{N}}_i}$ denote the maximum distance from $\overline{\mathbf{w}}_i$ to $\overline{\mathcal{N}}_i$, then the discrepancy measure of the region is defined as

$$\mathcal{D}_i = \mu_{\overline{\mathcal{N}}_i} + m_{\overline{\mathcal{N}}_i}. \tag{3.27}$$

Though we are only concerned with splitting and reprocessing unstable regions, in practice all regions are split, and the stable regions are fixed and will not change their descriptors. From $\mathcal{D}_i$ we are able to calculate a border measure for each node where $\max\{\mathcal{D}_j : j \in \{1, \ldots, N_k\}\} > 0$ as

$$\mathcal{B}_i = \mathcal{D}_i / \max\{\mathcal{D}_j : j \in \{1, \ldots, N_k\}\}, \tag{3.28}$$

where $N_k$ is the total number of nodes at the current decomposition level. If $\max\{\mathcal{D}_j : j \in \{1, \ldots, N_k\}\} = 0$ then we set $\mathcal{B}_i = 0$. $\mathcal{B}_i$ determines how $\overline{\mathbf{w}}_i$'s children descriptors are calculated. Let $\{\mathbf{w}_{(i,j)} j \in \{1, \ldots, 4\}\}$ denote the 4 initial descriptors of $\overline{\mathbf{w}}_i$'s children used in the next level of the quadtree. If $\mathcal{B}_i = 0$ then the region is stable and there is no chance of $\overline{\mathbf{w}}_i$ covering a boundary region and so we assign $\mathbf{w}_{(i,j)} = \overline{\mathbf{w}}_i$ for all children. When $\mathcal{B}_i > 0$ we let $\{\mathbf{v}_{(i,j)} : j \in \{1, \ldots, 4\}\}$ denote the descriptors of the child regions calculated as the normalized sum of $k$PIFS histograms in the same manner as at the starting level $l_{\text{start}}$. Then we obtain the new descriptors as

$$\mathbf{w}_{(i,j)} = (1 - \mathcal{B}_i)\overline{\mathbf{w}}_i + \mathcal{B}_i \mathbf{v}_{(i,j)}. \tag{3.29}$$

An example of the resulting decomposition is shown in Figure 3.10.

**Bottom-up merging of regions** Upon the completion of the top-down procedure we obtain a quadtree decomposition of the image with leaves representing non-overlapping stable image regions. The goal of the bottom-up procedure is to merge these leaves into homogeneous clusters which form the final segmentation.

Our approach begins by forming a planar graph $G$ so that the vertices of $G$ are the leaf nodes and an edge $(i, j)$ is formed between vertices representing adja-

Figure 3.10: Top-down decomposition of border regions. The shaded areas in the images in the left column are image parts with high discrepancy, i.e. border regions, and the right column shows the fixed point image.

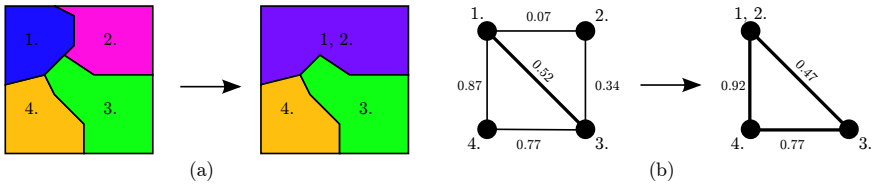



Figure 3.11: Bottom-up merging of image regions. Part (a) shows the obtained segments and (b) show the corresponding graph. Edge weights are given similarity between the segments. In the right hand of (a) and (b) segments 1 and 2 of the left sides of (a) and (b) are merged.

cent image regions with edge weight equal to $\delta_{L_1}(\overline{\mathbf{w}}_i, \overline{\mathbf{w}}_j)$, the distance between the associated fixed point descriptors. The bottom-up procedure then merges adjacent vertices of $G$ based on edge weight. Let $\alpha_i$ denote the percentage of the total image covered by vertex $i$. Then $\alpha_i$ is considered in the merging, so

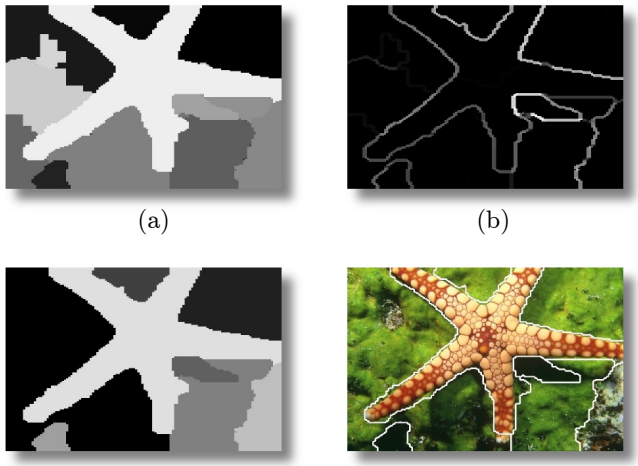


Figure 3.12: The bottom-up merging procedure. Figure (a) is the over-segmented image obtained from the top-down procedure where each segment has its own color. Figure (b) illustrates the boundary discrepancy with bright colors indicating high discrepancy. High discrepancy indicates large difference between segments. Figure (c) shows the final obtained segments, and Figure (d) is the final segments shown on the original image.

the smallest regions will be forced to merge with the most similar neighboring region and when merging any two vertices $i, j$ the ratio $\frac{\alpha_i}{\alpha_i+\alpha_j}$, $\frac{\alpha_j}{\alpha_i+\alpha_j}$ is used for weighing $i, j$ so that the merged vertex has a descriptor which is mostly influenced by the relatively larger region.

The merging of vertices is done in two steps. Initially we merge all vertex pairs $i, j$ where the edge weight is close to 0, i.e. less than some small positive $\epsilon$. These regions had nearly identical fixed points and the disparity is most likely only due to the fact that the fixed point is approximated. In the second step we let $\Delta_G$ denote the average weight in the current graph $G$ which is updated after each merging is performed. We proceed in merging the vertices $i, j$ with the smallest current edge weight until the relative weight $\delta_{L_1}(\overline{\mathbf{w}}_i, \overline{\mathbf{w}}_j)/\Delta_G$ is larger than some threshold $\gamma_{\text{merge}} \in [0, 1)$. Figure 3.11 gives an illustration of the process and Figure 3.12 shows an example on a real image.

## 3.4 Experimental results

In this section we show the experimental results of our procedure. The images used for testing our procedure are from the Berkley image database [101], the UIUCTex texture database [89], the Brodatz textures [24], the Internet, and images taken by the author. Our procedure has shown to be very powerful for texture segmentation, which is demonstrated on a set of composed and a set of natural gray scale images, and we compare our method composed textures in Fauzi and Lewis [48] and to natural images in the state of the art methods of Hong *et al.* [71] and Houhou *et al.* [74]. Our method is very flexible and we show that it can handle many different segmentation problems by changing the set of image descriptors. First this is demonstrated on a set of color images, which is done by adding color information to the $k$PIFS descriptors. We have also tested the procedure on images characterized by BIF features [32], see Section 2.3. Most results are displayed and qualitatively evaluated.

The first image set is from the UIUCTex texture database [89], and the images are built by composing randomly chosen texture images into five regions, consisting of four squares and an elliptical center region. Some segmentation results are shown in Figure 3.13. The first six segmentations results shown in Figure (a) to (f) are nearly perfect where all five regions are found and the borders are found very precisely. In the next four segmentations shown in Figure (g) to (j) the five regions are also found, but there are larger errors along the boundaries. Errors mostly occur where the boundaries are locally hard to distinguish. In the last six segmentations shown in Figure (k) to (p) the number of classes is wrongly found. For most of these images the textures can be very hard to separate because they visually appear similar, for example in Figure (o) where the transition from the center texture to the upper left texture is smooth. The lower right texture in Figure (p) is separated into two along a scale gradient. The blocky structure seen in for example Figure (h) probably caused by the textures being hard to characterize at a fine scale making the descriptors very influenced by the contextual inheritance. Therefore, the descriptors will stay in the same segment throughout the top-down procedure. Overall, we find the performance good. In most cases the right number segments are found and the boundaries are found well. For most wrongly segmented textures it is easy to see the reason for the errors.

Figure 3.14 shows results of the segmentation procedure on a set of natural images. The dominating textures are separated quite well, but especially in images with textured backgrounds there is a tendency for the background to be partitioned into many regions. This is for example the case in Figure (a), (b), and (h). A relatively easy image is shown in Figure (e) where the background is very uniform. In Figure (f) there is a horizontal segment which is probably

caused by the intensity gradient in this part. Figures (c) and (d) are especially nice segmentations, because the tree trunks are found well despite the highly textured background. These examples illustrate shows the large variation that the procedure can handle.

In Fauzi and Lewis [48] they perform unsupervised segmentation on a set of composed Brodatz textures [24]. We have compared the performance of our method to their results by composing a set of images from the same set of Brodatz textures. The compositions are made by randomly selecting a number of these textures. We composed the images similar to [48], which is shown in the first two rows of Figure 3.15. The composition into these square parts is very well suited for our method because the descriptors precisely cover one texture. To avoid this overlap between image composition and model we also made some images with a round center surrounded by four elements divided in a cross. The textures are D001, D011, D012, D017, D018, D024, D026, D053, D054, D055, D065, D066, D074, D077, D086, and D102. The result of our procedure is shown in Figure 3.15 and we obtain a very good segmentation for composed images of two to five different textures. For most images there were only small errors along the boundaries of the textures. We tested if the procedure was able to find the correct number of textures in the examples containing five samples, shown in row two of Figure 3.15, and in 19 out of 20 we were able to find the correct number. Only in the example with the texture shown in the lower right corner in Figure (h) was wrongly segmented, and the question is if this should be one or two textures. In [48] they found 7 of 9 composed images. We conclude that the selected Brodatz textures are quite easy because all images are characterized by one scale and with good sharpness and contrast. This is not the case for the for example UIUCTex data [89] (Figure 3.13), which was also harder to segment.

We have tested our procedure on the same set of images from the Berkley segmentation database [101] as was used in Hong *et al.* [71] and Houhou *et al.* [74]. The results are compared in Figure 3.16 and 3.17. Our method performs well compared to that of Hong *et al.* especially in Figure 3.16 (a) and (c). It should be noted that the focus of their paper is on texture scale applied to texture segmentation. The results of Houhou *et al.* is very comparable to our method and both methods find the depicted objects in all images. In Figure (e) and (f) our method finds some extra textures which are clearly distinct. In figure (k) and (l) both methods find segments that are not part of the starfish, but are clearly distinct textures. There are slight differences in the two methods, for example in Figure (a) and (b) where the object is merged with a part of the background in our method, whereas it is found very nicely in the method of Houhou *et al.* [74]. An example in favor of our procedure is Figure (m) and (n) where part of the head and the tail is not found well by their method, whereas it is segmented nicely by our procedure.

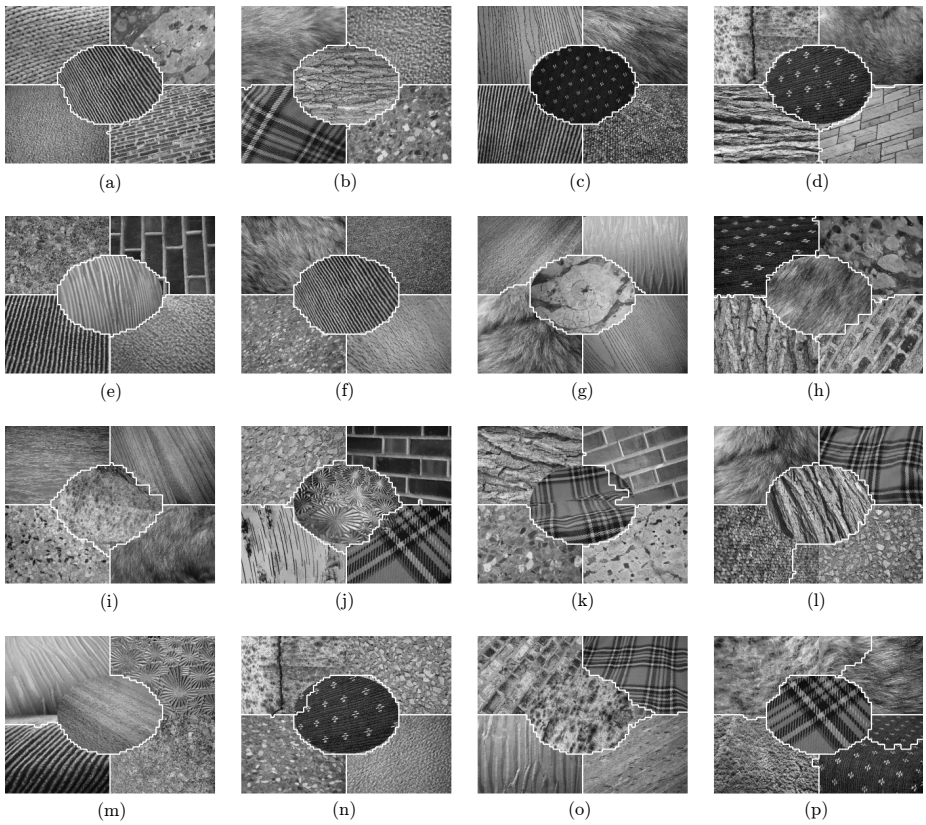



Figure 3.13: Segmentation results on a composed set of texture images from the UIUCTex database [88]. The borders are shown with a white line and texture segmentation is performed using the $k$PIFS texture features.

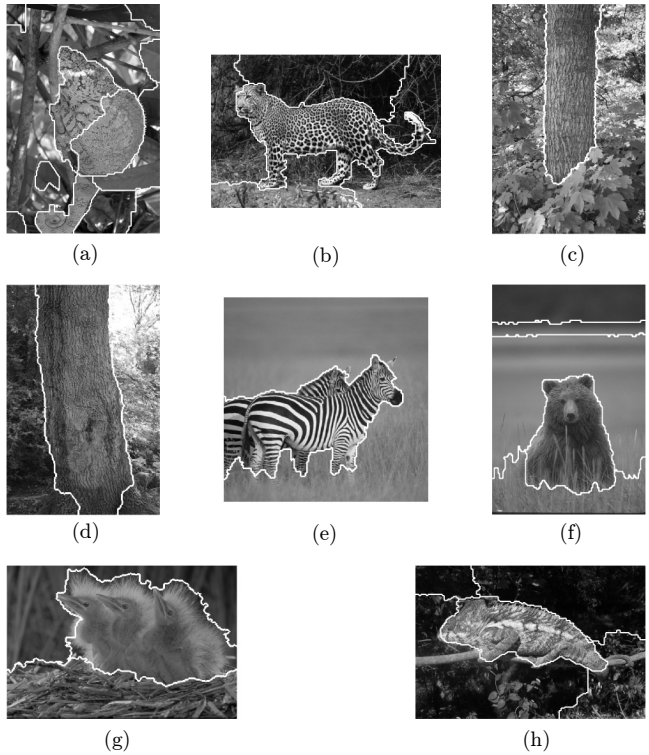



Figure 3.14: Segmentation results on a demanding set of images. The borders of the segmentation are shown with a white line and the segmentation is based on $k$PIFS.

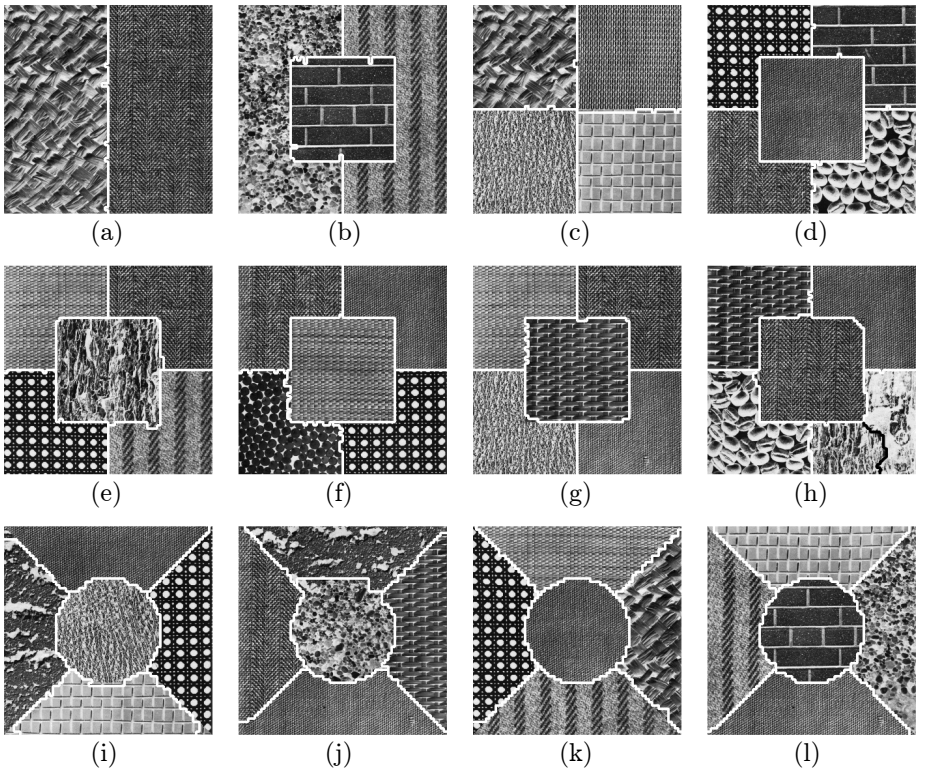
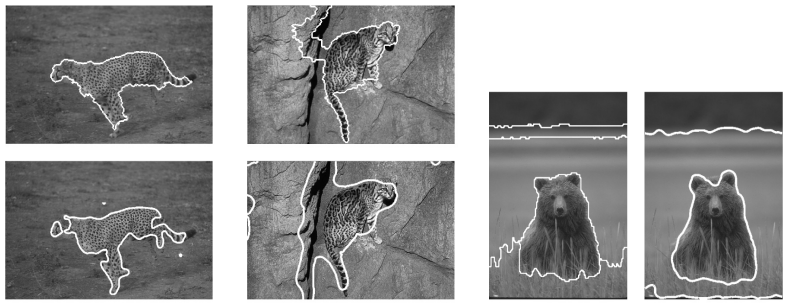



Figure 3.15: Segmentation of the Brodatz textures [24]. The composition of the textures is inspired by the segmentation procedure of Fauzi and Lewis [48]. Segmentations borders are marked with white lines except (h) where a part in the lower right is marked in black to make it visible.



(a) (b) (c)

Figure 3.16: Comparative results. This figure shows our results compared to that of Hong *et al.* [71]. Our results are on the top in (a) and (b) and right in (c).

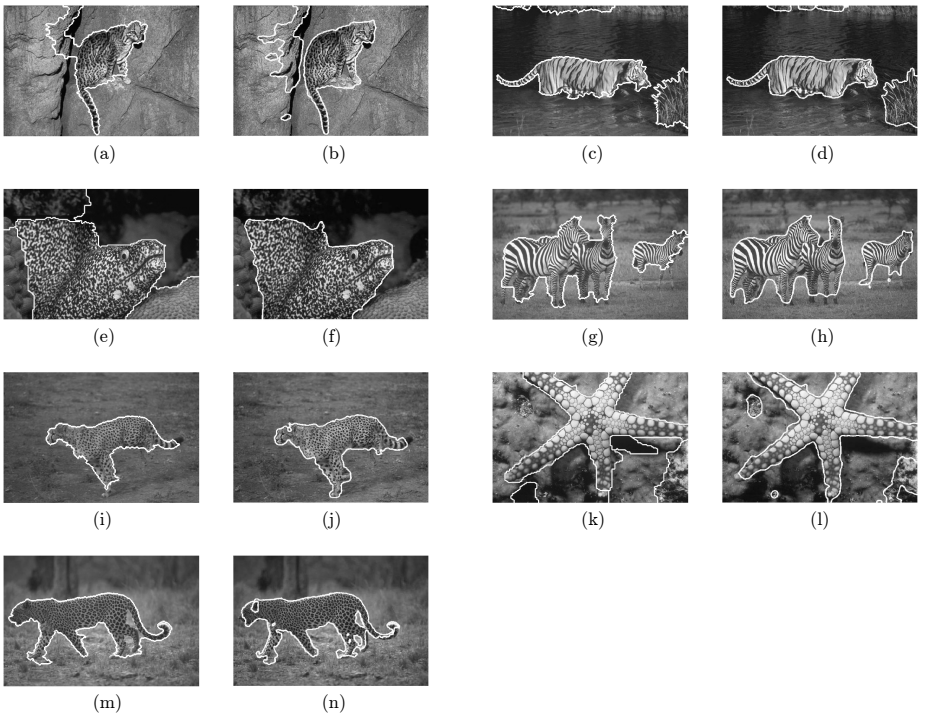



Figure 3.17: Comparative results. This figure shows our results in column one and three compared to the results from Houhou *et al.* [74] in column two and four.

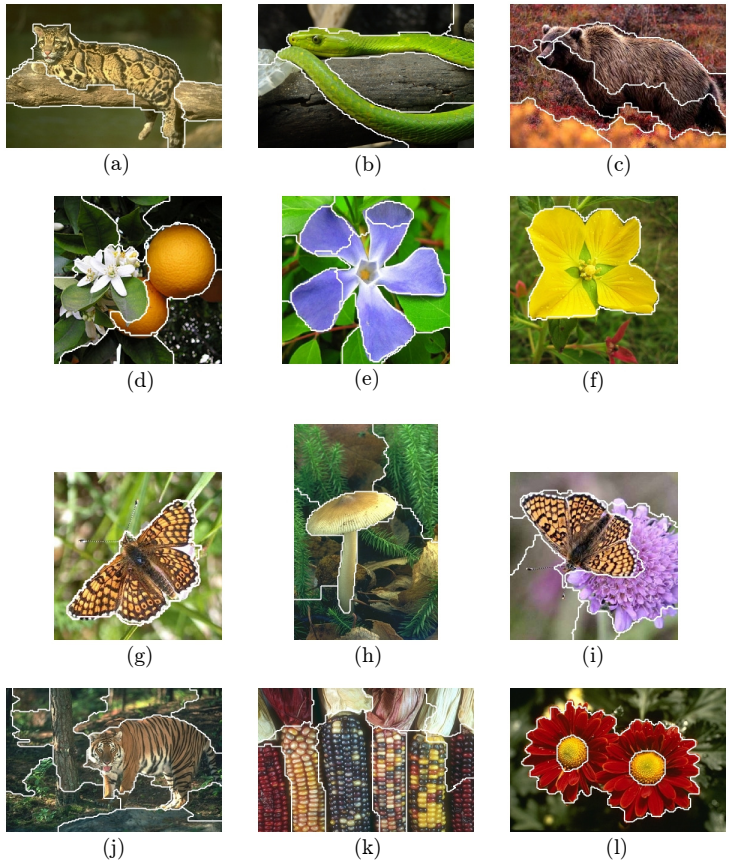



Figure 3.18: Segmentation results with descriptors including color.

In Figure 3.18 we show an experiment where we have included color in the descriptor. The performance of the color segmentation was quite good. Especially images with clear color differences, like Figure 3.18 (b), (f), and (l), came out very well. But also the oranges in Figure (d) and the flower of Figure (e) are segmented into meaningful parts. Images with complex combinations of textures and color are now also possible to segment. A good example is the tiger in Figure (j) but also the butterflies in Figure (g) and (i) are found very precisely. The conclusion is that the method is strengthened by adding color to the image descriptors. This experiment shows that the method is capable of doing a range of tasks by varying the input image descriptors, which is extended in the next experiment.

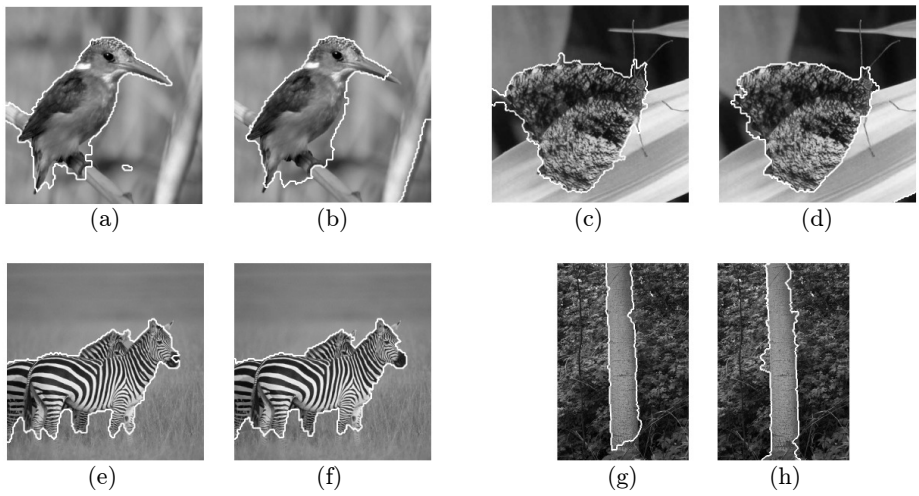



Figure 3.19: Experiments with $k$PIFS and BIF texture descriptors. The images in column one and three are $k$PIFS based segmentation and the images in column to and four are the texture segmentation based on BIF descriptors.

Figure 3.19 shows the performance of our method when the $k$PIFS descriptors are replaced by BIF features [32, 92], see Section 2.3. In this experiment we use BIF as a basis for the texture descriptors in the same way as the domain kernels of $k$PIFS. The image descriptors are built from the seven basic BIF responses across $n_b$ scales resulting in a $7 \times n_b$ histogram in each pixel. These histograms are $L_1$-normalized. The characterization of the sub-image in the top-down step of the procedure is done by averaging the descriptors. This characterization differs slightly from the description in [32] and [92] where they characterize the image by the dominating feature response. But we found this soft assignment to improve the performance of the model. We found three scales, i.e. standard deviations $\sigma_b = \{0.5, 1, 2\}$, to be a good choice. The results shown in Figure 3.19 illustrates that the model is also capable of performing equally well with the set of BIF features. In the first image of the bird shown in Figure (a) and (b) the $k$PIFS performs slightly better than the BIF by finding the front part of the bird very well, whereas in Figure (c) and (d) BIF performs better than $k$PIFS. But both descriptor sets are capable of precise image segmentation.

## 3.5 Conclusion on image segmentation

The problem of texture segmentation is to identify coherent image parts based on textural characteristics. Textured image elements can be hard to distinguish at a detailed level making precise boundary detection difficult. Furthermore, some textures have a similar visual appearance and a precise texture characterization essential for a segmentation procedure. We propose a method to solve these problems that makes use of contextual characterization and the principles of contractivity.

The proposed procedure builds on a top-down image decomposition and a bottom-up merging of similar image segments. To identify local structural characteristics of the image we introduced a novel set of texture features based on PIFS. These features employ a set of *domain kernels* which makes up an over-complete basis of simple image patterns. The top-down aspect of our method consists of a decomposition of the image from larger to smaller regions with each region forming a node in a quadtree and characteristic information is passed down the tree. By utilizing contractivity, we described a technique for iteratively exchanging information in a local neighborhood resulting in fixed points. Often these fixed points form a slight over-segmentation of the image, and to identify regions belonging to the same texture segment we employ a bottom-up merging step. This bottom-up merging is based on a neighborhood graph with edges between neighboring segments and edge weights explaining similarities between the segments. Connected nodes are merged if they fulfill a global similarity criterion. Experimentally we have demonstrated state of the art performance on a set of natural textured images. Furthermore, we showed the flexibility of our method, by obtaining equally good results using a basis set of over-complete scale space features, and color.

Texture segmentation is very useful for the logTracker system, and there are many tasks where segmentation is necessary for gaining useful information. This could for example be identifying wood logs in the forest, either before they are harvested or when they are placed on the forest floor. In many cases the wood logs have a characteristic texture that makes them distinguishable from the background. Another important issue is the quality of the wood which is related to the visual information. This includes the shape of the wood log, number and sizes of knots, presence of rot, width of year rings, etc. The identification of these characteristics could benefit from texture segmentation, and an example is segmenting the surface of a wood log or part of it and classifying the segments as being knots or bark.

The proposed procedure has potential for many applications because of its simplicity and high performance, but the method still needs improvements to make

it operational. Speed of the procedure is the first major concern. In our very simple MATLAB implementation it takes around five minutes to perform a segmentation of a $480 \times 720$ pixel image, which is very far from that half or one second that will be available in an industrial application. But parts of the procedure could be done in parallel, for example the descriptor assignment. Another speed-up could be to avoid many of the calculations in the contractive step, where descriptors are moved closer together. Some descriptors are moved very little and the calculation of their displacement could possibly be avoided. To enable this speed up it will be important to determine how large a part of the descriptors this concerns. Another obvious speed-up would be to choose another programming language. So, despite the apparent long calculation time, there is a high potential for bringing that down. Another important issue for improving the procedure is the image characterization. Some textures are hard to distinguish using our $k$PIFS texture characterization, so there are potentially better results using an improved kernel set. We have only used $k$PIFS blocks of size $8 \times 8$ with an overlap of half a block. These parameters could be adjusted and it might also improve the performance to conduct a multi scale feature assignment. There are still many issues to address for future improvements of our segmentation procedure, but despite this the method is simple, flexible, and has a high performance, which makes it potentially good for solving demanding segmentation problems. The general framework proposed may also be useful to other fields of computer vision.

Chapter 4

# Image classification

There is a wide range of uses for automatically assigning an image to a specific class, and this has made image classification an important and very active field of research. Many image elements are useful for guiding the classification, for example color, image gradients, or features such as edges, corners, blobs, etc. Images in particular pose a great difficulty, because textures contains many internal features such as edges and corners that are not at object boundaries. As discussed in the previous chapters, texture characterization is especially important for the motivating problem of this thesis, the classification of textured objects is essential for the logTracker system. In this chapter we will give an example of wood species classification based on Active Appearance Models (AAM) and texture, but many other classification problems are relevant for the logTracker system. An important step in solving these problems is an investigation of methods for texture classification.

We will start this chapter by briefly discussing some practical applications of texture classification and its potential use in the logTracker system before presenting our paper on AAM and co-occurrence matrices for classification of tree species [34]. The paper is included in its original form, with only minor clarifying changes. Finally, we conclude and give a perspective on future needs for classification applications in relation to the logTracker system.

(a) (b) (c)

Figure 4.1: Images of some biological objects. Image (a) contains three dominating textures which are examples of relatively different textures, i.e. the feathers of the birds, the nest, and the background. Image (b) is a close up image of the bark texture of an oak tree showing a relatively uniform texture. Image (c) shows a birch tree, which has a very non-uniform bark texture, and the background is also highly textured and non-uniform. Classification of textures in (a) and (c) would require a segmentation prior to the actual texture classification whereas (b) is a typical image for texture classification.

## 4.1 The problem of classification

The classification of biological objects addresses the general problem of categorization and recognition of highly varying objects. Many biological objects show particularly large visual variation with textured and irregular surfaces, some examples of which are shown in Figure 4.1. The problem of wood log classification are illustrated in Figure 4.1 (b) and (c). Wood logs have a highly textured and irregular bark, and the background forest as shown in Figure 4.1 (c) further enhances the difficulty for classification methods.

### 4.1.1 Related methods for texture classification

Classifying an image based on textural appearance requires a texture characterization and a classifier applied to these characteristics. This characterization concerns some of the same aspects as the texture characterization used for image segmentation, as described in Section 2.3. We will briefly discuss some issues that are special for the problem of texture classification before we give some examples of traditionally used classifiers.

**Image texture characterization** The characterization of texture for classification problems is typically done after a segmentation procedure, see for

example [3, 147], or on images containing only the one texture that has to be classified, see for example [36, 68, 89]. This excludes the issue of characterizing texture in boundary regions where there is a high risk that texture descriptors cover more than one texture, which is a great difficulty for texture segmentation. But the number of textures to distinguish in a segmentation procedure is typically less than many texture classification problems. This is reflected in the number of samples of some widely used test databases. Examples are the UIUCTex database [89] containing 25 textures, the CUReT containing 61 textures [36], and the KTH-TIPS containing an extension of 10 of the CUReT textures at different scale and angle. A unique texture characterization is also very important for classification, as we discussed in Section 2.3, especially considering the large number of textures to distinguish. We will now describe some of the commonly used classifiers.

**Classifiers** The choice of classifier can have great influence of the performance of a classification procedure. There is a wide range of classifiers, which has been designed to solve many problematic classification problems, see for example [18, 42, 67]. We will not go into detail in this large field of research, but rather mention just a few popular methods for image classification.

One of the simplest methods for classification is the $k$-nearest neighbor classifier (kNN). A sample is assigned to the majority among the $k$ nearest neighbors from a training set. Despite the simplicity of the method it is often a robust and flexible way to obtain good classification results. Another approach is to assign the sample to the most probable class using a maximum likelihood procedure, which is referred to as the Bayes classifier in our classification paper (Section 4.2). If the image descriptors are normally distributed random variables, then the probability of a given class can be estimated as

$$P(\mathbf{x} \in \mathbf{c_i}) = \frac{1}{(2\pi)^{N/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right), \tag{4.1}$$

where $\mathbf{x}$ is the image descriptor, $\mathbf{c_i}$ is class $i$, $\Sigma_i$ is the covariance matrix of class $i$, and $\mu_i$ is the mean value of class $i$. The mean value and the covariance matrix has to be computed from a training set. The covariance matrix is inverted so it is important that it has full rank, which can be a problem for high dimensional data. This can be solved by using a diagonal covariance matrix or using PCA to reduce the dimensionality of the data [67].

Fisher's linear discriminant analysis, also known as canonical discriminant analysis, is a classifier that is based on the idea of projecting the data to a hyperplane

with optimal separation. This hyperplane has a dimensionality of $\min(d, n_{\mathbf{c}} - 1)$ where $d$ is the dimensionality of the data points and $n_{\mathbf{c}}$ is the number of classes. This is done by minimizing the within class variance, which we denote $\mathbf{W}$, and maximizing the between class variance, which we denote $\mathbf{B}$. From this we get the following maximization problem

$$\max_{a} \frac{\mathbf{a}^T \mathbf{B} \mathbf{a}}{\mathbf{a}^T \mathbf{W} \mathbf{a}}, \tag{4.2}$$

where $\mathbf{a}$ is the $\min(d, n_{\mathbf{c}}-1)$ eigenvectors corresponding to the largest eigenvalues of $\mathbf{W}^{-1}\mathbf{B}$. The eigenvectors in $\mathbf{a}$ spans the data points [67].

Other popular classifiers include boosting, which is based on linearly combining a set of weak classifiers to construct a strong classifier [58]. Support vector machines (SVM) is a classification technique which is based on identifying an optimal decision boundary between data points, i.e. a boundary that minimizes classification error. This can be combined with the so-called kernel-trick, where data points are mapped to higher dimensions to perform non-linear classification using a linear classifier [67].

**Applications of texture classification** Texture classification is useful in many applications, and examples include remote sensing [29, 83, 100], medical applications [28, 139], and industrial applications [17]. Classification is also very important in relation to the logTracker project, for example identification of tree species. We will now discuss the use of classification for the logTracker system.

The tree species composition of the forest stands are important to consider when planning the harvest operations. Traditionally tree stands have been monocultures with only one species represented. But with the change in the direction of close-to-nature forest management, which has been a tendency in Europe the last 10–15 years [59, 87], forest stands with a mixture of tree species will become normal in the future. This makes a precise volume assessment of the different tree species hard. This assessment is normally done in the harvesting operation. A computer vision system for automatic species detection could be a great help for solving this, either as a system placed on the harvester or on one of the other machines involved in the transport of the wood logs from the forest to the industry. In the following we will present an investigation for species classification. Our investigation shows that it is possible to identify tree species from the surface characteristics of wood logs. This has previously been based on the public available BarkTex database containing bark texture of six tree species [86]. Examples are methods based on Gabor filters [113], co-occurrence

matrices [112, 118], and local binary patterns [119]. Our investigation differs in that it both involves a localization and segmentation based on an AAM and a texture analysis on the segmented bark.

Many issues have to be considered for a system to be operational in a real application. This includes placement of cameras, image acquisition in a forest environment with changing viewing angles, changing weather conditions, changing light, etc. These issues are essential and very difficult, and will require large amounts of data and intensive investigations. So, it is important to note that our work on classification by no means resembles a realistic species detection scenario, but it shows that species detection from bark characteristics is possible, in relation to segmentation and classification. It also indicates some of the problems that have to be solved in relation to the computer vision methods for a system to be operating in the forest. We will now present our paper on wood species classification.

## 4.2 Paper: Classification of biological objects

Our paper on classification of biological objects [34] is presented here in its original form.

**Authors:** Anders Bjorholm Dahl, Henrik Aanæs, Rasmus Larsen, and Bjarne Kjær Ersbøll, DTU Informatics, The Technical University of Denmark. (Presented at the Scandinavian Conference on Image Analysis, Aalborg, Denmark 2007).

### 4.2.1 Classification of Biological Objects using Active Appearance Modeling and Color Co-occurrence Matrices

**Abstract** We use the popular active appearance models (AAM) for extracting discriminative features from images of biological objects. The relevant discriminative features are combined principal component (PCA) vectors from the AAM and texture features from co-occurrence matrices. Texture features are extracted by extending the AAM's with a textural warp guided by the AAM shape. Based on this, texture co-occurrence features are calculated. We use the different features for classifying the biological objects to species using standard classifiers, and we show that even though the objects are highly variant, the AAM's are

well suited for extracting relevant features, thus obtaining good classification results. Classification is conducted on two real data sets, one containing various vegetables and one containing different species of wood logs.

### 4.2.2 Introduction

Object recognition is one of the fundamental problems in computer visions, and plays a vital role in constructing 'intelligent' machines. Our initial motivation for this work is the construction of an automated forestry system, which needs to keep track of wood logs. Many of the objects in our daily environment in general, and in our motivating problem in particular, are biological, and pose special problem to a computer vision system. The origin of these problems are the high degree of intraclass variation, which we as humans are very good at dealing with, e.g. consider the multitudes of ways a face or a potato can look. To enable biological variation to be handled in a classification system, we have to find methods for extracting discriminative features, from the depicted objects. AAM's have proven very well suited for addressing the problems of handling biological variation in the case of image registration [31]. It is thus highly interesting if this property of the AAM's also proves well for classification of objects, and how this should be implemented. Therefore, we have investigated AAM's for extracting discriminative features by conducting the following experiments

1. Classification based on **Multiple AAM's**, i.e. building an AAM for each class and assigning images of unknown objects to the most probable AAM.
2. Classification based on a **global AAM**, i.e. building one single AAM and using model parameters for assigning images of unknown objects to the most probable class.
3. Identify relevant discriminative patches from the use of an AAM. The object is identified by shape alignment from the **AAM and texture** is extracted and used for second order texture analysis.

Two data sets have been investigated in this paper, one containing vegetables and one containing wood logs. Experiment 1 and 2 have been conducted for both data sets and experiment 3 has been conducted only for the wood log data set.

**Related work** The environment plays a vital role in solving object recognition problems. In a natural environment objects may be seen from many different

angles, they may be occluded, light may change, etc. Efforts on solving this type of problem have been put in identifying local object features invariant to the changing conditions, see for example [127, 94, 95], and the way to match these features to a model, see for example [38, 39].

Controlling the environment in some way, gives the opportunity of easing the flexibility constraints of the object recognition system. In some situation object recognition on whole objects is a reasonable approach, giving the option of e.g. extracting global PCA features. This is done for face recognition by e.g. Turk & Pentland [144] with the eigenface, and Belhumeuer *et al.* [14] for their fisherface based on Fishers Linear Discriminant Analysis. No shape information is included with these methods. Edwards *et al.* [44] introduces the use of an AAM for face recognition based on both shape and texture information. Fagertun *et al.* [46] improves this method by the use of canonical discriminant analysis. AAM's have been used for related recognition problems, e.g. eye tracking by Stegmann [135] and Hansen *et al.* [62].

Pure texture has also been used for object recognition. The second order texture statistics based on co-occurrence matrices, was originally developed by Haralick *et al.* [64] for pixel classification. This method has been extended to object recognition by e.g. Chang & Krumm [27] using color co-occurrence histograms. Palm [112] does classification of different textures, including wood bark textures, using color co-occurrence matrices. He extends from gray level to color images and improves the classification.

In this paper we focus on object recognition in an environment with some degree of controlled conditions. We use a black background, controlled lighting, and we make sure that the whole object is visible.

### 4.2.3 AAM and texture features

In the following we describe the methods for extracting the discriminative features used in the three experiments.

**AAM** The AAM model - in 2D as it will be used here - is a description of an object in an image via it's contour or shape and it's texture. Each of these entities can be represented as a vector, i.e. $\vec{s}_i$ and $\vec{t}_i$ respectively, where the subscript, $i$, denotes an instance (image). The parameters of the AAM is, however, a lower dimensional vector, $\vec{c}_i$, and a specific AAM consists of an linear

mapping for $\vec{c}_i$ to $\vec{s}_i$ and $\vec{t}_i$, i.e.

$$\vec{m}_i = \begin{bmatrix} \vec{W}\vec{s} \\ \vec{t} \end{bmatrix}_i = \mathbf{\Phi}\vec{c}_i \ , \tag{4.3}$$

where $\mathbf{\Phi}$ is a matrix representing the linear map. The AAM or $\mathbf{\Phi}$ is estimated from an annotated training set. By optimizing the AAM to a new depicted object, an image close to the original is synthesized, and the model parameters $\vec{c}_i$ is a vector of principal components describing the unknown object with regards to the shape and texture of the object. The interested reader is referred to Cootes and Taylor [31] for a more detailed description and Stegmann *et al.* [136] for a detailed description of the model implementation.

**Features from multiple AAM's** In this case an AAM, $\mathbf{\Phi}_j$, is fitted to each class $\mathcal{C}_j$, i.e. the training set is divided into its component classes, and one AAM is fitted to each.

Here there is a feature vector $\vec{c}_i$ i specific for each model, and these features are not comparable, because they belong to a specific model and can not be used directly for classification.

Given an AAM for each class $\mathcal{C}_j$, you would expect the optimization of an image $i$ to perform best for the class that the object belongs to. Therefore, a goodness of fit would be a reasonable measure for classifying the object. For a given unknown object image textural difference between the object texture $g_{i_{obj}}$ and the model instance $\bar{g}_{mod}$ is calculated

$$E = \sum_{i=1}^{n} (g_{i_{obj}} - \bar{g}_{mod})^2 = ||g_{i_{obj}} - \bar{g}_{mod}||_2^2, , \tag{4.4}$$

where $E$ is the difference between the model image and the measured image by the squared 2-norm.

**Features from a global AAM** In this case a single global AAM, $\mathbf{\Phi}$, is fitted to instances of all classes. Following this, the $\vec{c}_i$ are calculated for each instance in the training set. The elements of $\vec{c}_i$, containing both shape and texture information, are used in a linear classifier.

**Textural warp** The basis for making a textural warp is knowledge of the log localization in the image. This comes from the AAM shape alignment. The warp is done by sampling pixels along elliptic curves on the surface of the logs using bilinear interpolation, see Figure 4.2. The elliptic curves are calculated from the shape of the end face of the wood log, and guided by the shape of the sides of the log. A square image of the bark texture is the result of the warp, which is illustrated in Figure 4.3. One end of the log is usually smaller than the other, resulting in a difference in the sampling intensity in the warped bark image. Other shape variations may result in the same kind of sampling variation. These small variations have not been considered as a problem for the texture analysis.

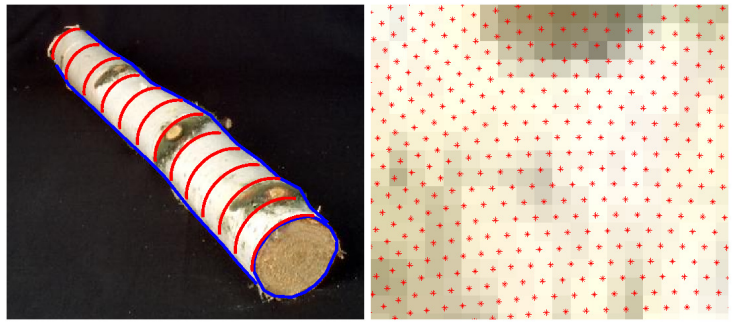


Figure 4.2: Illustration of bark texture warp. Left is an image of a Birch log shown with a few of the elliptic sampling curves shown in red. Blue lines show the AAM shape alignment. The right image is a close up of sampling points.

**Color co-occurrence matrices** As mentioned above, the AAM classification is extended by texture classification where the texture is obtained via texture warp as described in Section 4.2.3. This classification is done via second order textural statistics in the form of co-occurrence matrices (CM) [25, 64, 112]. The fundamental element of calculating CM's is the probability of a certain change in pixel intensity classes $(k, l)$ given a certain pixel displacement $\mathbf{h}$ equivalent to $Pr(k, l|\mathbf{h})$. The CM's can be extended to color, by calculating the displacements in each band and across bands. The CM's have proven useful for classification [112]. Sample CM's for the relevant bark textures is shown in Figure 4.3. In this paper the textures have been pre-processed by Gaussian histogram matching, in order to increase robustness to lighting conditions [25]. In this paper we use the following CM classes: contrast, dissimilarity, homogeneity, energy, entropy, maximum, correlation, diagonal moment, sum average, sum entropy, and sum variance.

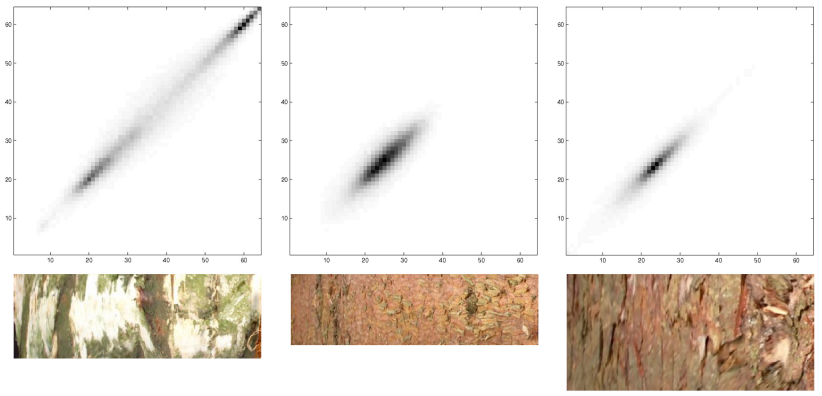


Figure 4.3: Illustration of co-occurrence matrix (*top*) of bark texture (*bottom*). Higher intensity illustrates larger co-occurrence. From left to right: Birch, Spruce, and Pine. The displacement is $(1, 1)$ in a 64 level image.

**Classifier** The classifiers used in experiments 1 to 3 are as follows:

1. **Multiple AAM's.** The model minimizing (4.4) is chosen.

2. **Global AAM.** Here three different classifications schemes based on the AAM feature vector are evaluated. These are: Bayes classifier, Canonical discriminant analysis, and LARS-EN classifier [45, 67, 161].

3. **AAM and Texture.** Here LARS-EN is applied to the texture, obtained via the AAM based warp described in Section 4.2.3.

### 4.2.4 Data

Experiments were conducted for two groups of biological objects: vegetables, see Figure 4.4 and wood logs see Figure 4.2. The vegetables are apples, carrots and potatoes and consist of 189 images totally where 27 are used for training the models, i.e. 9 from each group. The wood log data consists of the three species Scotch Pine, Norway Spruce and Birch. There was a total of 164 wood log images, 18 from each group were used for training. Also a reduced wood log data set, consisting of the 30 most characteristic logs from each group (90 in all) was used.

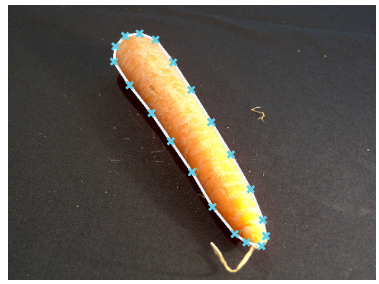


Figure 4.4: Illustration of AAM alignment. The light blue line illustrates the shape and the blue crosses mark the annotated points.

### 4.2.5 Experiments

All three experiments are illustrated in Figure 4.5. The procedure is as follows.

**Experiment 1** For each class there is built an AAM based on the training images. All models are matched to each of the test images giving model textures for all classes. The model texture is then compared to the original image by calculating the texture difference, see section 4.2.3. Classification is done by assigning the test image in question to the model giving the least texture difference.

**Experiment 2** In this experiment one AAM is built based on training images from all classes. The parameters from matching the model to a test image are used for classification. Based on these parameters the image is assigned to the most probable class.

**Experiment 3** As in experiment 2, one AAM is matched to a test image, but here the alignment is used for extracting texture features. To enable a calculation of co-occurrence features from the bark texture, a warp of the bark area of the image is conducted.

### 4.2.6 Results and discussion

Results of our three experiments are presented in table 4.1 and 4.2.

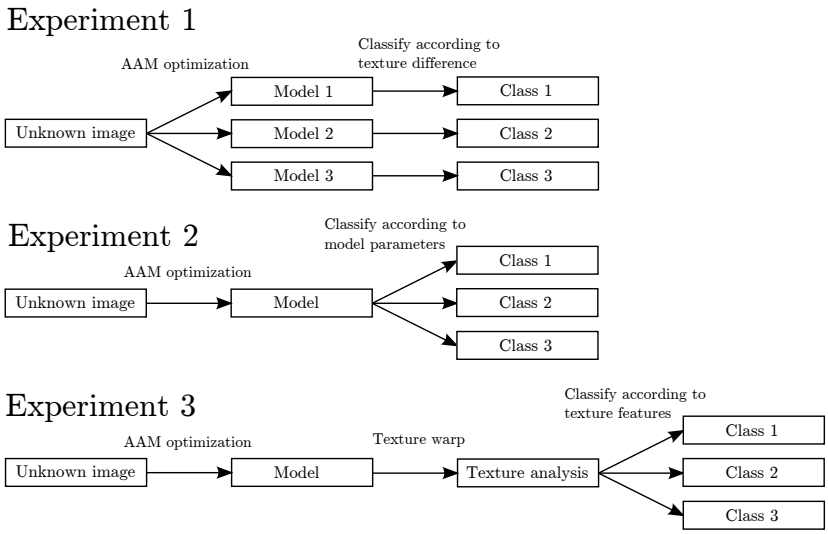



Figure 4.5: Schematic representation of the three experiments.

**Experiment 1** This experiment gave rather stable and good results. In the vegetable data only one image of a potato was misclassified as an apple, giving an average classification rate of 99.3%. The wood log data set gave stable classifications around 83% except for the whole log model, where many Spruce logs were misclassified as Pine logs and visa versa.

| Experiment | 1 | 2 | | |
|---|---|---|---|---|
| Model | Texture difference | Bayes | Canonical | LARS-EN |
| Vegetable | 99.3% | 100.0% | 93.7% | 100.0% |
| Log end | 82.8% | 70.2% | 71.0% | 75.5% |
| Large images | 83.5% | 64.1% | 48.3% | 82.1% |
| Whole log | 67.8% | 72.8% | 71.1% | 72.8% |
| Log end, reduced | 82.5% | 71.4% | 38.1% | 85.7% |

Table 4.1: Classification rates of experiment 1 and 2 using the different classifiers. In the Vegetable and Whole log experiments, the shape covers the entire object, whereas in the rest, only the end part of the object is covered. Large images refer to the use of higher resolution images. Reduced refers to the reduced data set.

**Experiment 2** For this experiment three different classifiers have been tried. The vegetable experiment obtains 100% correct classification with the Bayes classifier and LARS-EN but the canonical discriminant analysis gives only a

classification rate of 93.7%. The canonical discriminant analysis is also very unstable for the wood log data set. The Bayes classifier gives around 70% correct classification and LARS-EN around 80%.

AAM results in a relatively large number of features, and therefore, it is necessary to have many observations for training a classifier. A limited number of observations could be one reason for the relatively poor performance of the classifiers.

LARS-EN gives a good indication of the importance of the features in a linear model. For both data sets, the first two principal components are the most discriminative features. But there is large difference in the discriminative capabilities of the parameters from the two data sets, which also would be expected when the features are plotted, see Figure 4.6. The rest of the principal components are selected somewhat randomly, showing that feature reduction using PCA is not necessarily in accordance with classification criterions.

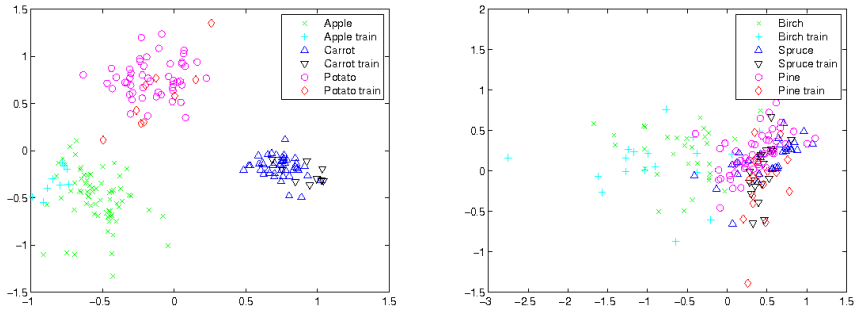



Figure 4.6: Plot of the first two principal components from the AAM in the vegetable experiment (*left*) and wood log experiment (*right*).

A problem encountered using canonical discriminant analysis, is a good separation of the training data, but a poor performance of assigning the unknown objects to the right classes. For the vegetable data, where we have a very good separation, this becomes very clear. We would expect to retain the separation, but that is not the case because of variation in training data. This sensitivity towards variation is a clear limitation to the canonical discriminant analysis.

The AAM's in the reduced data set is based on only 9 images of each class. This is probably too few to get good estimates of $\mathbf{\Phi}$, and could be a cause of the poor AAM classification performance.

**Experiment 3** The best performance for the wood logs is achieved by the texture analysis in experiment 3, reaching close to 90% correct classification rates for the whole data set, and 96.8% correct classification in the best experiment of the reduced data set. This shows an accordance between what we see as humans and the predictions of the model.

| Model | LARS-EN | |
|---|---|---|
| *Data set* | *whole* | *reduced* |
| Gray level | 78.4% | 93.7% |
| Gray level directional | 89.9% | 96.8% |
| Color | 89.5% | 90.5% |

Table 4.2: Classification rates of experiment 3 using LARS-EN for classification based on texture features.

A varying number of features are calculated because the three models shown in Table 4.2 contain a varying number of distances, directions, and color bands. In the first model we have 33 features (*only different distances*), in the second 132 features (*different distance and direction*), and 528 features in the third (*varying all three*).

Looking at which features are selected by the LARS-EN classifier, we can see a pattern in the features selected for classification. The sum average and the diagonal moment are the most frequently used features, even though, there is not a clear pattern in which features works best for classifying the wood logs. In contrast to Palms [112] investigations, the performance in our experiments is not improved by extending the analysis to include color images.

A hard task using the LARS-EN algorithm is to find the right stopping criterion [161]. The results presented here is the number of features giving the best classification rates. Therefore, the LARS-EN algorithm will be problematic to implement in a real world application.

In these experiments we have used logs of young trees where some important biological characteristics have not yet developed, for example the colored core of Scotch Pine, which could improve the hard distinguishing of Pine and Spruce.

### 4.2.7 Conclusion

We have investigated the use of active appearance models (AAM's, cf. [31]) for classification of biological objects, and shown that this approach is well suited

for different objects. Two data set, one of vegetables and one of wood logs have been investigated.

In experiment 1 an AAM is built for each class, and we obtain results close to 100% correct classification for the vegetable data, and around 80% classification rates for wood logs.

In experiment 2 one AAM for all classes is built, and model parameters for test images are used for classification. Most models gave 100% correct classification for the vegetables. On average the classification for the wood logs was not as good as experiment 1, and especially canonical discriminant analysis gave very poor results.

In experiment 3 LARS-EN has been used for classifying texture features, where only the wood log data is investigated. This experiment gave the best results classifying about 90% of the test set correct in the best cases of the whole data set, and up to 96.7% correct classification using a reduced data set.

It is hard to find a good stopping criterion for the LARS-EN model, which is problematic for classification. Therefore, we conclude that the most promising classification model is the texture difference used in experiment 1.

### 4.2.8 Acknowledgements

Thanks to Mikkel B. Stegmann for the AAM API and Karl Skoglund for the LARS-EN classifier [134, 136]. Also thanks to Dralle A/S for partial financial support.

## 4.3 Conclusion on image classification

We have presented a segmentation and classification procedure for solving the problem of classifying samples of wood logs to tree species. The presented method is based on Active Appearance Models and second order statistical texture features from co-occurrence matrices.

The segmentation element was not the focus of this procedure and we initialized the AAM close to the final position. Note that this should be automated for this application to operate in a real world system. Furthermore, the images of the wood logs are acquired under very controlled conditions from one viewing

angle and with plenty of light. For the logTracker system the requirements for a system will be more demanding and it should be able to handle changing light, occlusions, background clutter, etc. An advantage for a vision system placed on for example a harvester is that it will be possible to acquire a series of images of each log. This will provide the opportunity to get multiple views of the same log, and thus information from one image could be supported by the next. Many images will also increase the computational demands, so another important element is to keep the computational cost low. Most of these issues are not solved in the suggested procedure and should be investigated in future projects.

Our investigation shows that it is possible to perform species classification based on bark features. A classification rate around 90%, as we have achieved as the best results, is probably adequate for an operational system, but this issue has to be investigated further.

Bark textures are highly varying and the question is how suited the second order statistics, applied in our procedure, are for characterizing the common bark structure of a tree species. One issue is scale that is important for texture characterization, see Section 2.3, which is not accounted for in these co-occurrence matrices. The bark texture changes scale within the same wood log, and typically the texture is coarse at the bottom of a trunk and becomes finer towards to top, and the trunk might also change color. This issue has to be solved, which requires a texture characterization that is robust towards scale and large visual variation. Another way to handle this change in appearance could be to introduce subclasses within the same tree species. Another problem that has to be considered is the change of visual appearance over time. For example, wet bark looks different than dry bark, and bark with snow poses a special problem. Typically, these issues affect all the trees in a stand, so there will be some common properties for the same tree species. To handle these large variations the classification procedures have to be developed and new investigations have to be done. Despite the difficulties issues for this problem, it should be possible to find a solution to classify the perhaps 5-10 most important tree species used in forestry in Northern Europe.

Chapter 5

# Image retrieval

Image retrieval is the problem of searching a large collection of image data. This is traditionally done by associating metadata with the images, and using text search methods for image matching. An example is image search in web browsers where the associated text on the web pages act as metadata. This kind of metadata is not accurate, which a normal web search reveals. The typical experience is that many of the retrieved images not contain what was searched for. This is one example that has motivated an intensive research in content based image retrieval. Image retrieval is also a way to perform individual object recognition, which is done using a database of known objects. The kind of object recognition is what the motivation of this topic in relation to the logTracker system. An image retrieval system could potentially solve the tracking problem of wood logs and make it possible to document timber origin.

In this chapter we present a *bag-of-words (BOW)* image retrieval system that is based on our work in [33]. We have tested our procedure on a public image database [109]. Our contribution is a large scale database search approach that builds on SIFT features [95], where the dimensionality is reduced and color is added to the feature descriptors. We will begin this chapter by discussing the background for image retrieval. Then we will present our paper on the suggested approach for large database image search. Finally we will discuss the limitations of the BOW approach in relation to the logTracker system and give some ideas for how to overcome these limitations.

## 5.1 Related methods for image retrieval

Many of the methods for image retrieval builds on local image features, which are obtained from image regions with expected high information content related to the depicted objects. These regions are the so called interest points, and the image is characterized as a set of interest point descriptors. We will now discuss some techniques for image retrieval and the associated image descriptors.

**Image interest points and descriptors** Typically, image retrieval is done for large image collections and time is a critical issue. A popular approach for solving this has been to use the interest point descriptors as a bag-of-visual-words, where the descriptors are treated independently. The expectation is that two images will contain the same objects if they have the same set of descriptors. Therefore, it is important that the distribution of descriptors characterizing an object is unique, and that this distribution can be found under changing conditions of the image acquisition. This is done by making the descriptors invariant to scale, rotation, translation, deformation, light, occlusion, clutter, etc. This invariance includes both the detection of the interest points and the associated descriptors. We will now discuss the detection of interest points, and then discuss the assignment of an image descriptor to these points.

Schmid and Mohr [127] were among the first to build a model for image retrieval based on a distribution of interest point descriptors. They find interest points as the maximum principal curvature. In the Scale Invariant Feature Transform (SIFT) developed by Lowe [94, 95] interest points are found as the maximum in difference of Gaussians in scale space, making the SIFT interest points invariant to scale change. Affine transformations, which for example happens when the viewpoint change, is a problem for the SIFT interest point detection. But an example, where this has been solved, is the Maximum Stable Extremal Regions (MSER) [102]. This procedure builds on ideas related to segmentation and watershed algorithms. Interest points are detected as image regions that resemble maximum stability as a function of intensity change. The principle is to threshold the image at each intensity level, for example the interval $[0, 255]$, and look at how connected regions change. Regions that change very little over a large interval are detected as maximum stable. By fitting an elliptical shape to the obtained regions, and calculating the descriptors relative to this, the characterization is made invariant to affine transformation. MSER is one example of an affine covariant region detector, and it is compared to five other detectors in [61]. This include the Harris-affine and Hessian-affine detectors, see for example [104], edge-based and intensity based region detectors, see for example [145, 146], and an entropy-based region detector [80]. The detectors were compared relative to change in viewpoint, scale, blur, JPEG artifacts, and

light. The conclusion of the survey was that all detectors perform well, but the MSER performed best on most image types. Lately, the MSER region detector has been extended to color regions [55] resulting in better performance.

After the detection of interest points a descriptor is calculated. In the procedure of Schmid and Mohr [127] this was done as a nine dimensional histogram of Gaussian derivatives at a number of scales, which makes this descriptor invariant to scale change. The SIFT descriptor is typically found as a 128 dimensional histogram of directional image gradients calculated around the interest point. In Lowe's original procedure [94, 95] the scale invariance is found by estimating the gradients at the same scale as the interest point was found. Furthermore, the descriptor is made invariant to rotation by orientating it relative to the maximum gradient direction. The SIFT descriptor has also been used for affine invariant descriptors, for example by estimating the SIFT around a MSER [109]. In [105] image region descriptors are compared, which includes SIFT [95], gradient location and orientation histogram (GLOH) [105], shape context [16], PCA-SIFT [82], spin image [88], steerable filters [57], differential invariants [84], complex filters [125], moment invariants [148], and cross correlation of sampled pixel values. The descriptors are evaluated in relation to the same changes as the region detection survey [61], but also including image rotation. GLOH and SIFT are concluded to be the best high dimensional descriptors, whereas gradient moments and steerable filters are the best performing low-dimensional descriptors.

**Image retrieval techniques** Having characterized the image as a collection of interest point descriptors, the next step is to estimate the similarity between images. In Schmid and Mohr this was done using a nearest neighbor voting algorithm [127]. The object recognition system presented by Lowe [95] is also based on nearest neighbors. To improve speed he employs a Best-Bin-First search algorithm [13], but despite fast nearest neighbor approximation algorithms, the complexity of this nearest neighbor feature search grows dramatically with the number of images, and it can only handle a limited image database. Sivic and Zissermann [133] introduced the idea of characterizing an image as a distribution of descriptor representatives, the so called "visual words", instead of the original descriptors. These representatives are found by clustering the features and selecting the cluster center point as the representative. Furthermore, they adopted ideas from text retrieval and used indexing techniques for large image collection based on the visual words. It should be noted that the frequency of visual words is related to how useful they are for uniquely characterizing an image. Visual words occurring in many images contain less information than visual words that occur in few. Therefore, frequent words are weighed down relative to the infrequent ones. The images are matched by building a database index with a reference to each image in the database. This is done in a pre-

processing step, and this can dramatically speed up the matching at runtime. Assigning the image descriptors to visual words become a bottleneck in very large image collections. Therefore, Nistér and Stevénius [109] propose to use a hierarchical tree structure, both for clustering the image descriptors and for retrieving the images. This tree based search significantly increases the speed of the matching. Lately other approximation algorithms have been developed to solve this problem of feature assignment in high dimensional data, see for example [106, 116].

In our image retrieval procedure we have addressed some of these issues, which build on the ideas of invariant features and a bag-of-visual-words. In the following we will present our paper [33] in its original form. The paper was presented at The First International Workshop on Internet Vision, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska, 2008.

## 5.2 Paper: Image retrieval

**Authors:** Anders Bjorholm Dahl and Henrik Aanæs, DTU Informatics, The Technical University of Denmark. Presented at International Conference on Computer Vision and Pattern Recognition, Anchorage 2008, Alaska, USA.

### 5.2.1 Effective Image Database Search via Dimensionality Reduction

**Abstract** Image search using the bag-of-words image representation is investigated further in this paper. This approach has shown promising results for large scale image collections making it relevant for Internet applications. The steps involved in the bag-of-words approach are feature extraction, vocabulary building, and searching with a query image. It is important to keep the computational cost low through all steps. In this paper we focus on the efficiency of the technique. This is obtained by substantially the dimensionality of the features by the use of PCA and addition of color. Building of the visual vocabulary is typically done using k-means. We investigate a clustering algorithm based on the leader follower principle (LF-clustering), in which the number of clusters is not fixed. The adaptive nature of LF-clustering is shown to improve the quality of the visual vocabulary using this. In the query step, features from the query image are assigned to the visual vocabulary. The dimensionality reduction enables us to perform exact feature labeling using kD-tree, instead of

approximate approaches normally used. Despite the dimensionality reduction to between 6 and 15 dimensions we obtain improved results compared to the traditional bag-of-words approach based on 128 dimensional SIFT feature and k-means clustering.

### 5.2.2 Introduction

The bag-of-words approach have shown promising results for large scale image search [77, 109, 116, 126, 133] and for image categorization [49, 143, 155]. There are three main steps in the bag-of-words representation: (*i*) Feature extraction from the database images, (*ii*) building the bag-of-words representation, (*iii*) and searching with a query image.

**Image features (*i*)** The image features are descriptions of local image patterns, see for example [61, 95, 102, 104]. In the bag-of-words representation they are treated as an independent collection of data points characterizing the depicted scene. Features are typically n-dimensional vectors of unit length, thus points on an n-dimensional hypersphere.

Representing features as 128 dimensional SIFT vectors have shown to be very effective for object recognition problems [95, 105]. Despite the discriminative power of the SIFT features it is computational expensive to represent image features with 128 dimensions. PCA have been applied to SIFT features of different dimensionality [77, 82, 105]. The performance of these descriptors was comparable to the original descriptor, with a reduction to the range of 20 to 36 dimensions.

In this paper we show it to be efficient to reduce the dimensionality of the feature descriptor much further. We use PCA on SIFT features with a reduction in the range of 6 to 15 dimensions including 3 color dimensions. This is a very compact representation compared to the 128 dimensional SIFT features.

**Visual vocabulary (*ii*)** Features have been used for object recognition where similarity between images is found by comparing features directly, see for example [94, 95, 127]. For large image collections this is not feasible, which is the motivation for the bag-of-words representation.

The bag-of-words approach is based on representing features by canonical representative instead of features themselves. Canonical feature representatives can

be viewed as visual words. Each feature in an image is labeled with a reference to a visual word. This way the image is described as a histogram of visual words - a frequency vector, which is a much more compact representation than the individual features.

It was proposed by Sivic and Zisserman [133] to build a image search method based on he idea of text retrieval in large document collections. They demonstrated an efficient algorithm based on inverted files and feature weighing. Feature weights are found from the distribution of visual words in the database images.

Visual words are typically found by clustering of features from a database of images. Often k-means clustering is used for building the visual vocabulary. In k-means the number of clusters is fixed, which can lead to undesirable partitioning of the data. This way natural clusters have the risk of being merged or split, leading to mislabeling of features and loss in discriminative power. Philbin *et al.* [116] did not find other clustering methods than k-means an option, because of the high number and high dimensionality of the features.

The dimensionality reduction applied in this paper, enables us to use a clustering approach inspired by the sequential leader-follower clustering (LF-clustering) [42, 97]. This way we are able to find clusters in a very effective manner, without knowledge about the number of clusters before the clustering takes place.

**Image query (*iii*)** The features in a query image is labeled with a reference to the nearest visual words in the vocabulary. The complexity of this assignment is dependent on the size of the visual vocabulary. It is empirically shown that visual vocabularies have to be relative large to be effective. I.e. in the range from 5K to 1M visual words typically of 128 dimensional SIFT descriptors [77, 109, 116, 126, 143, 155]. Finding the exact nearest neighbor in high dimensions is hard to carry out in less than $O(n^2)$ complexity, so approximations are used.

Nistér and Stewénius [109] came up with the idea of using a tree structure to simultaneously speed up the clustering and the feature labeling. This *Vocabulary Tree* is made by a hierarchical k-means clustering approach. The tree can be used for an approximate feature search which reduces the complexity from $O(n^2)$ to $O(n \log n)$. The approximate feature search is improved in [126] based on the ideas of [13].

Despite the efforts of improving the feature assignment for the query images, the computation is still approximate. The dimensionality reduction of image features that we use enables us to effectively use a kD-tree for feature labeling.

This way we obtain an exact nearest feature assignment.

These modifications simplifies the computation of the bag-of-words representation, in both building, storing and searching the image database. This is even done with an improvement in recognition quality. Effective computation is essential for reaching Internet scale image search.

**Methods** The methods used for improving the efficiency of the bag-of-words approach is described in the following. The improvements are related to reducing the size of the feature descriptors, improving the clustering approach, and improving the feature assignment. The steps in building the bag-of-words model is illustrated in figure 5.1.

**Feature representation** PCA is applied to reduce the dimensionality of the feature vectors. We calculate the eigenvectors for the PCA from all features in our training set. The reduction of the SIFT descriptor is from 128 to between 3 and 12 dimensions.

After dimension reduction we add color to our features. The color is simply the mean RGB value in a $10 \times 10$ pixels patch around the localization of each feature. The color patch is concatenated to the PCA reduced SIFT vector

$$s = [\alpha s_{PCA}, (1 - \alpha) s_{RGB}], \tag{5.1}$$

where $s_{PCA}$ is the PCA reduced SIFT feature, $s_{RGB}$ is the mean RGB values, and $\alpha$ is a weighing parameter. In our experiments we found $\alpha = 0.5$ to be a good choice. So, the final feature gets a size of 6 to 15 dimensions.

We use histogram equalization of the whole image for the R, G, and B bands separately, before we extract the color features. This is done to minimize the effect of change in light between two images of the same scene. Both the PCA-SIFT and color feature are normalized to unit length before concatenation, and normalized again after concatenation. This way we try to avoid non intended bias because of difference in size of the two features.

**Clustering** We investigate the use of LF-clustering [42, 97] as an alternative to k-means. The motivation for this method is to avoid the risk of undesirable partitioning of data caused by a wrong number of clusters. In the LF-clustering

algorithm, data is treated sequentially without any iterative steps. This gives the method a potential for being less computational expensive than k-means.

LF-clustering builds on the idea of defining a minimum dissimilarity between clusters. This is similar to the Mean-Shift clustering algorithm [30], where clusters are found according to a certain bandwidth. But the computational cost of LF-clustering is far less than Mean-Shift.

**Algorithm 3** Hierarchical Leader Follower clustering (LF-clustering)

```
Set data in one cluster: d_cl
Initialize number of levels: n, bandwidth start: b_s, bandwidth end: b_e
Set number of clusters: n_cl = 1
Bandwidth step b_st = (b_S − b_e)/(n + 1)
for i = 1 to n do
  b_now = b_e + (n − i + 1)b_s + b_e
  for j = 1 to n_cl do
    cluster(d_cl, b_now)
    update n_cl, d_cl
    save clusters
  end for
end for
```

The hierarchical LF-clustering algorithm is summarized in Algorithm 12. The procedure of the cluster step is to treat the features sequentially one at a time. The first feature makes up the first cluster. If the next feature is closer to the existing cluster than the defined bandwidth, it will be assigned to this cluster. Otherwise, it is will make a new cluster. When a feature is assigned to an existing cluster, the center of the cluster is updated. We use Euclidean distance, so the cluster center is updated by

$$c_n = \frac{c_o n_o + f_a}{n_n}, \tag{5.2}$$

where $c_n$ is the new cluster center, $c_o$ is the old cluster center, $n_o$ is the number of features in the cluster before the feature $f_a$ is added, and the number of features is updated with one: $n_n = n_o + 1$.

To avoid clusters coming too close together we merge clusters being closer than the bandwidth, and we also set a minimum limit to the number features in a cluster. Clusters with too few point will be merged with the nearest cluster. In both steps the center of the cluster is updated according to equation (5.2).

The speed of this algorithm is dependent on the number of clusters. The ex-

pected complexity is $O(kn)$, where $k$ is the number of clusters and $n$ is the number of points. Small bandwidths and high dimensions result in many clusters and slows down the algorithm. To compensate for that we perform the clustering hierarchically, starting with a large bandwidth and shrink it through the clustering process. We start clustering the whole point set. The obtained clusters are subsequently clustered into new clusters in the following steps. This way we obtain a substantial increase in speed. To avoid clusters getting to close the hierarchical clustering is followed by a merging step. All clusters closer than the bandwidth are removed.

We compare the performance of the LF-clustering to k-means. k-means is also applied hierarchically to obtain increased speed. We also define a minimum number of clusters for k-means clustering. Further clustering is stopped if a cluster has less than a minimum number of points. This can lead to very small clusters, but in our experiments the number of clusters containing very few points is negligible.

With hierarchical clustering we get clusters at each level in the hierarchy. But we only use the clusters found at the last level for the visual vocabulary.

**Feature assignment** Similarity of images are found by comparing frequency vectors of a query image to images in the database. Frequency vectors are made from the frequency of visual words in an image weighed with an entropy weight. The entropy weight is based on the distribution visual words in the database images. We use the same weight as used in [109], which is defined as

$$w_i = log(\frac{N}{n_i}), \tag{5.3}$$

where $w_i$ is the weight of word $i$, $N$ is the total number of images in the database, and $n_i$ is the number of images where word $i$ occurs.

Frequency vectors for database images are given from clustering. But for a query image we need to label the features with visual words, so we need to find the visual words closest to the features in the query image. The dimensionality reduction of the features enables us to effectively use a kD-tree instead. This makes the feature assignment exact [37]. For small dimensions the expected complexity of the kD-tree is $O(n \log n)$, whereas for high dimensions the complexity becomes $O(n^2)$. Therefore, the kD-tree is only applicable with substantial dimensionality reductions.

**Image matching** Frequency vectors are compared using the $L_1$ norm, which is found to be superior to the Euclidean distance just as observed in for example [109]. Our explanation for the $L_1$ norm being superior to the $L_2$ norm is the nature of the problem we are solving. We expect the same features to occur in images from the same scene. So, the frequency vector overlap is a good measure for similarity between images. The $L_1$ norm gives equal weight to the overlapping and non overlapping parts, whereas the $L_2$ norm gives more weight to the non overlapping parts. Before comparing the frequency vectors are normalized to unit length using the $L_1$ norm.

Inverted files are used for fast image retrieval. An inverted file is kept for each visual word in the vocabulary. In the inverted file is a reference to the images in the database containing that word. The frequency vector value of each reference image is stored together with the reference. Image retrieval is obtained by first calculating the frequency vector for the query image using the weights in equation (5.3). With the frequency vector value for the reference images stored in the inverted files, we can compute the $L_1$ norm without looking up the entire frequency vectors of the reference images. This can be done because the $L_1$ norm can be calculated from the frequency vector overlap

$$L_1 = 2 - 2O, \tag{5.4}$$

where $L_1$ is the $L_1$ norm and $O$ is the frequency vector overlap:

$$O = \sum_{i|q_i \neq 0 \wedge d_i \neq 0} \min(q_i, d_i), \tag{5.5}$$

where $q_i$ and $d_i$ is the frequency values of query and database images respectively. Instead of ranking images by smallest $L_1$ norm we rank by largest overlap. Storing a value for each image together with the pointer in the inverted file has a memory cost, which should be viewed in relation to looking up frequency vectors for relevant images.

### 5.2.3 Experiments and Results

The results in this paper are primarily found through empirical studies described in the following section.

**Data set** We use the first 1400 images from the Nistér and Stewénius data set [109, 137] in our experiment. This data set contains a series of 4 images of the same scene, so we have 350 different scenes. We use three of the images from one scene to train the model and the last for testing. The test result is the percentage of the correct images ranked in top 3. This data set is relatively small compared to other experiments, but we found it sufficient for illustrating the effects of our model choices. In future work this should be extended to a larger data set. We also use the preprocessed SIFT features supplied with this data set.

**Experiments** To test the effect of using color added SIFT features and LF-clustering we have made experiments with and without color features and with ordinary k-means and LF-clustering. We have also built a model based on the 128 dimensional SIFT features with and without color to illustrate the performance of the ordinary SIFT features.

**Color added PCA SIFT** These features are made as described in section 5.2.2. We use 3, 8, and 12 dimensional PCA SIFT features, so the resulting color added features are 6, 11, and 15 dimensions. To compare to features without color we take SIFT features reduced with PCA to 6, 11 and 15 dimensions.

**Clustering experiments** For all test sets we have done clustering using k-means and LF-clustering. The number of features from the LF-clustering are in the range from 8,000 to 12,000 clusters, so we have chosen to let the k-means cluster hierarchically to 10 clusters in 4 levels resulting in 10,000 clusters.

**Results** We have summarized the experimental results in table 5.1 and 5.2. The best classification results are obtained with LF-clustering with 15 dimensional color added features. LF-clustering is slightly better than k-means. But the most pronounced effect is the addition of color, which significantly improves the result. It should be noticed, that performance is improved in relation to the full SIFT feature, even with added color. In the full model we also use exact feature assignment for comparison, even though it would not be fast enough for a real application.

Addition of color also gave a ranking of the images that seemed more logical, which is shown in figure 5.2.

| With color | | | |
|---|---|---|---|
| Method | 6 dim | 11 dim | 15 dim |
| k-means | 81.7 | 87.2 | 88.4 |
| LF-clustering | 84.0 | 87.5 | **89.9** |
| Without color | | | |
| Method | 6 dim | 11 dim | 15 dim |
| k-means | 73.3 | 81.6 | 82.2 |
| LF-clustering | 76.3 | 81.9 | 83.9 |

Table 5.1: Model performance with different clustering methods, dimensions of the features, and with and without color. Notice the effect of adding color the SIFT features. The best performance is marked in bold.

| Method | 128 dim | 131 dim (color) |
|---|---|---|
| k-means | 85.7 | 89.6 |

Table 5.2: k-means clustering for the full SIFT features. The results for 128 dimensions is the normal SIFT feature and the 131 dimensions is with color added.

### 5.2.4 Discussion of the paper

Our experiments shows that it is possible to obtain a good recognition performance with the bag-of-words model with a substantial reduction in dimensionality of the features. A reduction to between 6 and 15 dimensions makes it possible to use exact methods for feature assignment, where we use a kD-tree.

For our experiments the best results are obtained with a vocabulary based on 15 dimensional PCA color features using on LF-clustering. It even outperforms the full SIFT features including color, and it is substantially better than the normal 128 dimensional SIFT features. Even with 11 dimensions we obtain better results than with normal SIFT.

Especially the effect of adding color to the PCA reduced SIFT features is very important for the performance. The method for adding color to the features is extremely simple and yet very powerful, which indicates that there are much information in image color. The histogram equalization works well for this data set. But this might be overly simple for images with high variation in viewpoint. Other ways of combining the gradient information from the SIFT features and color information might be even more powerful, and should be investigated further, see for example [1].

Another very important benefit from the information gain from adding color, is the dimension reduction of the features. Low dimensional features makes the model computational less expensive. This is mostly in relation the clustering for building the visual vocabulary, which can be done off line. But on line feature assignment can also be computed faster and have a higher quality, because of the option of doing exact feature search using a kD-tree.

The LF-clustering for building the vocabulary showed a slight improvement in performance. The speed of the hierarchical LF-clustering was about the same as k-means in our implementations, but the LF-clustering has potential for being faster because each point is only treated once in each level in the clustering hierarchy. A requirement is, that the number of clusters does not become too large in one clustering operation.

We did not apply LF-clustering to the 128 dimensional SIFT features, because we found it performing very poorly. The data did not cluster, so we had either one cluster containing all features or all features in their own cluster depending on the bandwidth. For the 128 and 131 dimensional features we chose only to use the k-means clustering.

Our results are good compared to [77, 109] but this might partly be due to the fact that we have only conducted experiments on 1400 images. We also just look at one data set, so for future work the model should be tested on a larger set of data. It is worth noting that the improvement is also shown relative to the normal 128 dimensional SIFT feature, but this observation should also be shown to hold for a larger data set.

We have not included any form of blocking of visual words, as suggested in [133]. In [155] stop words are used in relation to different criteria, but without any clear improvement. We experience that the entropy weighing of the descriptors improves the results. Information about the entropy of the visual words could be included already in building of the visual vocabulary, so the total entropy of the model would be maximized relative to the number of clusters, and we might be able to obtain good performance with a small vocabulary. This is to be done in future work.

A problem of the design of the bag-of-words model is it's static nature. It is not designed for adding or removing images from the database, because it will require a new clustering of all the images in the database, which is very time consuming. In future work it will be relevant to investigate how the method could be designed for the dynamic nature of many databases, for example image databases on the Internet.

### 5.2.5 Conclusion of the paper

We have shown a way to substantially reduce the dimensionality of the SIFT features used in the bag-of-words model through PCA and addition of color to the features. This has enabled us to use a kD-tree for feature labeling, which is an exact method. When 128 dimensional SIFT features are used it is necessary to apply approximations. The reduction in feature dimensions enabled us to apply a clustering algorithm based on the leader follower principle (LF-clustering) where the number of clusters is a result of the clustering. The addition of color and use of LF-clustering are compared to normal PCA SIFT and k-means clustering, which is normally applied in the bag-of-words model. We obtain a clear improvement in performance on a test set containing 1400 images. Especially adding color to the features improves the performance, whereas the clustering algorithm gives a slight improvement. We also get a clear performance improvement compared to the bag-of-words model based on 128 dimensional SIFT features and k-means clustering.

## 5.3 Conclusion on image retrieval

We have presented a paper on image retrieval based on the ideas of a bag-of-visual-words representation of interest point descriptors. In relation to the logTracker system, image retrieval could be a solution for identifying individual wood logs for documenting their origin. There are some issues that have to be investigated for such a system to be feasible. These issues relate to interest point localization, descriptor assignment, and matching of the images. Also practical issues concerning processing and storage of data have to be addressed.

The idea, of the logTracker system, is to place cameras on machines that handle the wood logs. In a normal harvesting and transport operation this concerns the harvester, the forwarder, the road side truck, and can also include transport trucks at the industry, see Section 1.1. The first machine handling the wood logs is the harvester, and the idea is to build a model of each log to store in a database for later recognition, based on images taken from the harvester. New images of the wood logs will be acquired at the later transport operations, and these images should be matched to the database. This will result in practical issues concerning processing, storage, and exchange of data that have to be solved. An issue that can make the recognition problem easier would be to use a GPS position of where the logs are photographed, and this way reduce the number of wood logs from perhaps several thousands to perhaps fifty to one hundred.

The computer vision problems for tracking of wood logs are not solved by the presented method. The core elements of the method are a possible way to approach a solution, but there are still many unanswered questions that have to be addressed. It should be investigated if the existing interest point detectors are good at handling the appearance of wood logs. Descriptor assignment is another issue, and it should be tested if the typical SIFT-like descriptor is sufficient to uniquely characterize wood logs. Otherwise, object specific information could be added to the descriptor, for example by using object specific knowledge. In the process of obtaining the initial characterization of the log, it would be an advantage if the log was seen from all sides. Then an arbitrary view could be matched to the model. This could be solved by having a matching procedure that takes each view into account and the effectiveness of the bag-of-words approach is well suited for this.

The computational issues, which we addressed in our paper, are important, not only for the logTracker system, but also for computer vision in general. Examples are computation time, data storage, etc., especially when large amounts of images should be handled. In this case a reduction from 128 dimensional descriptors to for example 10 dimensions can be what makes a system possible.

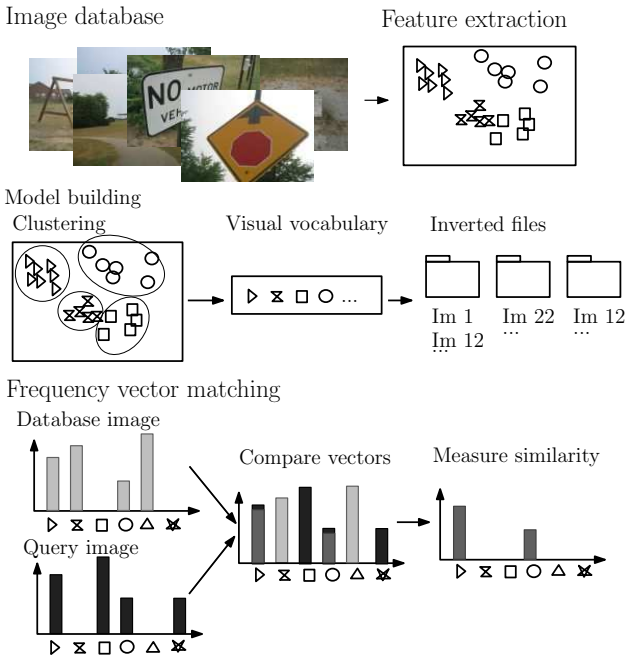



Figure 5.1: Illustration of the bag-of-words model. Features are extracted from the image database. The visual vocabulary is build using clustering, and inverted files is made based on the feature labeling of the database images. Finally feature vectors are used for matching images, with the frequency vector overlap giving the matching score.


Figure 5.2: First 6 images retrieved using 11 dimensional vectors with and without color. With color the highest ranked images looks alike, whereas without color number 5 and 6 are quite different from the rest. This was observed as a general trend.

Chapter 6

# Conclusion of public part

The motivation for this thesis is a computer vision system for improving the efficiency of timber harvesting for the use in a system called logTracker. We have investigated three vital problems in relation to this computer vision system. First, we considered the low-level vision problem of image segmentation, where we focused on textured images. Second, we studied the high-level vision problem of the classification of wood logs. Third, we studied the high-level problem of image retrieval. We have identified solutions within all three problem areas and we will now give a detailed description of the contributions.

Our segmentation procedure deals with the problem of segmenting inhomogeneous and textured images that are difficult to segment because of variability of the image patterns. Our hypothesis is that coherent image parts corresponding to depicted objects can be found using an appropriate image characterization. We verified this by constructing a segmentation procedure based on two elements: a texture characterization and an image segmentation step. The method is based on partition iterative function systems, which were developed for image compression, and we successfully apply them to segmentation. Our method for image characterization is based on a domain kernel set, and the image is characterized by the distribution of mapping these kernels to the image. This characterization shares properties with textons and basic image features, but the novelty is a separation of the characterization to the elements of texture, intensity, variance, and color. The segmentation step is constructed as an

iterative function system, where the function's parameters are estimated from self-similarities in a local neighborhood in the image, and segments are obtained as attractor sets. This has shown to be very effective for obtaining good segmentation results, and the procedure has been validated by segmenting synthetic and real images where we obtained improved results compared to state of the art. Our contribution is a simple, robust, and flexible method for unsupervised image segmentation.

The second study concerns the classification of wood logs to identify tree species from visual surface characteristics. This classification problem is demanding because the wood logs have highly varying visual appearance within the same tree species. Our assumption is that the same species have distinctive texture elements, and we investigate this based on an active appearance model combined with second order texture statistics. This assumption is confirmed through high classification rates: a result which is valuable for this study and shows that it would be reasonable to base tree species classification on a texture characterization. The novelty of the study concerns partly the application, but also the usefulness in precise object localization for extracting essential visual information.

Image retrieval for large scale image databases is the subject for our third study. Time and precision are especially important considerations for image retrieval problems. Our method is based on the bag-of-visual-words approach where an image is characterized as a distribution of descriptor representatives. Our hypothesis is that there is a tradeoff between uniqueness of visual features and performance of the procedure. Higher uniqueness can be obtained by increasing the size of the visual vocabulary and through higher dimensionality of the image descriptors. The consequence will be higher computational costs. The purpose of our investigation is to demonstrate that the uniqueness of the visual words can be retained using low dimensional features and relatively small visual vocabularies. This is achieved through PCA dimensionality reduction of SIFT descriptors and an addition of color. Furthermore, we introduce a simple leader-follower clustering algorithm that also increases the performance. The novelty of our procedure is the very low dimensional descriptors that we obtain significantly lower than previous methods for dimensionality reduction while high performance rates are preserved. This is an important contribution in advancing state of the art within image retrieval. Our hope is that the presented procedures can be a small contribution to advance computer vision and help to build better and more reliable vision systems in the future.

# Bibliography


[1] A. E. Abdel-Hakim and A. A. Farag. Csift: A sift descriptor with color invariant characteristics. *CVPR, 2006.*, 2:1978–1983, 2006.

[2] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, Jun 1994.

[3] A. Akselrod-Ballin, M. Galun, R. Basri, A. Brandt, M.J. Gomori, M. Filippi, and P. Valsasina. An Integrated Segmentation and Classification Approach Applied to Multiple Sclerosis Analysis. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 1122–1129. IEEE Computer Society Washington, DC, USA, 2006.

[4] S. Alexander. *Multiscale Methods in Image Modelling and Image Processin*. PhD thesis, 2005.

[5] S. Arivazhagan and L. Ganesan. Texture classification using wavelet transform. *Pattern Recognition Letters*, 24(9-10):1513–1521, 2003.

[6] G. Aubert, M. Barlaudi, O. Faugeras, and S. Jehan-Besson. Image segmentation using active contours: Calculus of variations or shape gradients. *SIAM Applied Mathematics*, 63:2003, 2003.

[7] J.-F. Aujol, G. Aubert, and L. Blanc-Feraud. Wavelet-based level set evolution for classification of textured images. *IEEE Transactions on Image Processing*, 12(12):1634–1641, 2003.

[8] S. Bagon, O. Boiman, and M. Irani. What is a good image segment? a unified approach to segment extraction. In Forsyth D., P. Torr, and

Zisserman A., editors, *Computer Vision – ECCV 2008*, volume 5305 of *LNCS*, pages 30–44. Springer, 2008.

[9] M. F. Barnsley. *SuperFractals*. Cambridge University Press, New York, NY, USA, 2006.

[10] M. F. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. *Royal Society of London Proceedings Series A*, 399:243–275, June 1985.

[11] R. Bartle and D. Sherbert. *Introduction to Real Analysis*. Wiley, third edition, 2000.

[12] G. Behiels, F. Maes, D. Vandermeulen, and P. Suetens. Evaluation of image features and search strategies for segmentation of bone structures in radiographs using active shape models. *Medical Image Analysis*, 6(1):47–62, 2002.

[13] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Computer Vision and Pattern Recognition, 1997.*, pages 1000–1006, 1997.

[14] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.

[15] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *In NIPS*, pages 831–837, 2000.

[16] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[17] M. H. Bharati, J. J. Liu, and J. F. MacGregor. Image texture analysis: methods and comparisons. *Chemometrics and Intelligent Laboratory Systems*, 72(1):57–71, 2004.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[19] O. Boiman and M. Irani. Detecting irregularities in images and in video. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 1:462–469 Vol. 1, 2005.

[20] O. Boiman and M. Irani. Similarity by composition. *NIPS*, 2006.

[21] O. Boiman and M. Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[22] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, page 46, 2004.

[23] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Computer Vision - ECCV 2002 : 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002*, pages 639–641, 2002.

[24] P. Brodatz. *Textures; a photographic album for artists and designers*. 1966.

[25] J. M. Carstensen. *Description and Simulation of Visual Texture*. PhD thesis, Institure of Mahematical Statistics and Operations Research, Technical University of Denmark, Kgs. Lyngby, 1992.

[26] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[27] P. Chang and J. Krumm. *Object Recognition with Color Cooccurrence Histogram*. 1999.

[28] Y. Chen and C. I. Chang. A new application of texture unit coding to mass classification for mammograms. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, 2004.

[29] M. Chica-Olmo and F. Abarca-Hernández. Computing geostatistical image texture for remotely sensed data classification. *Computers and Geosciences*, 26(4):373–383, 2000.

[30] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[31] T. F. Cootes and C. J. Taylor. Statistical models of appearance for medical image analysis and computer vision, 2004. In Proc. SPIE Medical Imaging.

[32] M. Crosier and L. D. Griffin. Texture classification with a dictionary of basic image features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7, 2008.

[33] A. B. Dahl and H. Aanaes. Effective image database search via dimensionality reduction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[34] A. B. Dahl, H. Aanaes, R. Larsen, and B. K. Ersboll. Classification of biological objects using active appearance modelling and color cooccurrence matrices. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4522 LNCS:938–947, 2007.

[35] A. B. Dahl, P. Bogunovich, A. Shokoufandeh, and H. Aanæs. Texture segmentation from context and contractive maps. Submitted for publication, 2009.

[36] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. A. N. J. Koenderink. Reflectance and Texture of Real-World Surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

[37] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[38] M. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner. Manyto -many feature matching using spherical coding of directed graphs, 2004.

[39] S. Dickinson, L. Bretzner, Y. Keselman, A. Shokoufandeh, and M. F. Demirci. Object recognition as many-to-many feature matching. *International Journal of Computer Vision*, 69(2):203–222, 2006.

[40] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. From volumes to views: an approach to 3-D object recognition. *CVGIP: Image Understanding*, 55(2):130–154, 1992.

[41] R. Distasi, M. Nappi, and M. Tucci. Fire: fractal indexing with robust extensions for image databases. *IEEE Transactions on Image Processing*, 12(3):373–384, 2003.

[42] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[43] D. P. Dykstra, G. Kuru, R. Taylor, R. Nussbaum, W. B. Magrath, and J. Story. *Technologies for Wood Tracking: Verifying and Monitoring the Chain of Custody and Legal Compliance in the Timber Industry*. World Bank, 2003.

[44] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Face recognition using active appearance models. *Computer Vision - ECCV'98. 5th European Conference on Computer Vision. Proceedings*, pages 581–95 vol.2, 1998.

[45] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–451, 2004.

[46] J. Fagertun, D. D. Gomez, B. K. Ersbøll, and R. Larsen. A face recognition algorithm based on multiple individual discriminative models. In Søren I. Olsen, editor, *DSAGM 2005*, DIKU Technical report 2005/06, pages 69–75, Universitetsparken 1, 2100 København Ø, aug 2005. DIKU, University of Copenhagen.

[47] K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications. 2nd Edtion.* 2003.

[48] M. F. A. Fauzi and P. H. Lewis. Automatic texture segmentation for content-based image retrieval application. *Pattern Anal. & App.*, V9(4):307–323, November 2006.

[49] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. *ICCV 2005.*, 2:1816–1823, 2005.

[50] R. D. Fernald. The Evolution of Eye. *Brain behavior and evolution*, 50(4):253–259, 1997.

[51] R. D. Fernald. Evolution of eyes. *Current Opinion in Neurobiology*, 10(4):444–450, 2000.

[52] C. Fischer, F. Aguilar, P. Jawahar, and R. Sedjo. Forest Certification: Toward Common Standards. 2005.

[53] Y. Fisher. *Fractal Image Compression - Theory and Application.* Springer-Verlag, New York, 1994.

[54] I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Biological Cybernetics*, 61(2):103–113, 1989.

[55] P. E. Forssen. Maximally Stable Colour Regions for Recognition and Matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8, 2007.

[56] B. Forte and E. Vrscay. Inverse problem methods for generalized fractal transforms, 1998.

[57] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[58] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[59] C. Gamborg and J. B. Larsen. 'Back to nature' - a sustainable future for forestry? *Forest Ecology and Management*, 179(1-3):559–571, 2003.

[60] B. V. Ginneken, M. B. Stegmann, and M. Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: A comparative study on a public database. *Medical Image Analysis*, 10(1):19–40, 2006.

[61] L. van Gool, T. Kadir, F. Schaffalitzky, J. Matas, A. Zisserman, C. Schmid, T. Tuytelaars, and K. Mikolajczyk. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[62] D. W. Hansen, M. Nielsen, J. P. Hansen, A. S. Johansen, and M. B. Stegmann. Tracking eyes using shape and appearance. In *IAPR Workshop on Machine Vision Applications - MVA*, pages 201–204, dec 2002.

[63] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[64] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans Syst Man Cybern*, SMC-3(6):610–621, 1973.

[65] K. Haris, SN Efstratiadis, and N. Maglaveras. Watershed-based image segmentation with fast region merging. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 338–342, 1998.

[66] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[67] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[68] E. Hayman, B. Caputo, M. FritZ, and J. O. Eklundh. on the Significance of Real-World Conditions for Material Classification. In *ECCV*. Springer, 2004.

[69] J. H. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *IEEE ECCV*, May 2006.

[70] A. Herbulot, S. Jehan-Besson, M. Barlaud, and G. Aubert. Shape gradient for image segmentation using information theory. *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3:iii–21, 2004.

[71] B.-H. Hong, S. Soatto, K. Ni, and T. Chan. The scale of a texture and its application to segmentation. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[72] Y. Horita, T. Murai, and M. Miyahara. Region segmentation using k-mean clustering and genetic algorithms. *Proceedings of 1st International Conference on Image Processing*, 3:1016–1020 vol.3, 1994.

[73] B. K. P. Horn. *Robot vision*. MIT Press, Cambridge, MA, USA, 1986.

[74] N. Houhou, J.P. Thiran, and X. Bresson. Fast texture segmentation model based on the shape operator and active contour. In *CVPR*, pages 1–8, 2008.

[75] X. Huang, Z. Qian, R. Huang, and D. Metaxas. Deformable-model based textured object segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3757 LNCS:119–135, 2005.

[76] A. E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IP*, 1(1):18–30, January 1992.

[77] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *IEEE Conference on Computer Vision & Pattern Recognition*, june 2007.

[78] S. Jehan-Besson, M. Barlaud, and G. Aubert. Dream2s: deformable regions driven by an eulerian accurate minimization method for image and video segmentation. *International Journal of Computer Vision*, 53(1):45–70, 2003.

[79] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[80] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. 2004.

[81] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[82] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *CVPR 2004.*, 02:506–513, 2004.

[83] T. Knudsen and A. A. Nielsen. Detection of buildings through multivariate analysis of spectral, textural, and shape based features. In *IGARSS*, volume 5, 2004.

[84] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367–375, 1987.

[85] P. M. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *CVPR*, pages 18–25, Washington, DC, USA, 2005. IEEE Computer Society.

[86] R. Lakmann. Barktex texture database, 1998.

[87] J. B. Larsen and A. B. Nielsen. Nature-based forest managementWhere are we going? Elaborating forest development types in and with practice. *Forest Ecology and Management*, 238(1-3):107–117, 2007.

[88] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. 2003.

[89] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[90] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. *IEEE ICCV*, 2:1010, 1999.

[91] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3954 LNCS:581–594, 2006.

[92] M. Lillholm and L. D. Griffin. Novel image feature alphabets for object recognition. In *ICPR*, 2008.

[93] J. Lounasvuori, S. Ibrahim, and S. Ali. Tracking the wood. 2006.

[94] D. G. Lowe. Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2:1150–1157, 1999.

[95] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[96] J. H. Luo and C. C. Chen. Singular value decomposition for texture analysis. In *Proceedings of SPIE*, volume 2298, page 407. SPIE, 1994.

[97] R. López de Màntaras and J. Aguilar-Martín. Self-learning pattern classification using a sequential clustering technique. *Pattern Recognition*, 18(3-4):271–277, 1985.

[98] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *IEEE ICCV*, pages 918–925, 1999.

[99] B. Mandelbrot. How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science*, 156(3775):636, 1967.

[100] B. S. Manjunath and W. Y. Ma. Texture Features for Browsing and Retrieval of Image Data. *PAMI*, 18(8), 1996.

[101] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, July 2001.

[102] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, London, 2002.

[103] A. Materka and M. Strzelecki. Texture Analysis Methods–A Review. 1998.

[104] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[105] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[106] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. 2009.

[107] P. Mörters. Fractal geometry - from self-similarity to brownian motion. *http://people.bath.ac.uk/maspm/fract.ps*.

[108] M. Nappi, G. Polese, and G. Tortora. First: Fractal indexing and retrieval system for image databases. *Image and Vision Computing*, 16(14):1019–1031, 1998.

[109] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, June 2006.

[110] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[111] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-9(1):62–66, 1979.

[112] C. Palm. Color texture classification by integrative co-occurrence matrices. *Pattern Recognition*, 37(5):965–976, 2004.

[113] C. Palm and T. M. Lehmann. Classification of color textures by gabor filtering. *Machine Graphics and Vision*, 11(2/3):195–220, 2002.

[114] N. Paragios and R. Deriche. Geodesic active regions: A new framework to deal with frame partition problems in computer vision. *Journal of Visual Communication and Image Representation*, 13(1-2):249–268, 2002.

[115] H. O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals: New Frontiers of Science*. Springer, 2004.

[116] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR 2007.*, pages 1–8, 2007.

[117] M. Pi, M. Mandal, and A. Basu. Image retrieval based on histogram of new fractal parameters. *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, 3:III–585, 2003.

[118] A. Porebski, N. Vandenbroucke, and L. Macaire. Iterative Feature Selection for Color Texture Classification. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 3, 2007.

[119] A. Porebski, N. Vandenbroucke, and L. Macaire. Haralick feature extraction from lbp images for color texture classification. *2008 First Workshops on Image Processing Theory, Tools and Applications*, pages 1–8, 2008.

[120] A. R. Rao. *A taxonomy for texture description and identification.* 1990.

[121] M. Rousson and R. Deriche. A variational framework for active and adaptative segmentation of vector valued images. *Proceedings Workshop on Motion and Video Computing (MOTION 2002)*, pages 56–61, 2002.

[122] W. Rudin. *Principles of mathematical analysis International series in pure and applied mathematics.* McGraw-Hill, third edition, 1976.

[123] W. Rudin. Real and Complex Analysis (3rd), 1986.

[124] C. Sagiv, N. A. Sochen, and Y. Y. Zeevi. Integrated active contours for texture segmentation. *IEEE Transactions on Image Processing*, 15(6):1633–1646, 2006.

[125] F. Schaffalitzky and A. Zisserman. Multi-view Matching for Unordered Image Sets, or" How Do I Organize My Holiday Snaps?". In *Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 414–431. Springer-Verlag London, UK, 2002.

[126] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. *CVPR 2007.*, pages 1–7, 2007.

[127] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[128] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

[129] L. Shafarenko, M. Petrou, and J. Kittler. Automatic watershed segmentation of randomly textured color images. *Image Processing, IEEE Transactions on*, 6(11):1530–1544, 1997.

[130] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I, 2001.

[131] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[132] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[133] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. *ICCV 2003.*, (vol.2):1470–7 vol.2, 2003.

[134] K. Skoglund. The lars-en algorithm for elastic net regression - matlab implementation, 2006. kas@imm.dtu.dk.

[135] M. B. Stegmann. Object tracking using active appearance models. In Søren I. Olsen, editor, *Proc. 10th Danish Conference on Pattern Recognition and Image Analysis*, volume 1, pages 54–60, Copenhagen, Denmark, jul 2001. DIKU.

[136] M. B. Stegmann. The AAM-API, 2003. Platform: MS Windows.

[137] H. Stewénius and D. Nister. Recognition benchmark images (http://www.vis.uky.edu/štewe/ukbench/), 2006.

[138] H. Sun, J. Yang, and M. Ren. A fast watershed algorithm based on chain code and its application in image segmentation. *Pattern Recognition Letters*, 26(9):1266–1274, 2005.

[139] R. Susomboon, D .S. Raicu, and J. D. Furst. Pixel-Based Texture Classification of Tissues in Computed Tomography. In *CTI Research Symposium*, 2006.

[140] Ojala T., Pietikäinen M., and Mäenpää T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. 2002. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971 - 987.

[141] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[142] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

[143] B. Triggs, F. Jurie, and E. Nowak. Sampling strategies for bag-of-features image classification. *Lecture Notes in Computer Science*, 3954:490–503, 2006.

[144] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, 1991.

[145] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 412–425, 2000.

[146] T. Tuytelaars and L. Van Gool. Matching Widely Separated Views Based on Affine Invariant Regions. *International Journal of Computer Vision*, 59(1):61–85, 2004.

[147] M. Unser. Texture classification and segmentation using wavelet frames. *Image Processing, IEEE Transactions on*, 4(11):1549–1560, 1995.

[148] L. J. Van Gool, T. Moons, and D. Ungureanu. Affine/Photometric Invariants for Planar Intensity Patterns. In *Proceedings of the 4th European Conference on Computer Vision-Volume I-Volume I*, pages 642–651. Springer-Verlag London, UK, 1996.

[149] M. Varma and R. Garg. Locally invariant fractal features for statistical texture classification. pages 1–8, 2007.

[150] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *PAMI*, 13(6):583, 1991.

[151] E. Vrscay. Mathematical theory of generalized fractal transforms and associated inverse problems, 1996.

[152] B. Wohlberg and G. De Jager. A review of the fractal image coding literature. *Image Processing, IEEE Transactions on*, 8(12):1716–1729, 1999.

[153] Y. Xu, H. Ji, and C. Fermuller. A Projective Invariant for Textures. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 2*, pages 1932–1939. IEEE Computer Society Washington, DC, USA, 2006.

[154] Y. Xu and J. Wang. Fractal coding based image retrieval with histogram of collage error. *Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005.*, pages 143–146, 2005.

[155] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. *Proceedings of the ACM International Multimedia Conference and Exhibition*, pages 197–206, 2007.

[156] Kouzani A. Z. Classification of face images using local iterated function systems. *Machine Vision and Applications*, 19(4):223–248, July 2008.

[157] G. Zeng and L. Van Gool. Multi-label image segmentation via point-wise repetition. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[158] A. Zhang, B. Cheng, and R. Acharya. An approach to query-by-texture in image database systems. *Proceedings of the SPIE - The International Society for Optical Engineering*, 2606:338–349, 1995.

[159] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu. What are textons? *International Journal of Computer Vision*, 62(1-2):121–143, 2005.

[160] S. C. Zhu, T. S. Lee, and A. L. Yuille. Region competition: unifying snakes, region growing, energy/bayes/mdl for multi-band image segmentation. *Proceedings. Fifth International Conference on Computer Vision (Cat. No.95CB35744)*, pages 416–423, 1995.

[161] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 67(2):301, 2005.