Illumination Invariant Classification of 3D Surface Textures

Andreas Penirschke

March 2002

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Dr. Mike Chantler for his ideas, support and guidance throughout this project.

I also would like to thank Prof. Dr.-Ing. Eberhard Hänsler and Dipl.-Ing. Maximilian Gauger from Darmstadt University of Technology and Dr. Mike Chantler from Heriot-Watt-University for making my projekt possible.

I am particularly grateful to all the people in the Texture Lab for their help and patience in the lab and their entertainment during free time, especially for the short trips to Durham, St. Andrews and Loch Lohmont.

I am especially indebted to my girlfriend Carmen for all her support and visits in Edinburgh.

Special thanks to all my friends in Edinburgh and Germany and to my family for their support and encouragement.

Abstract

Changes in the angle of illumination incident upon a 3D surface texture can significantly alter its appearance. Such variations affect texture feature images can dramatically increase the failure rates of texture classifiers. Changes in illumination direction causes texture clusters to describe a two dimensional trigonometric function in feature space, which is dependent on illumination tilt and slant angle.

Starting from a sinosodial prediction for texture features we analytically derive two different classification algorithms. First a classifier is introduced, which is robust to changes in illumination tilt direction. The classifier is tested with both, simulations and experiments. Finally we introduce a classifier, which is robust to changes in illumination direction.

Chapter 1

Introduction

1.1 Motivation

This report is concerned with the application of texture analysis to classify rough textured surfaces. Side lighting which is used in many texture databases, enhances the appearance of the surface texture but produces an image, which is a directionally filtered version of the surface height function. Except for [3] and [7], the effect due to variations in illumination direction is neglected in the published literature. They pointed out, that a classifier, which has been trained using images of surfaces at a given illumination direction, which classifies well under these conditions, may not be able to accurately classify the same surfaces illuminated under different directions.
In this report a new model for the behaviour of illuminated surfaces in feature space, which was proposed by [4] and investigated by [14] is used for illumination invariant classification of surfaces. Laws' masks as well as Gabor filters are used to obtain features of the images.
Our aim is to develop a classifier, which is robust to the change of illumination direction. First we develop a sub-problem, the illumination tilt[1] independent classification, to validate the model for tilt invariance. The classification is verified with both, simulations and experiments. The next step is to develop a classifier, which is invariant to the illumination direction. This classification is also verified by simulations and experiments. The final step is to compare both classifiers in order to misclassification and accuracy of illuminant direction estimation.

[1] In the axis system we use, the camera is parallel to the z-axis, illuminant tilt is the angle the illuminant vector makes with the x-axis when it is projected into the xy-plane, and illumination slant is the angle that the illuminant vector makes with the camera axis.

1.2 Area of Research

This work uses a linear reflection model based on Kube and Pendland's linear model, investigated by [4] and [14]. A classifier can be divided in measurement, feature extraction and discrimination. Measurement is the transformation of an illuminated surface to its image. Feature extraction is the extraction of information from the image. The discrimination is described by an algorithm, which uses the extracted features to assign the image to a particular class.

To extract information from an image we use Laws' masks and Gabor filters. Laws [11] formulated "texture energy measures" to extract information from an image. The two-dimensional generalisation of Gabor filters was first defined by Daugman [6]. Randen [12] compared filters for texture classification and referred to Jain and Faarrokhina [9], which showed that Gabor filters can be applied successfully to texture discrimination.

Kube and Pendland [10] presented a linearized model of the Lambertian reflection model, which is valid for many natural surfaces. Chantler [4] used this model to investigate the response of textures to illumination rotation. He showed that the behaviour of texture features may be modelled as a sinosodial function of the illumination tilt angle(τ).

We use this model to develop a classification method, which is robust to changes in illumination direction.

1.3 Dissertation Organisation

The dissertation is organised into six chapters. **Chapter 2** contains the background theory for this work. First we describe the Two-Dimensional Fourier Transformation. Then a short introduction into filters used in image processing is given. Also the Lambertian reflectance model and Kube and Pendland's linear model are described. These describe the connection between the surface and its illuminated image. The chapter ends with an introduction in texture feature generation followed by the model used for illumination invariant classification in this work. **Chapter 3** presents real surfaces whereas **Chapter 4** describes four synthetic surfaces, which are used in this report. Starting with a detailed introduction into classification **Chapter 5** presents the classification model and the results for illuminant tilt invariant and illuminant direction invariant classification of textures. Fi-

nally **Chapter 6** summarizes the results and conclusions of this work and draws together conclusions for further investigations.

Chapter 2

Background Theory

The aim of this chapter is to introduce the basics, required to understand this project. First we present the Two-Dimensional Fourier Transform, which connects the spatial domain to the frequency domain. In the next section we give a short overview about two dimensional filters and masks used in texture analysis. Gabor filters and Laws' masks are choosen for this project so that they are discussed in detail. We introduce the Lambertian reflectance model, which is a diffuse reflection model that describes the influence of illumination to the image. Assuming distant point sources for illumination and no self-shadowing of the surface itself, we reach the linearised reflection model derived by Kube and Pendland [10].

2.1 Two-Dimensional Fourier Transform

This report is concerned with analysing and processing textured images. A very important tool, which connects the spatial domain to the frequency domain is the Fourier Transform. The two dimensional continuous form is given in the next equation pair.

$$
\begin{aligned}
F(\omega_1, \omega_2) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t_1, t_2) e^{-j\omega_1 t_1} e^{-j\omega_2 t_2} dt_1 dt_2 \\
f(t_1, t_2) &= \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\omega_1, \omega_2) e^{+j\omega_1 t_1} e^{+j\omega_2 t_2} d\omega_1 d\omega_2
\end{aligned}
$$

where

$f(t_1, t_2)$ is the signal in spatial domain and
$F(\omega_1, \omega_2)$ is the signal in frequency domain.

This transformation cannot be used for numeric calculations, because of the infinite integrals, which can only approximately numerically calculated. The Discrete Fourier Transform is appropriate to represent the Fourier Transform for digital computer realization, because it is discrete and of finite length in both the spatial and the frequency domain. Its two dimensional realization, the Two-Dimensional Discrete Fourier Transform (TDDFT), is shown in the next two equations.

$$
\begin{aligned}
F(u,v) &= \frac{1}{M\,N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f\,(x,y)e^{-j\frac{2\pi}{M}u\,x}e^{-j\frac{2\pi}{N}v\,y} \\
f(x,y) &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F\,(u,v)e^{+j\frac{2\pi}{M}u\,x}e^{+j\frac{2\pi}{N}v\,y}
\end{aligned}
$$

where

$f\,(x,y)$	is the signal in spatial domain,	
$F\,(u,v)$	is the signal in frequency domain	and
$M \times N$	is the size of the image.	

In image processing the TDDFT is used pixelwise. If the horizontal distance between two pixels is given by Δx and the vertical by Δy the interval between two successive frequencies is:

$$\Delta u = \frac{1}{M\cdot\Delta x} \quad \text{and} \quad \Delta v = \frac{1}{N\cdot\Delta x}\ .$$

Two very useful properties are:

Linearity	$a\cdot p(x,y) + b\cdot q(x,y)$	$\longleftrightarrow$	$a\cdot P(u,v) + b\cdot Q(u,v)$
Multiplication	$p(x,y)\otimes q(x,y)$	$\longleftrightarrow$	$P(u,v)\cdot Q(u,v)$

The symmetrical relationchips of the Fourier Transform can be used to simplify the calculation if the signal to be transformed is known.
Important relationships are:

Spatial domain	**Frequency domain**
Real	Real even & imaginary odd
Imaginary	Real odd & imaginary even
Real and assymetrical	Complex and hermetian
Imaginary and assymetrical	Complex and antihermetian

and vice versa.
More properties and an investigation in detail can be found in [1].

2.2 Filter

In image processing filters are normally used to suppress either high frequencies (lowpass filters), or low frequencies (highpass filters) in the image. Also bandpass filters are used. Filters can be used to reduce the noise in the image, to 'blur' images (remove detail), to highlight detail and so on. For classification filters are used to extract features out of the image (to get the feature a postprocessing stage as presented in section 5.1 on page 30 is needed). Beside nonfiltering approaches for feature generation like co-occurrence (statistical) and autoregressive (model based) features, linear and nonlinear filters are used. A detailed discussion about filters for texture classification can be found in [12]. Linear filters which are very popular in texture analysis are *Gabor* filters, *Laws'* masks, *Wedge* and *Ring* filters, *Wavelet* transforms etc. Gabor filters and Laws' masks are extensively used in this report, so that we discuss them in detail.

2.2.1 Laws' Masks

One of the first filtering approaches for texture identification was the use of a bank of separable filters to get a two dimensional frequency band split. It was presented by Laws [11], who used three one dimensional filters to generate two dimensional filter masks (*Laws'* masks).
The one dimensional filters are:

$L3 = (1, 2, 1)$	Level detection,	
$E3 = (-1, 0, 1)$	Edge detection	and
$S3 = (-1, 2, -1)$	Spot detection.	

Before illustrating the two dimensional approach of Laws, we will investigate the one dimensional filters in detail. All these filters are used pixelwise. (Note we investigate only a few specific filters to describe the groups of filters.)

Level detection filters
Level detection filters are lowpass filters. The simplest filter is the mean filter, which is used to reduce noise in an image. It simply calculates the

average value of its neighbour pixels including itself. Another level detection filter is the Gaussian smoothing filter. It is similar to the mean filter, but instead of constant filter values the filter shape is gaussian ("bell-shaped").

Edge detection filters
Edge detection filters are highpass filters. They are used to find the boundaries between two areas of different levels.

Spot detection filters
Spot detection filters are bandpass filters.

Laws convolved these filters with each other, to provide a set of symmetric and antisymmetric centre weighted masks with all but the level filters being zero sum. He convolved these filters in turn with transponses of each other to give various sizes of square masks. The most usefull filters he found are

$$E5L5 = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$R5R5 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

$$L5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

$$E5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix}$$

The letters L, E, S and R (Ripple detection) indicates the one dimensional filters, which where convolved to generate the two dimensional filters. L5, S5 and R5 are lowpass, bandpass and highpass filters with zero phase whereas

E5 is a bandpass filter that causes a phase shift of 90°. In the following the generation of the two dimensional masks is shown for $E5L5$. The basic filters for $E5L5$ are Edge detection and Level detection.
They are

$L3 = (1, 2, 1)$ Level detection and
$E3 = (-1, 0, 1)$ Edge detection.

Its magnitude frequency responses are:

$$\begin{aligned} |H_{L3}(\omega)| &= \left|e^{-j\omega} + 2 + e^{j\omega}\right| = 2(1 + \cos(\omega)) \\ |H_{E3}(\omega)| &= \left|-e^{-j\omega} + e^{j\omega}\right| = 2\sin(\omega) \end{aligned}$$

Convolving the filters with itself, we get:

$$\begin{aligned} L5 &= L3 * L3 = (1, 2, 1) * (1, 2, 1) = (1, 4, 6, 4, 1) \\ E5 &= E3 * E3 = (-1, 0, 1) * (-1, 0, 1) = (-1, -2, 0, 2, 1) \end{aligned}$$

For the magnitude frequency responses, we get:

$$\begin{aligned} |H_{L5}(\omega)| &= |2(1 + \cos(\omega)) \cdot 2(1 + \cos(\omega))| = 4(1 + |\cos(w)|) \\ |H_{E5}(\omega)| &= |2\sin(\omega) \cdot 2\sin(\omega)| = 4\,|sin(\omega)(1 + cos(\omega))| \end{aligned}$$

The convolution of the transposed Level detection filter and the Edge detection filter is shown in the next equation.

$$\begin{aligned} E5L5 &= (E5)^T * L5 \\ &= \begin{pmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{pmatrix} * (1\,,\,4\,,\,6\,,\,4\,,\,1) = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \end{aligned}$$

The magnitude frequency responses for the Laws' filter masks are:

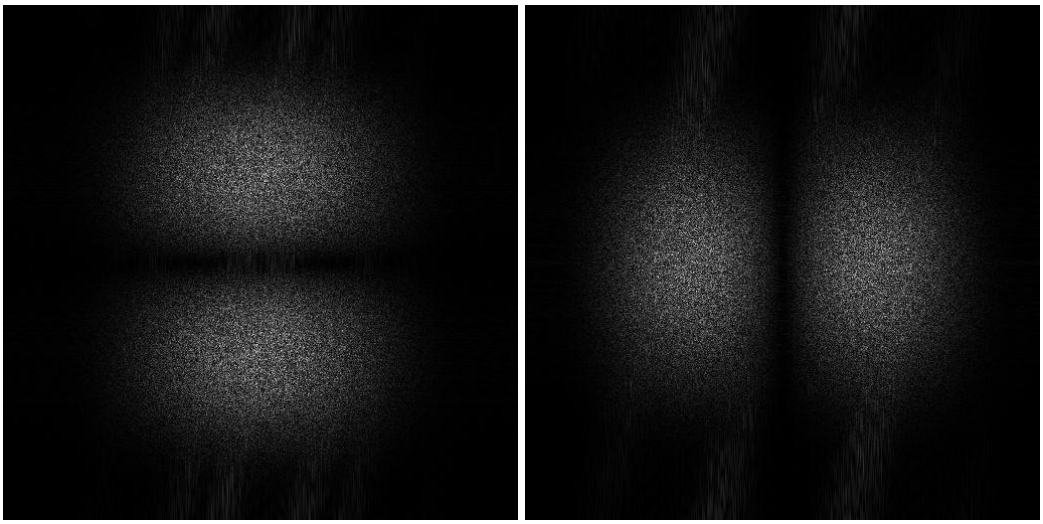


Figure 2.1: White noise filtered with two dimensional Laws' masks are shown in frequency domain. On the left side the mask *E5L5* is used and on the right side the transposed mask *L5E5* is used.

$$
\begin{aligned}
|H_{E5L5}(\omega_1,\omega_2)| &= |H_{E5}(\omega_2)\cdot H_{L5}(\omega_1)| \\
&= 4\sin(\omega_2)(1+\cos(\omega_2))\cdot 4(1+\cos(\omega_1)^2) \\
|H_{R5R5}(\omega_1,\omega_2)| &= |H_{R5}(\omega_2)\cdot H_{R5}(\omega_1)| \\
&= 4(1+\cos(\omega_2))^2\cdot 4(1+\cos(\omega_1)^2) \\
|H_{L5S5}(u,v)| &= |H_{L5}(\omega_2)\cdot H_{S5}(\omega_1)| \\
&= 4\sin^2(\omega_2)\cdot 4(1-\cos(\omega_1)^2) \\
|H_{E5S5}(u,v)| &= |H_{E5}(\omega_2)\cdot H_{S5}(\omega_1)| \\
&= 4\sin^2(\omega_2)\cdot 4\sin(\omega_1)(1+\cos(\omega_1)^2)
\end{aligned}
$$

The Laws' masks are normally used in pairs of a mask and its transposed mask. Transposition means that the filter matrix is transposed. For the $E5L5$ mask the transposed mask is $L5E5$. The magnitude frequency response is the same for both masks, but there is a shift of 90° in the filter direction. Figure 2.1 shows white noise filtered with an E5L5 Laws' mask on the left and a L5E5 Laws' mask on the right.

2.2.2 Gabor Filter

Gabor filters are widely used in texture analysis. They became very popular because the human vision system is also thought to use banks of directional bandpass filters such as Gabor filters. Gabor filters are defined having a Gaussian envelope modulated by a complex exponential. In the one dimensional case the filter is expressed in spatial and frequency domain by equations 2.1 and 2.2. It is obvious that the filter can be described by the extension of the Gaussian envelope σ_f, the center frequency ω_0 and the phase displacement ϕ.

$$g(x) = exp[\frac{-x^2}{2\sigma_f^2}]\, exp[j2\pi\omega_0\, x + \phi] \tag{2.1}$$

$$G(\omega) = exp[-2\pi\sigma_f^2\,(\omega - \omega_0)^2] + exp[-2\pi\sigma_f^2\,(\omega - \omega_0)^2] \tag{2.2}$$

The equivalent two dimensional form, first introduced by [6] is described in the next two equations.

$$g(x,y) = exp[-\frac{1}{2}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2})]\, exp[j2\pi u_0 x] \tag{2.3}$$

$$G(u,v) = A(exp[-\frac{1}{2}(\frac{(u-u_0)}{\sigma_u^2} + \frac{v^2}{\sigma_v^2})] + exp[-\frac{1}{2}(\frac{(u+u_0)}{\sigma_u^2} + \frac{v^2}{\sigma_v^2})]) \tag{2.4}$$

where $\sigma_u = \frac{1}{2\pi\sigma_x}$ and $\sigma_v = \frac{1}{2\pi\sigma_y}$.

As with the one dimensional case, a two dimensional filter is completely described by its center frequencies u_0, v_0 and the standard deviations σ_u and σ_v. Gabor filters can also be described by their centre frequency and their directionality, which is defined as the direction of the maximum intensity. We use the notation $typeF\Omega F\Theta$ to denote a Gabor filter with a centre frequency of Ω cycles per image-width and a direction of Θ degrees. The type of the filter is either complex (*com*), real (*real*) or isotropic (*iso*).

Figure 2.2 (on the left side) shows the filter *comF20A135* in frequency domain. A Gabor filter (colours inverted) is shown on the right side of figure 2.2. Here the definition of center frequency Ω, direction Θ and the nyquist frequency fc of the filter is visualized. For classification a filter with a good spatial resolution and a good spectral resolution is needed. The resolution

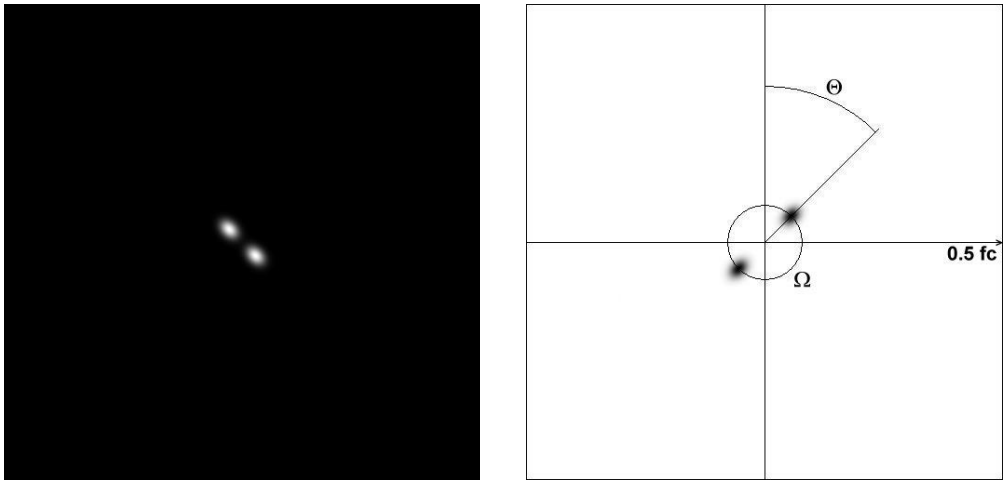


Figure 2.2: a) *comF20A135* (real part) b) Visualization of filter parameters in frequency domain

for Gabor filters in both, spatial and frequency domain, is dependent on the standard deviation of the Gaussian envelope. For a one dimensional filter in spatial domain the standard deviation is found in the denominator of the exponential whereas for in the frequency domain it is found in the enumerator. This causes a trade-off between spectral and spatial resolution. A high resolution in spatial domain causes a low resolution in frequency domain and vice versa. This is also true in the two dimensional case. The Gaussian envelope in the spectral domain is dependent on the reciprocal of the standard deviation in spatial domain. Nevertheless Gabor filters exhibit good localization properties in spatial domain as well as in frequency domain.

A simplification of the complex Gabor filter is the real Gabor filter, where only the real part of the complex transfer function is used. The next equation shows the transfer function of a real Gabor filter.

$$G(u,v) = Aexp[-\frac{1}{2}(\frac{(u-u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2})] + exp[-\frac{1}{2}(\frac{(u+u_0)^2}{\sigma_v^2})]$$

The advantage of much less computation is bought by the disadvantage that the trade-off between the spatial variance and the bandwidth is no longer optimal. The transfer function of the isotropic Gabor filter is given in the following equation:

$$H(\omega) = exp(-\frac{(\omega-\omega_0)^2}{\sigma^2})$$

A investigation of isotropic and real Gabor filters can be found in [14].

2.2.3 Filter Bank

In texture analysis, filters are used to determine features from images. The number of features which are needed depends on the pre-knowledge of the textures. To solve a particular problem, a small set of features might be enough to describe a particular texture, but for classification of a large set of textures without any pre-knowledge a large bank of filters is needed. The selection of the filters is also dependent on the classification problem. Without any pre-knowledge the filters of a filter bank are choosen to cover a wide range of different information of the textures. The Gabor filter banks which are used in this report can be found in table 2.1. The Gabor filter bank 6 for example contains the filters *comF20A0*, *comF20A90*, *comF30A45*, *comF30A135*, *comF40A0* and *comF40A90*. The mixed filter banks are shown in table 2.2. The covered area in frequency domain for each filter bank is presented in appendix B.3 on page 83.

filter	**Gabor filter bank**						
	12	10	8	6	4	3	2
comF20A0	X	X	X	X	X	X	X
comF20A45	X	X	X				
comF20A90	X	X	X	X	X	X	X
comF20A135	X	X	X				
comF30A0	X						
comF30A45	X	X	X	X	X	X	
comF30A90	X						
comF30A135	X	X	X	X	X		
comF40A0	X	X	X	X			
comF40A45	X	X					
comF40A90	X	X	X	X			
comF40A135	X	X					

Table 2.1: Gabor filter banks used for classification as well for real as for synthetic surfaces

filter	Mixed filter bank					
	9	6	5	4	3	2
comF25A0	X		X	X	X	X
comF25A45	X	X	X	X		
comF25A90	X		X	X	X	X
comF25A135	X	X	X	X		
comF50A45	X	X	X		X	
realF25A45	X					
isoF25	X	X				
L5E5	X	X				
E5L5	X	X				

Table 2.2: Mixed filter banks used for classification as well for synthetic as for the real surfaces in database 1

2.3 Image Formation

The first steps of classification are measurement and feature extraction. For feature extraction the physical surface has to be linked to its resulting image. Postprocessing of the image results in features. In this section we describe mechanisms to link the surface to its image intensity.

2.3.1 Lambertian Reflectance Model

When light illuminates a surface, two different types of reflectance occur. The *interface* (or *surface*) reflection occurs at the interface between the surface and the air. The fraction of the light that penetrates into the surface, is partly absorbed and partly scattered around within the surface material. After that it is reflected back to the surface into the air again. This type of reflection is called *body* or *diffuse* reflection. The Lambertian reflectance model is a diffuse reflection model, which describes the relationship between the surface derivates and the image intensity. Diffuse reflection is widely encountered in machine vision and must be modelled. The model shows that the perceived intensity of a surface is only dependent on the relative geometry of the facet and the illuminant direction. The radiant intensity varies with the visible area of the facet while the radiance of the facet is constant to the viewer's position.
The equation

$$i = \rho \frac{\vec{s} \cdot \vec{l}}{\left| \vec{s} \cdot \vec{l} \right|} \tag{2.5}$$

describes this model assuming an orthographic projection of the surface to the viewer. The radiant intensity i is the normalized dot product of the facet normal vector $\vec{s}$ and the illumination vector $\vec{l}$ times the albedo ρ of the surface. This is only one model, which uses diffuse reflectance. Other diffuse models were investigated, because the Lambertian model is inadequate for many applications. A detailed treatment can be found in [7], which proved that the Lambertian model is a good approximation for a wide range of textures especially with low slope angles.

2.3.2 Kube and Pendland's Linear Model

A feature model described by the Lambertian surface reflectance model, as shown in section before, is very difficult to implement in a classification algorithm because of the nonlinearities caused by the normalization factor. Kube and Pendland presented a linear model based on the Lambertian reflectance model in [10]. The paper essentially applies a simplified version of the Lambertian surface reflection model to an expression for the power spectral density of the fractal height map. The following equations describe this model for the assumption of an orthographic projection onto the xy-plane.

If we assume that:

- the surface is Lambertian,
- the surface is only illuminated by distant point sources and
- the surface is not self-shadowing.

We can derive the power spectra density from equation 2.5.

$$i(x,y) = \rho(x,y) \frac{\vec{s}(x,y) \cdot \vec{l}(x,y)}{\left| \vec{s}(x,y) \cdot \vec{l}(x,y) \right|}. \tag{2.6}$$

$$\vec{s}(x,y) = \left[-\frac{\partial h}{\partial x} \quad -\frac{\partial h}{\partial y} \quad 1 \right] = [-p(x,y) \quad -q(x,y) \quad 1] \tag{2.7}$$

where

$i(x, y)$ is the radiant intensity,
$\rho(x, y)$ is the albedo of the surface,
$\vec{s}(x, y)$ is the normal vector field of the surface height function $h(x, y)$,
$\vec{l}(x, y)$ is illumination vector field,
$p(x, y)$ is the slope in x direction and
$q(x, y)$ is the slope in y direction.

The assumption of point sources at an infinite distance causes the illumination vector field to have a uniform direction.
The illumination vector is then described by:

$$\vec{l}(x, y) = [\cos(\tau)\sin(\sigma), \quad \sin(\tau)\sin(\sigma), \quad \cos(\sigma)]$$

where

τ is the illumination tilt angle and
σ is the illumination slant angle.

The definition of tilt and slant angle is illustrated in figure 5.2 on page 31.
We get the normalized image intensity $i(x, y)$ by

$$i(x, y) = \frac{p(x, y)\cos(\tau)\sin(\sigma) + q(x, y)\sin(\tau)\sin(\sigma) + \cos(\sigma)}{\sqrt{p(x, y)^2 + q(x, y)^2 + 1}} \tag{2.8}$$

Using the Taylor series of $i(x, y)$ at the normal $(p, q) = (0, 0)$ we obtain:

$$\begin{aligned} i(x, y) \quad \approx \quad & p(x, y)\cos(\tau)\sin(\sigma) + q(x, y)\sin(\tau)\sin(\sigma) + \cos(\sigma) \\ & -\frac{\cos(\sigma)}{2}(p(x, y)^2 + q(x, y)^2) \end{aligned} \tag{2.9}$$

This approximation can be used if $p, q \ll 1$. Kube and Pendland [10] stated that for real surfaces the maximum slope is rarely more than 15°, so that $(p(x, y)^2 + q(x, y)^2) < 0.1$. If the absolute value of the slant angle σ between the average illuminant and the viewer's position is more than 6° the linear terms in equation 2.10 will dominate. The report deals only with slant angles between 30° and 75° so that we neglect the other case. For a detailed investigation for the case $|\sigma| < 6°$ we refer to [10]. In the following the case $|\sigma| > 6°$ is considered. Neglecting the last term in equation 2.10 and transforming it to the frequency domain, we get the final equation:

$$I(\omega, \theta) = \rho^2\omega^2\cos^2(\theta - \tau)\sin^2(\sigma)\, H(\omega, \theta) \tag{2.10}$$

where

$H(\omega, \theta)$	is the surface power spectrum,
$I(\omega, \theta)$	is the image power spectrum,
ρ	is the albedo of the surface,
ω	is the radial frequency and
θ	is the polar angle.

This equation describes a linear dependency between the surface power spectrum and the image power spectrum. This is essential for further investigations of this project.

Chapter 3

Real Surfaces

A short overview of the real textures used in this report is given in this chapter. Two different databases are used to solve two different types of problems. Database 1 contains images of 29 textures illuminated with an illumination slant angle of 45° and illumination tilt angles between 0° and 180°. We use this database for classification of textures with known slant, but unknown tilt angle. In real cases the illumination slant angle is also unknown so that the classification algorithm is extended to classify textures with unknown illumination slant and tilt. For this classification we cannot use database 1, because it contains no further information about the behaviour of the illuminated textures in slant direction. Database 2 contains information about the behaviour of rough textures that have been illuminated from various directions of tilt and slant. It represents an extension of database 1 where the images captured with a new camera varying the illumination source. We use the terms illumination tilt angle and illumination slant angle to represent the illumination direction, because they are commonly used in texture analysis. Figure 5.2 on page 31 explains them. The illumination tilt angle describes the angle in the horizontal plane whereas the illumination slant angle describes the angle in the vertical plane. Zenith and azimuth are alternative terms for slant and tilt repectively.

3.1 Database 1

Database 1 contains 29 real world textures captured with an illumination slant angle of 45° and illumination tilt angles between 0° and 180°. For textures *and1* to *and7* the illumination tilt angle of two consecutive images is 15° whereas for the other textures the difference is 10°. The database encloses textiles, rocks, stones, wallpapers and several kinds of food. One image of every texture illuminated with 45° of slant and 0° of tilt can be found in appendix B.1 on page 81.

3.2 Database 2

Database 2 contains 25 real world textures choosen out of the *PhoTex* Database, which can be found under the URL:
http://www.cee.hw.ac.uk/texturelab/database/dbase/index.html .
The database contains most of the textures of database 1 recaptured with a new camera varying the illumination source. Every texture is imaged with two different slant angles (45° and 60°) and illumination tilt angles between 0° and 180° in steps of 30°. One image of every texture illuminated with 45° of slant and 0° of tilt can be found in appendix B.2 on page 82.

Chapter 4

Modelling Rough Synthetic Surfaces

4.1 Introduction

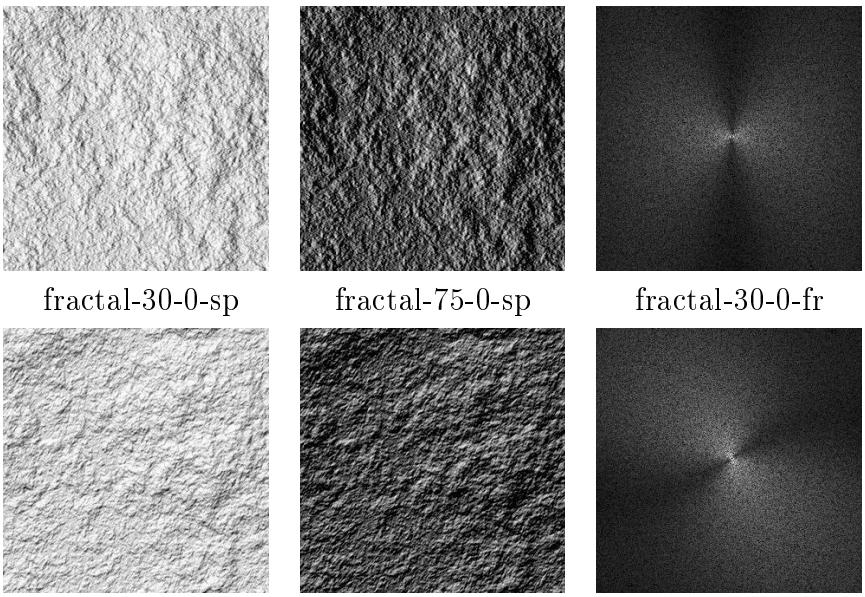
In this section we introduce a model of syntetic surfaces. Although there are disadvantages to use simulated surfaces for classification instead of real textures, synthetic surfaces are useful for investigating the boundary conditions of the classifier. Consider a texture to be a realisation of a two dimensional random process the texture can be described by its means and its phase. Sinn [15] investigated the necessity of the phase spectrum for structured surfaces, because here most of the information is in the phase spectrum. For unstructured surfaces the phase spectrum is assumed to be equally distributed. This report deals only with unstructured syntetic surfaces so that the phase spectrum is neglected. Modelling signals where a certain degree of spatial cohesion or correlation is expected, first order statistics only give a limited insight into their behaviours. Because of this it is customary to characterize textures by their second order statistics. While the autocovariance function (ACF) is used to characterize texture in spatial domain, in frequency domain its Fourier equivalent, the power spectral density (PSD) is used. The surface height map is generated from the surface height function using the Inverse Discrete Fourier Transform. In order to select directional surfaces as well as isotropic surfaces, we select four different surfaces and investigate the behaviour due to changing illumination direction. Therefore we illuminate the generated surface height maps with a Lambertian reflection model (can be found in section 2.3.1 on page 13) varying the illumination direction, calculate the best fit feature equations for every slant angle. The feature curves and its values are plotted.

4.2 Fractal

The first model we investigate is a fractal model, which models natural surfaces. The PSD of Fractal is shown in equation 4.1.

$$S(\omega) = \frac{k_1}{\omega^3} \tag{4.1}$$

The bode plot of Fractal is a straight line (angle of 45°). The information is held in the lower frequencies, which means that the illuminated surface is uneven.



fractal-30-0-sp fractal-75-0-sp fractal-30-0-fr

fractal-30-60-sp fractal-75-60-sp fractal-30-60-fr

Table 4.1: On the left side and in the middle illuminated surfaces of Fractal with a tilt slant angle of 30° (left) and 75° (middle) and a tilt angle of 0° (figure at the top) and 60° (figure below) are shown. On the right side the (to the left side) correspondent PSD are shown.

Table 4.1 shows surfaces and their PSD illuminated with different illumination directions. We use the nomenclature *model-σ-τ-domain* to describe the illumination direction of the image, where σ is the illumination slant angle, τ is the illumination tilt and domain is either the frequency domain (*fr*) or the spatial domain (*sp*).
A description of the illuminant angles can be found in figure 5.2 on page 31. Both figures on the left side are illuminated with an illumination slant angle

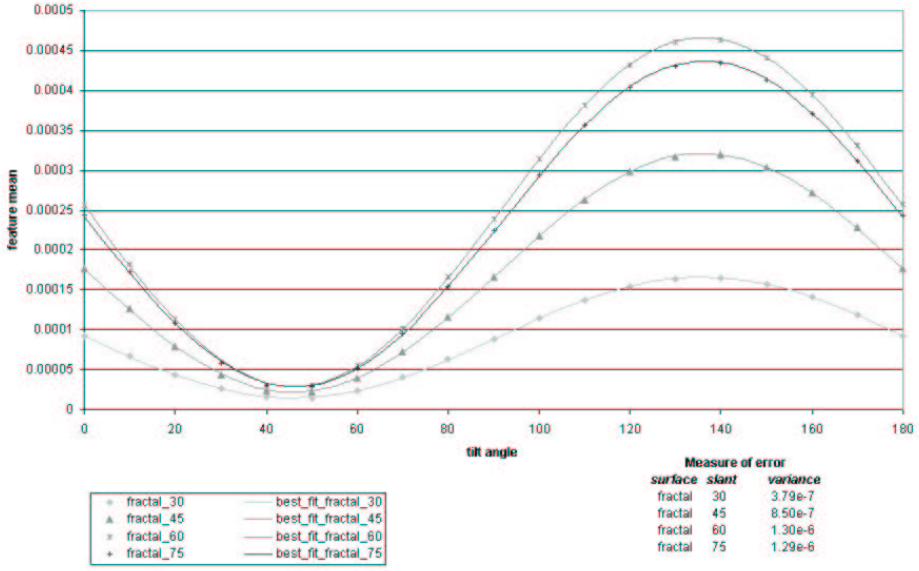


Figure 4.1: Feature values and the corresponding best fit curves of the Gabor F20A45 filtered synthetic surface fractal

of 30°. The texture, rough and uneven, looks like rocks. Changing the illumination slant angle to 75° the image energy falls. The left side of the table shows the PSD for the surfaces illuminated with a slant angle of 30°. The isotropic behaviour due to illumination rotation as well as the distribution of the image energy can be seen. The level of grey value is proportional to the image energy so that for $\omega = 0$ (the center of the figures) the image energy is high. The isotropic behaviour can be seen by comparing both figures. Changing the illumination direction cause a change in the direction of the energy in the PSD.

Figure 4.1 shows the behaviour of a fractal dependent on the illumination direction in feature space. The feature values which are dependent on the illumination tilt angle, describes a sinosodial curve in feature space. The best fit curve is approximated by the feature values. Every best fit curve describes different illumination slant angles. It can be seen that independent of the slant angle the best fit curves fit well so that we assume good results for classification as well as for estimation of illumination tilt.

4.3 Mulvaney

The Mulvaney surface is model developed for machined surfaces. Machined surfaces are assumed to be rough, but even so that the image energy is modeled up to higher frequencies. The bode plot is a horizontal line for low frequencies and a straight line (angle of 45°) for high frequencies.

$$S(\omega) = k_1(k_2\omega^2 + 1)^{-\frac{3}{2}} \tag{4.2}$$

Equation 4.2 shows the PSD of a Mulvaney surface.

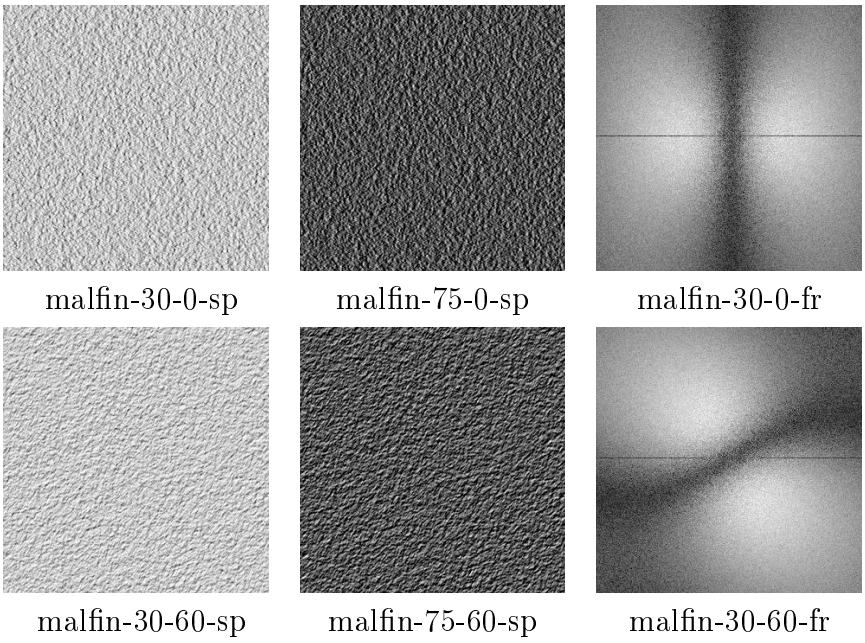


Table 4.2: On the left side and in the middle illuminated images of a Mulvaney surface with a tilt slant angle of 30° (left) and 75° (middle) and a tilt angle of 0° (figure at the top) and 60° (figure below) are shown. On the right side the (to the left side) correspondent PSD are shown.

Table 4.2 shows a set of illuminated surfaces and its PSD in the same order as that for the fractal surface. It can be seen that the surface is even and that the effect of shadowing for high illumination slant angles is low. The figures on the left side show the PSD of a Mulvaney surface for illumination tilt angles of 0° and 60°. The isotropic behaviour as well as the low energy for low frequencies and rising energy for higher frequencies is shown.

Figure 4.2 shows the behaviour of a Mulvaney surface in feature space.

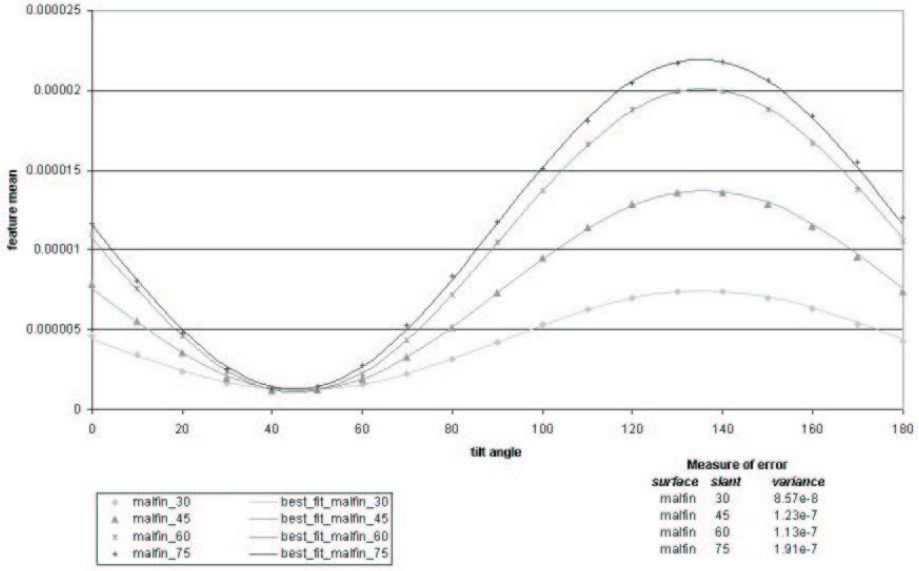


Figure 4.2: Feature values and best fit curves of the Gabor F20A45 filtered synthetic surface Malfin

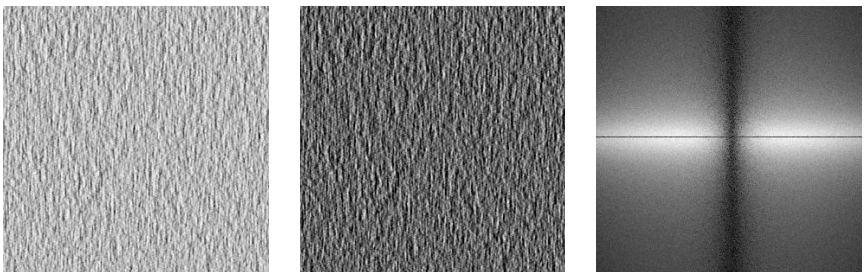
The feature values fit very well to the best fit curves so that we expect a good accuracy for the classification of a Mulvaney surface.

4.4 Ogilvy

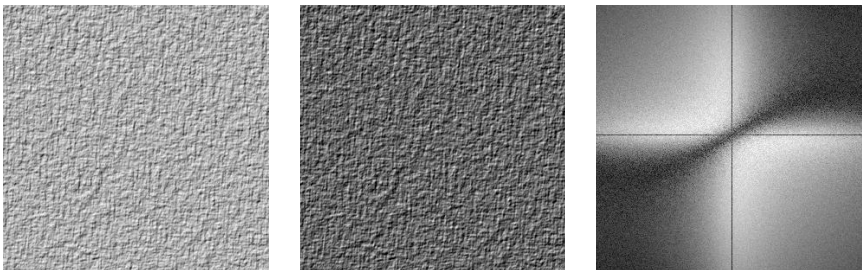
The last two synthetic surfaces we will introduce are directional. While the PSD of the last two surface models where independent of the image direction the Ogilvy PSD is dependent on both surface directions.

$$S(u,v) = k_1 \frac{1}{\frac{1}{\lambda_1^2+u^2}} \frac{1}{\frac{1}{\lambda_2^2+v^2}} \tag{4.3}$$

Equation 4.3 represents the PSD of an Ogilvy surface. In u as well as in v direction the Ogilvy surface acts like a fractal. Choosing the image energy high in one direction and low in the other direction, the surface gets highly directional. Choosing the image energy high in both directions the surface gets directional in both directions.



ogil-30-0-sp ogil-75-0-sp ogil-30-0-fr



ogil-30-60-sp ogil-75-60-sp ogil-30-60-fr

Table 4.3: On the left side and in the middle illuminated images of an Ogilvy surface with a tilt slant angle of 30° (left) and 75° (middle) and a tilt angle of 0° (figure at the top) and 60° (figure below) are shown. On the right side the (to the left side) correspondent PSD are shown.

Table 4.3 presents illuminated surfaces and its PSD of an Ogilvy. The illuminated surfaces show the directionality and the behaviour due illumination direction changing. Only less shadows occur so that Ogilvy surfaces fits well to our assumptions of the classifier.

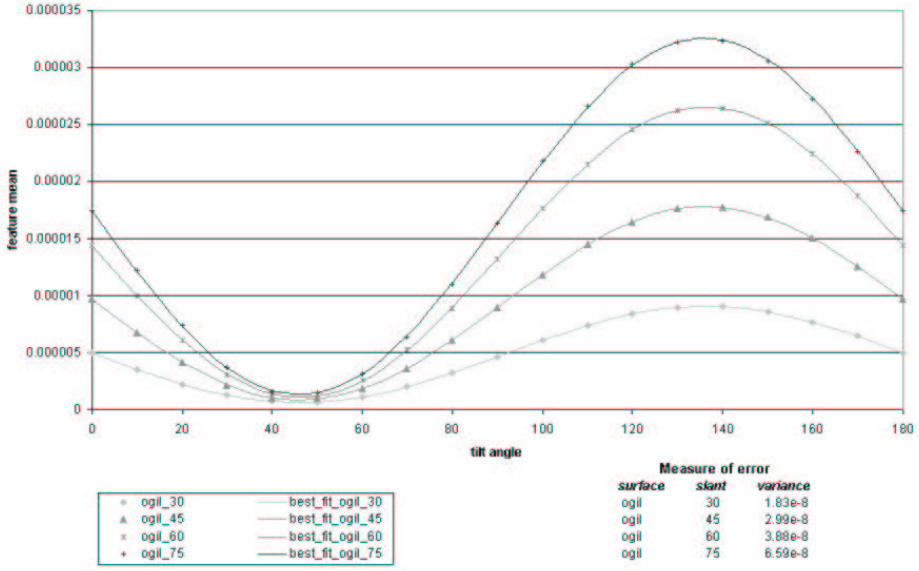


Figure 4.3: Feature values and best-fit-curves of the Gabor F20A45 filtered synthetic surface ogil

The assumption of accuracy is underpinned by figure 4.3, which shows a very good fit of the best fit curve to the corresponding feature values.

4.5 Sand Ripples

Sand ripples is a highly directional fractal surface, which has all its energy in one direction. The PSD that of a fractal surface, where the peak frequency is moved away from DC.

$$S(\omega) = \frac{1}{(f - F_{peak})^{\beta}} \tag{4.4}$$

Where the peak frequency F_{peak} is given by

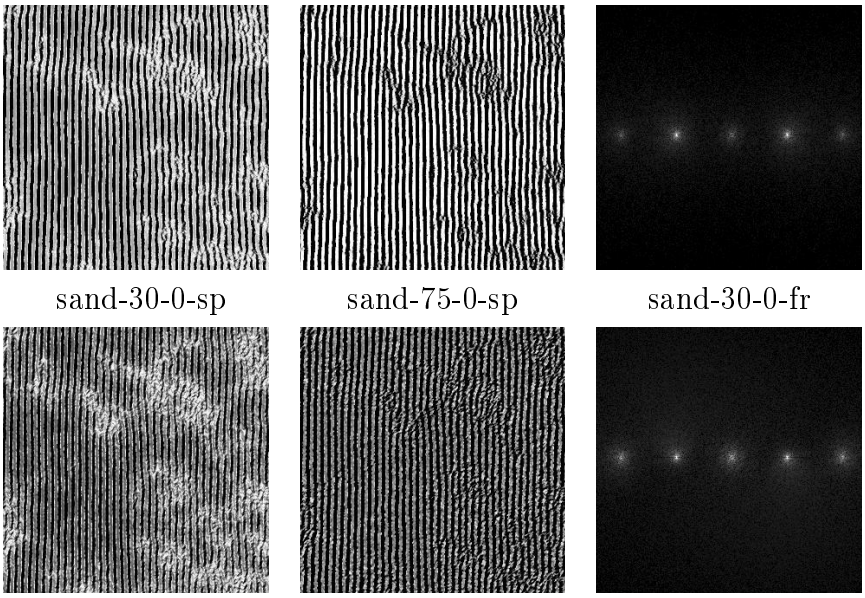
$$F_{peak} = \sqrt{f_x^2 + f_y^2}$$

and the direction Θ of the texture is given by

$$\Theta = \tan^{-1}(\frac{f_y}{f_x})$$

The classification of directional surfaces is demanding related to the feature generation. If only directional bandpass filters are used the feature values are low when the filter direction differs from the surface direction. Another essential problem is shadowing, which occures if the illumination direction is changed. The figures in table 4.4 show that shadowing occures for all the viewed slant angles. The PSD of Sand ripples are only two peaks (the harmonics fall into the viewed frequency range). Changing the illumination direction has no effect of the image direction, the only effect is the change of the level of the image energy.

As assumed the feature values in figure 4.4 vary large from the best fit curve so that an accurate classification of Sand ripples is not expected.



sand-30-0-sp	sand-75-0-sp	sand-30-0-fr
sand-30-60-sp	sand-75-60-sp	sand-30-60-fr

Table 4.4: On the left side and in the middle illuminated images of Sand ripples with a tilt slant angle of 30° (left) and 75° (middle) and a tilt angle of 0° (figure at the top) and 60° (figure below) are shown. On the right side the (to the left side) correspondent PSD are shown.

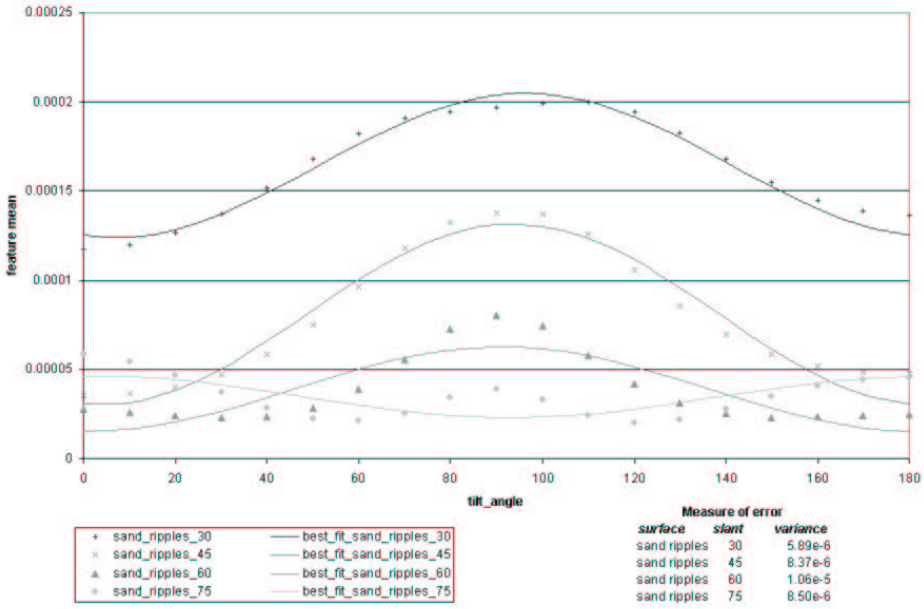


Figure 4.4: Feature values and best fit curves of the Gabor F20A45 filtered synthetic surface Sand ripples.

Chapter 5

Texture Feature Generation

5.1 Introduction

Feature extraction is a fundamental subprocess of the classification problem. In this chapter we start with a short overview over popular and useful feature extraction techniques. After that we introduce the feature generation using linear filters and image energy, which is used in this dissertation.

5.1.1 Feature Extraction Techniques

Feature segmentation and classification is a wide area of research. Roughly, feature segmentation can be categorized into:

- feature selection,
- feature extraction and
- feature reduction.

In the following we treat the feature extraction, which is essential for supervised classification. The main purpose of texture feature extraction is to map differences in spatial structures, either geometric or stochastic into differences in grey value.

As shown in figure 5.1 feature extraction can be categorized into structural, feature based and model based. There is no strict border between feature based and model based methods. Because model parameters are used as texture features, model based methods could be considered as a subclass of feature based methods. The structural methods assume that there exist well defined texture primitives, which compose the textures. Model based

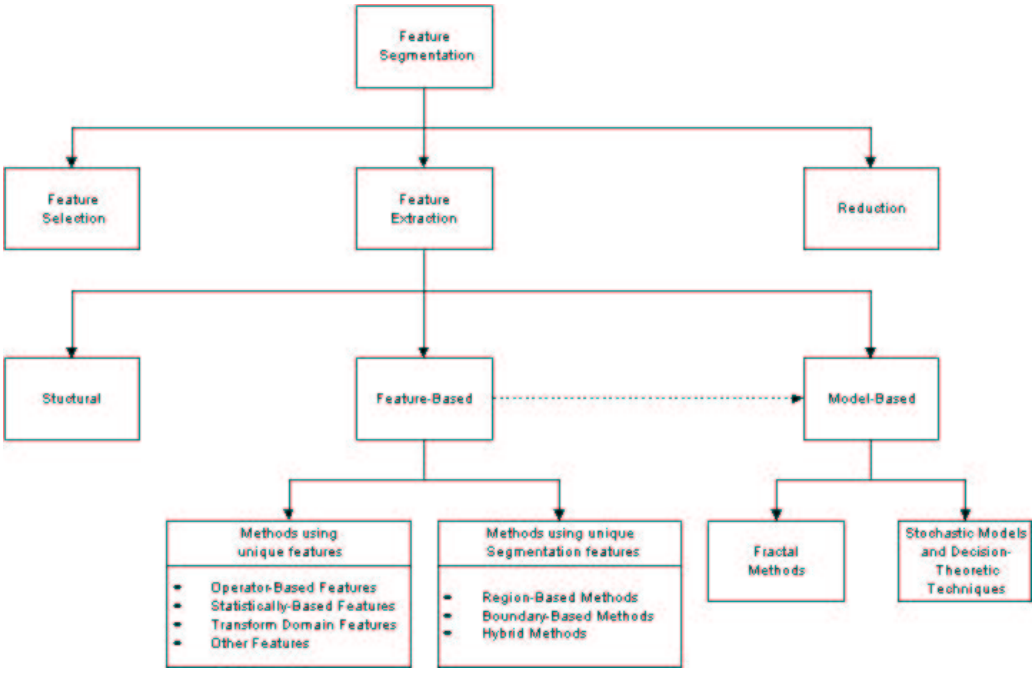


Figure 5.1: Overview on feature extraction techniques according to Reed and Du Buf [13](claims no demand of completeness)

methods can be categorized into fractal and stochatic models. Based on a specific model, (which can be used for generation of synthetic surfaces) the parameters that model the textures best are used as features. Feature based methods are widely used. We can devide them in two groups. The first group uses unique features like operator based features, statistically based features, transform domain features and others. An example for operator based features is the convolution with center weighted filter masks, like Laws masks with the image. The second group is using unique segmentation features. These are region-based, boundary-based and hybrid methods.

So far, we have considered texture extraction methods as being either feature based, model based or structural in nature. Another approach which is used in texture analysis is to group the techniques into statistical methods and those using spatial-frequency or spatial/spatial frequency methods. In the past statistical methods seemed to be superior compared to the spatial-frequency or spatial/spatial-frequency methods, because in the early frequency analysis methods there was no possiblility to investigate the local behaviour in spatial as well as in frequency domain. Spatial/spatial-frequency methods are based on image representations that indicate the frequency content in localized regions in the spatial domain. Such methods are able to

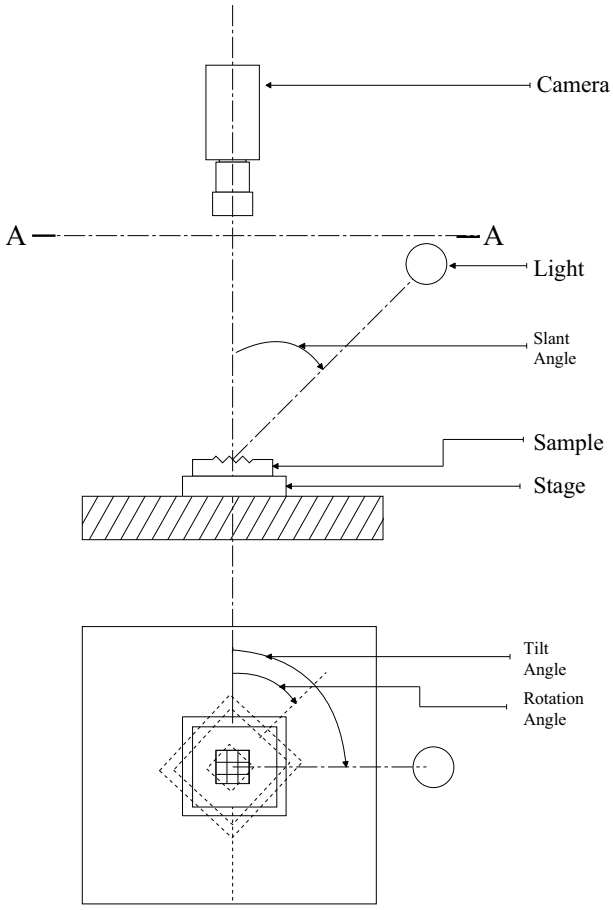


Figure 5.2: Measurement Setup (taken from PhoTex database)

archieve a high resolution in both the spatial and spatial-frequency domains and are consistent with recent theories on human vision.
We use a spatial/spatial frequency method to extract the features. The features are able to characterize local properties of the image so that with a set of features we are able to distinguish between textures as well as for illuminated textures with different illumination directions.

5.1.2 Feature Measurement and Generation

The extraction of features from a texture is divided into two sub-processes. The first sub-process is the measurement.

Figure 5.2 shows the measurement setup. The camera is placed perpendicular to the test surface. For illumination of the texture we use a light

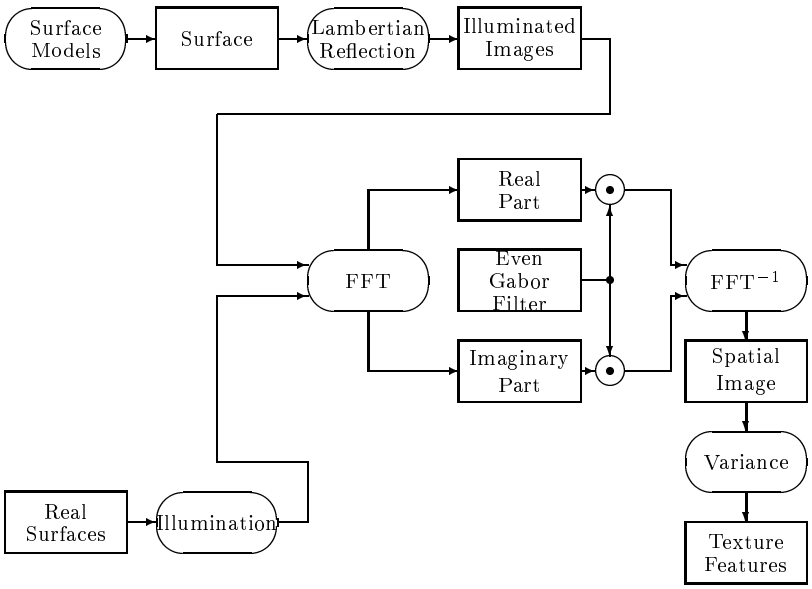


Figure 5.3: Simulation Setup

source, which is assumed to produce parallel light beams equally to a point light source at infinity distance. This is necessary to assume the illumination process as a linear process. We can take shots of the texture under different illumination directions. The camera has no postprocessing stage so that we get the illuminated image of the texture. In the feature generator the illuminated image is filtered with different linear filters followed by a variance estimator. Linear filters are needed to get a filtered image, which is proportional to the illuminated image, so that the feature output is a linear function of the illumination process.

Figure 5.1.2 passes the simulation and test setup to generate the features. As described above the real surfaces passes through an illumination process, are transformed into the frequency domain, filtered with Gabor filters, transformed back into spatial domain and finally a variance estimation process extracts the features. The difference for synthetic surfaces is that here the surface height map passes through a Lambertian reflection algorithm to generate illuminated surfaces.

Chapter 6

Classification

6.1 Introduction

The aim of this chapter is to present a new classification method using linear feature generation. First we give a short introduction of classification techniques, then we present a classifier based on Kube and Pendland's linear model that uses a Maximum Likelihood approach for discrimination. We verify the functionality of the classifier with two different problems. The first is to classify illuminated surfaces with unknown tilt but known slant. This is a simplification of the main problem, to classify illuminated surfaces with unknown tilt and slant angle.

6.2 Classification Techniques

Many different texture classification algorithms have been reported in literature in the last years. Rotation invariance or illumination invariance are important problems in texture classification. Classification consists of measurement, feature extraction and discrimination. In general, texture classification can be divided into supervised and unsupervised classification. Unsupervised classification techniques are mostly automated, while supervised classification normally requires human input in the classification process. The main difference between the techniques can be found in the class generation process. For unsupervised training the training data is used to separate the classes without any previous assumptions about their spatial distribution on the image. Supervised training assumes prior knowledge about the spatial distribution of the classes on the image. The final stage of the image classification process is the discrimination of the image, either pixelwise or areawise.

6.3 Classification using Kube's Linear Model

In this section we describe a general model for a supervised classifier of texture surfaces. We assume the test textures have the following properties:

- Lambertian reflection,
- no self- or cast-shadowing,
- no interreflection,
- small slope angles and
- orthographic projection.

Later we will discuss the problems that occur when textures fail to perform some assumptions. The first two parts of our classifier, measurement and feature extraction, have already been described in previous chapters. Now we will concentrate on the main process of classification: discrimination. As presented in chapter 2.3.2 on page 14 the behaviour of illuminated surfaces can be described by equation 2.10:

$$I(u,v) = \rho^2\omega^2 \cos^2(\theta - \tau) \sin^2(\sigma)\, H(\omega,\theta)$$

We use a feature generator, which is a combination of a linear filter and a variance estimator.

$$f(\tau) = \mathcal{VAR}(o(x,y))$$

Where $o(x,y)$ describes the output of the linear texture filter. As shown in [14] the best fit feature equation can be derived as follows. Assuming the average of $o(x,y)$ as zero, we get:

$$\begin{aligned}
f(\tau) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} O(\omega,\theta)\, dx\, dy \\
&= \int_0^{\infty} \omega^3 sin^2(\sigma) \int_0^{2\pi} cos^2(\theta-\tau) A(\omega,\theta)\, d\theta\, d\omega \\
&= \int_0^{\infty} \omega^3 sin^2(\sigma) \int_0^{2\pi} \frac{1}{2}\,[1 + cos(2\theta)cos(2\tau) \\
&\quad + sin(2\theta)sin(2\tau)] A(\omega,\theta)\, d\theta\, d\omega \\
&= a\sin^2(\sigma) + b\cos(2\tau)\sin^2(\sigma) + c\sin(2\tau)\sin^2(\sigma) \qquad (6.1)
\end{aligned}$$

where

$$
\begin{aligned}
a &= \frac{1}{2}\int_0^{\infty}\omega^3\int_0^{2\pi}A(\omega,\theta)\,d\theta\,d\omega \\
b &= \frac{1}{2}\int_0^{\infty}\omega^3\int_0^{2\pi}cos(2\theta)A(\omega,\theta)\,d\theta\,d\omega \\
c &= \frac{1}{2}\int_0^{\infty}\omega^3\int_0^{2\pi}sin(2\theta)A(\omega,\theta)\,d\theta\,d\omega
\end{aligned}
$$

$A(\omega,\theta)$ is the notional power spectrum of the output of the linear texture filter applied directly to the surface height function.
The above parameters a, b and c are all functions of the surface height function and the linear filter of the texture feature, but none is a function of illuminant tilt angle(τ) or illuminant slant angle(σ). For this reason it is sufficent to know a, b and c in order to describe the behaviour of texture features as a function of the illumination direction. The parameters can be calculated by extracting the texture features for every surface and every linear filter with several different values of illumination tilt and illumination slant. For every texture filtered with n linear filters we get:

$$
\begin{aligned}
a_i &= \frac{1}{n\sin^2(\sigma)}\sum_{j=1}^{N}f_j \\
b_i &= \frac{2}{n\sin^2(\sigma)}\sum_{j=1}^{N}\cos(\tau_j)\,(f_j-a_i) \\
c_i &= \frac{2}{n\sin^2(\sigma)}\sum_{j=1}^{N}\sin(\tau_j)\,(f_j-a_i)
\end{aligned}
$$

where $\vec{\tau}=(\tau_1,\tau_2,\tau_3,...,\tau_N)^T$ covers exactly one period.
Filtering every surface with n linear filters the behaviour of a surface in feature space can be described by its feature vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$, where:

$$
\begin{aligned}
\vec{a} &= (a_1,a_2,a_3,...,a_n)^T \\
\vec{b} &= (b_1,b_2,b_3,...,b_n)^T \\
\vec{c} &= (c_1,c_2,c_3,...,c_n)^T
\end{aligned}
$$

The standard deviation s_i of the feature values to the corresponding best filt curve can be calculated by the following equation:

$$s_i = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}(f_i(\tau_j) - (a_i + b_i\cos(\tau_j) + c_i\sin(\tau_j)))^2}$$

where

$f_i(\tau_j)$ is the feature value i for a texture illuminated with a tilt angle of τ_j

An elementary investigation of this model can be found in [5] and [14].

6.3.1 Maximum Likelihood Approach

In the previous section we derived the theoretical behaviour of texture features due to illumination direction changing. To model the behaviour of textures caused by changes in the direction of the light source and to consider the effects of background noise, shadowing nonlinearities etc., we choose a multivariate Gaussian distribution model. We explain the reasons for choosing a Gaussian distribution in section 6.4.1 on page 38. The real behaviour of texture feature $\hat{x}_i$ can be modeled by the following equation:

$$f_i(\tau,\phi) = \frac{1}{\sqrt{2\pi}\sigma_i} exp[-\frac{(y_i-\mu_i)^2}{2s_i^2}] \tag{6.2}$$

where

$$\mu_i = \sin^2(\sigma)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau))$$

and y_i is the real feature value. For every surface filtered with n different linear filters, we get:

$$\vec{f}(\tau,\sigma) = (f_1(\tau,\sigma), f_2(\tau,\sigma), f_3(\tau,\sigma), ..., f_n(\tau,\sigma))^T \tag{6.3}$$

The idea of classification is to identify the texture, to which all the vector elements are nearest. For the given probability density functions $f_i(\tau,\sigma)$, where the unknown values are τ and σ, a solution can be found by maximizing the product of all the probability density functions. A solution of this problem is the Maximum Likelihood approach, which is a method for determining estimators and their parameters. A detailed introduction of the use of estimation algorithms in signal processing can be found in [8]. In our case

the Likelihood function $L(\tau, \sigma)$ is the probability density for the occurence of a sample configuration $\vec{a}_i$, $\vec{b}_i$, $\vec{c}_i$ and $\vec{s}_i$ where i defines a choosen surface.

$$L(\tau, \sigma) := f_1(\tau, \sigma) \cdot f_2(\tau, \sigma) \cdot f_3(\tau, \sigma) \cdot \ldots \cdot f_n(\tau, \sigma)$$

The Maximum Likelihood estimator $(\tilde{\tau}, \tilde{\sigma})$ are the values for which $L(\hat{\tau}, \hat{\sigma})$ is maximal. This is defined by the condition:

$$\frac{d\, L(a)}{d\, a} = 0 \quad \text{or} \quad \frac{d\, ln\, L(\hat{\tau}, \hat{\sigma})}{d\, a} = 0$$

The Maximum Likelihood estimator is:

- consistent,
- at least asymptotically unbiased,
- sufficent and
- best asymptotically normally distributed (BAN) estimator.

6.4 Classification of Illuminated Surfaces with unknown Tilt

As shown in the introduction, changes of illumination significantly can change the appearance of a 3D surface. In this section we specify a general model to get a discrimination algorithm, which is able to classify textures and to estimate their illumination tilt direction. We assume that the illumination slant angle is unknown, but constant for training and classification.

6.4.1 Tilt direction independent classification model

In section 6.3 on page 34 we derived a general model of our feature extraction algorithm (equation 6.1 on page 34).
We got:

$$\begin{aligned} f(\tau) &= \int_0^{\infty} \omega^3 \sin^2(\sigma) \int_0^{2\pi} \frac{1}{2}[1 + \cos(2\theta)\cos(2\tau) \\ &\quad + \sin(2\theta)\sin(2\tau)]A(\omega,\theta)\, d\theta\, d\omega \\ &= a\sin^2(\sigma) + b\sin^2(\sigma)\cos(2\tau) + c\sin^2(\sigma)\sin(2\tau) \end{aligned}$$

where:

$$\begin{aligned} a &= \frac{1}{2} sin^2(\sigma) \int_0^{\infty} \omega^3 \int_0^{2\pi} A(\omega,\theta)\, d\theta\, d\omega \\ b &= \frac{1}{2} sin^2(\sigma) \int_0^{\infty} \omega^3 \int_0^{2\pi} cos(2\theta) A(\omega,\theta)\, d\theta\, d\omega \\ c &= \frac{1}{2} sin^2(\sigma) \int_0^{\infty} \omega^3 \int_0^{2\pi} sin(2\theta) A(\omega,\theta)\, d\theta\, d\omega \end{aligned}$$

If the slant angle is known, we obtain:

$$f(\tau) = d + e\cos(2\tau) + f\sin(2\tau) \tag{6.4}$$

where

$$\begin{aligned} d &= a\sin^2(\sigma) \\ e &= b\sin^2(\sigma) \\ f &= c\sin^2(\sigma) \end{aligned}$$

The above parameters (d, e and f) are all functions of illuminant slant (σ), the surface height function and the linear filter of the texture feature, but none of them is a function of illumination tilt. Thus these parameters can be used for illumination tilt invariant classification. To use this model for real textures we have to investigate the effect of noise. The term noise is used as a collective term that includes quantisation noise and temporal noise. As shown in [7] the noise can be modelled as a white noise source. We choose a Gaussian distribution function to model the effect of noise in the image. The Gaussian distribution is defined by its mean and its variance. The mean is assumed as the values of the feature function. Assuming an ergodic process the variance of the Gaussian distribution is equal to the variance calculated by the differences between the measured feature values and its approximation.
The Gaussian distribution is defined as:

$$g(y) = \frac{1}{\sigma\sqrt{2\pi}}\, exp[-\frac{(y-\mu)^2}{2\sigma^2}] \tag{6.5}$$

where y is the input value, μ is the mean and σ is the variance. For a featureset of a texture, generated with one linear filter and several images with different illumination tilt angles, we get d_i, e_i, f_i and the variance s_i. The mean can be derived as:

$$\mu_i(\tau) = d_i + e_i cos(2\tau) + f_i sin(2\tau) \tag{6.6}$$

rewriting equation 6.5 gives:

$$g_i(\tau, y_i) = \frac{1}{s_i\sqrt{2\pi}}\, exp[-\frac{(y_i - (d_i + e_i cos(2\tau) + f_i sin(2\tau))}{2s_i^2}]$$

filtering the texture with n different filters we get :

$$G = [g_0(\tau, y_0), g_1(\tau, y_1), ..., g_N(\tau, y_n)]$$

Assuming independence of the errors estimated for every texture we can calculate the product of $g_i(\tau, y_i)$ where i is the feature function using the i^{th}

filter. The resulting function describes the membership of image y belonging to a texture in the multidimensional feature space. A high result leads to a close relationship of the image to belong to the texture. Calculating the maximum of this function, which is only dependent on the illumination direction we can estimate the direction of the unknown image. This procedure is equal to the Maximum Likelihood (ML) estimation. The discrimination algorithm describes a ML estimation of the illumination tilt direction for a specific texture. The ML estimator itself describes a value, which is proportional to the probability that the unknown image belongs to this texture. Choosing the texture where the ML estimator is maximal we get an estimation of the unknown illuminated image.
As described in section 6.3.1 on page 36 before, the Likelihood function $L(\tau)$ is the product of the feature functions.

$$\begin{aligned} L(\tau) &= g_0(\tau, y_0) \cdot g_1(\tau, y_1) \cdot ... \cdot g_N(\tau, y_N) \\ &= \prod_{i=1}^{n} \frac{1}{s\sqrt{2\pi}} exp[-\frac{(y_i - (d_i + e_i cos(2\tau) + f_i sin(2\tau)))^2}{2s^2}] \end{aligned} \quad (6.7)$$

The value $\tau(y_0, y_1, ..., y_n)$ for which $L(\tau)$ is maximal is the Maximum Likelihood estimator.
It is defined by the condition:

$$\frac{dL(a)}{da} = 0$$

To simplify we take the natural logs:

$$\frac{d \ln L(a)}{da} = 0$$

We get:

$$\begin{aligned} \frac{\delta}{\delta\tau} \ln(L(\tau)) &= \frac{\delta}{\delta\tau} \ln \prod_{i=1}^{n} \frac{1}{s\sqrt{2\pi}} exp[-\frac{(y_i - (d_i + e_i cos(2\tau) + f_i sin(2\tau)))^2}{2\sigma^2}] \\ &= \frac{\delta}{\delta\tau} \sum_{i=1}^{n} \ln(\frac{1}{\sqrt{2\pi} s_i}) \\ &- \frac{\delta}{\delta\tau} \sum_{i=1}^{n} \frac{(y_i - (d_i + e_i \cos(2\tau) + f_i \sin(2\tau))^2}{2s_i} \\ &= 0 \end{aligned} \quad (6.8)$$

This equation can be solved either numerically or by subdividing the sum and substituting $\sin(2\tau) = X$.
Equation 6.8 results in a 4^{th} order polynomial.

$$AX^4 + BX^3 + CX^2 + DX + E = 0 \tag{6.9}$$

The solutions of this equation can simply be found by calculating the roots of the polynomial. We get four solutions for X, which corresponds to eight possible solutions for τ. Choosing the solution, which maximizes the Likelihood function, we get a estimation of tilt beside the value of $L(\hat{\tau})$ for each surface in the database. We can discriminate the unknown surface by choosing the surface with the highest Likelihood value. The result is beside the classification of the surface the estimation of the illumination tilt angle. In the next section we validate this model with a simulation of four synthetic surfaces and experiments using two different databases of 29 and 25 different real world textures.

6.4.2 Simulations and Experiments

This section is divided in two parts, simulation using synthetic surfaces and experiments using real world textures. The synthetic surfaces are used to validate the classification algorithm.

Simulations

The first part consists of a simulation of four different synthetic surfaces imaged with illumination tilt angles between 0° and 180° in 10° steps and illuminated slant angles of 30°, 45°, 60° and 75°. For every slant angle we use the tilt angles $0°, 20°, 40°, \ldots, 160°$ to train the classifier and the remaining for classification. The training database consists of the estimated parameters d_i, e_i, f_i and the variance s_i for every trained surface.

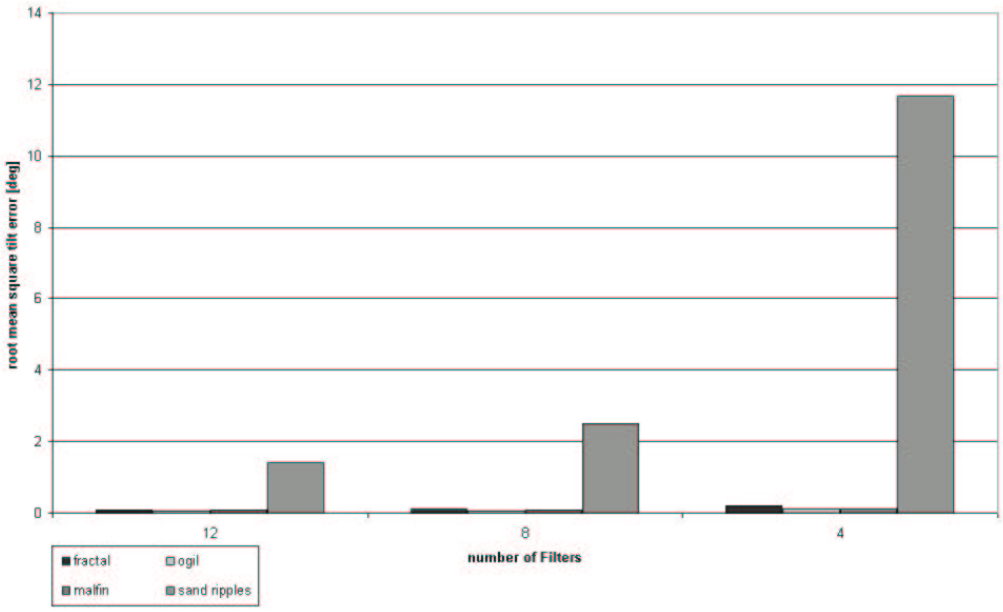


Figure 6.1: Root mean square error of synthetic surfaces as a function of number of filters

As discussed in section 4 on page 19 we expect good results for the Ogilvy and Fractal surface as well as for the Mulvaney surface Malfin. For Sand ripples we expect bad results as well for low as for high slant angles. We use three different complex Gabor filter sets with four, eight and twelve filters. This is a total amount of 12 classifications (four datasets with different slant angles filtered with three different datasets). Surprisingly no misclassifica-

tion occured. Even in the worst case, for the dataset imaged with a slant angle of 75° and a filter bank of only four filters all images are classified correct. Problems occured for the estimation of the illuminant tilt angle. As expected the tilt error for Sand ripples is always much higher than for the other surfaces.
First we will give a short overview about the range of the tilt errors, then we discuss the classification results where the tilt error is large.
Figure 6.1 on the page before shows the root mean square tilt error as a function of the filter bank. We want to point out that the filter banks are choosen intuitively so that they are not choosen for a specific problem. The errors presented are the mean of three classifications for the different slant angles. That means that the root mean square error for Fractal for the Gabor filter bank 12 is the mean of the root mean square errors that occured for Fractal filtered with the Gabor filter bank 12 with the slant angles $30°, 45°, 60°$ and $75°$. It can be seen that except Sand ripples for all surfaces the illumination tilt angle is estimated well. As expected the root mean square error for Sand ripples rises up to more than 10° for a slant angle of 75°. Equation 6.10 on the following page defines the root mean square error.

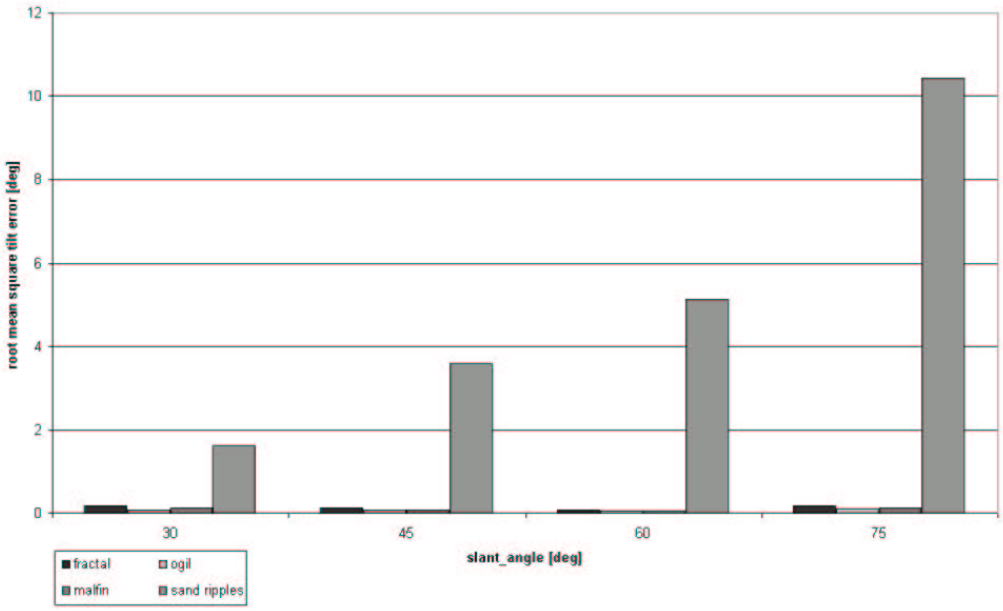


Figure 6.2: Root mean square error of synthetic surfaces as a function of slant

$$e_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tau_i - \hat{\tau})^2} \tag{6.10}$$

τ_i is the real illumination tilt angle and
$\hat{\tau}$ is the predicted illumination tilt angle.

In the following the abbrevation rms error instead of root mean square error is used.

In the same way as for figure 6.1 on page 42 we can calculate the mean of the used filter banks. Figure 6.2 on the preceding page shows the mean rms error for the surfaces as a function of the illumination slant angle. The figure shows that except Sand ripples the rms error is independent of the slant angle. Summarizing both results shows that if the surfaces fits the assumed linear illumination model, the classifier is able to detect beside the surface itself the illumination tilt angle well. If the features of the illuminated surfaces differ obvious from the feature model, we get a less accuracy for tilt estimation and misclassifications can occur. Now we will investigate the accuracy problems of Sand ripples in detail for the Gabor filter bank 4 and a slant angle of 30° and 75°.

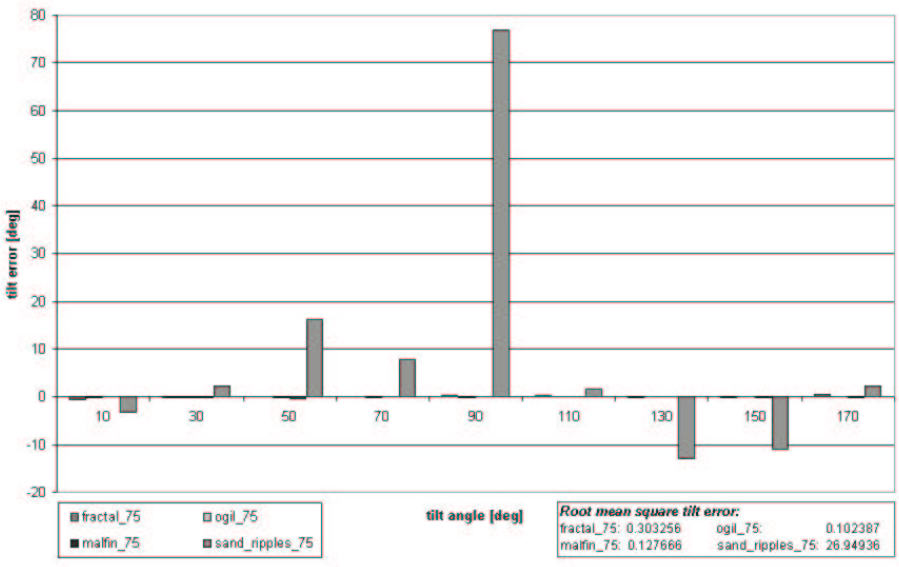


Figure 6.3: Tilt error for four synthetic surfaces for a slant angle of 75° filtered with four different complex Gabor filters.

Figure 6.3 on the page before represents the worst case that occured. The figure presents the estimated tilt error between the input image and the classified image as a function of the illumination tilt angle. For all surfaces except Sand ripples the tilt error is below 1°. For Sand ripples the tilt error rises up to 77° so that a accurate estimation of illuminant tilt with the Gabor filter bank 4 is impossible for high slant angles. To get better results for Sand ripples we investigate the effects that occur when we use the Gabor filter bank 12, which contains 12 complex Gabor filters.

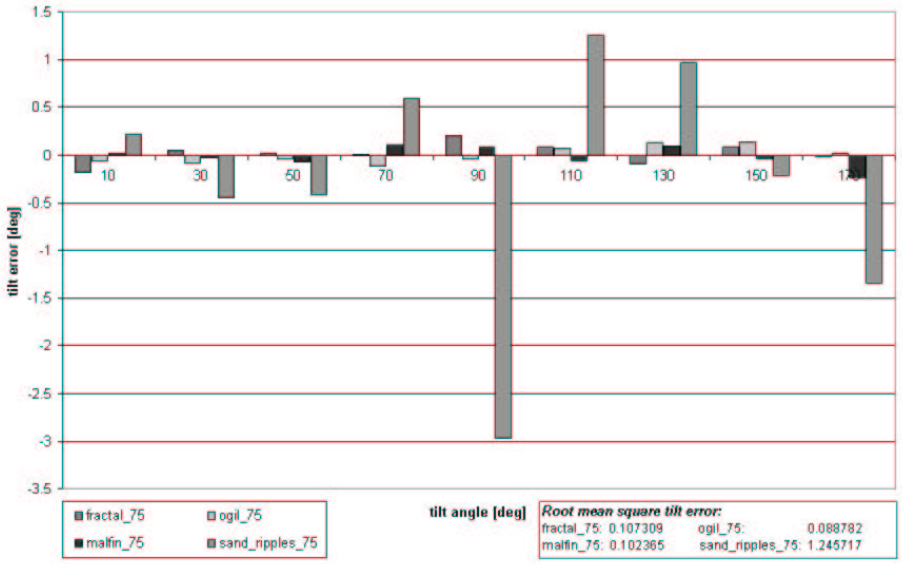


Figure 6.4: Tilt error for four synthetic surfaces illuminated with a slant angle of 75° filtered with 12 different complex Gabor filters.

Figure 6.4 shows the estimated tilt error for Gabor filter bank 12 and a illumination slant angle of 75°. It can be seen that the maximum tilt error decreases to 3° absolute. The accuracy of the classification result is 'bought' with an increasing of the neccessary image information.

Finally we investigate the behaviour of the estimated tilt error by varying the illumination slant angle. Figure 6.5 on the next page shows the estimated tilt error as a function of illumination tilt using Gabor filter bank 4 for a illumination slant angle of 30°. Although we guess that the classifier is able to classify both the surface and the illumination tilt angle well (we recognized a rms error of less than 2°) the maximum of the absolute tilt error for Sand ripples is more than 5°. In appendix A.1 on page 73 the tilt errors for classification with four filters for surfaces illuminated with slant angles of 45° and

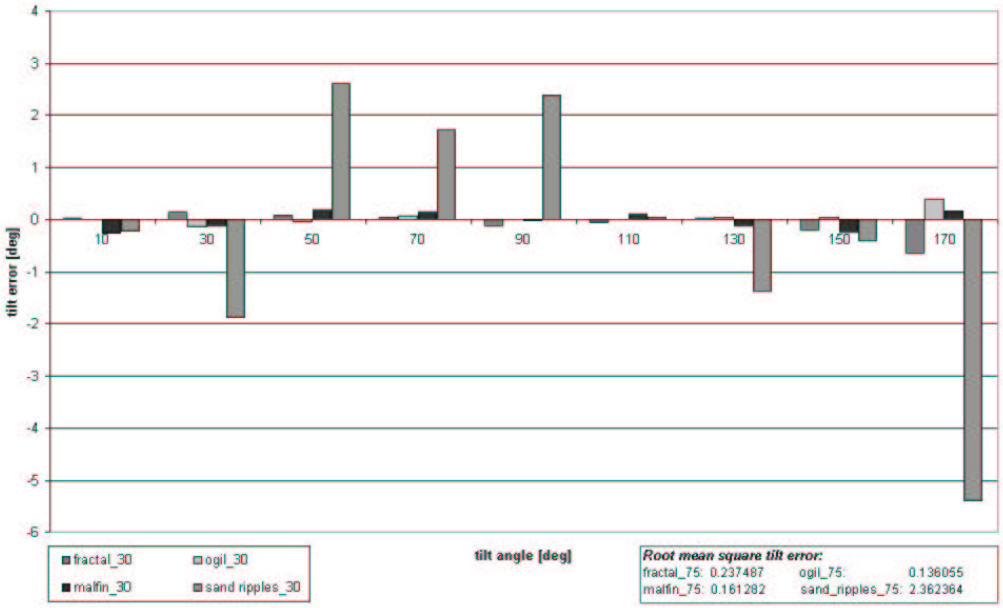


Figure 6.5: Tilt error for four synthetic surfaces illuminated with a slant angle of 30° filtered with 4 different complex Gabor filters.

60° are shown. The behaviour of the surfaces is similar to the classifications, which where investigated just before.

The investigation of synthetic surfaces for a illuminant tilt independent classification showed good results for most of the surfaces. Only for Sand ripples we got large estimated tilt errors as expected.

Experiments

The next step is to classify real world textures. Two different datasets where classified with two different filter bank sets. Both databases can be found in appendix B on page 80. Database 1 contains 29 real world textures imaged with tilt angles between 0° and 180° in 10° steps rspective for *and1* to *and7* in 15° steps. The illumination slant angle is fixed at 45°. We used several different constellations of mixed filter banks and also different training datasets to classify the textures. For classification of database 1 (see appendix B.1 on page 81) the filter banks are choosen out of two Laws' masks (*L5E5* and *E5L5*), one real Gabor filter (*realF25A45*), one isotropic Gabor filter (*isoF25*) and five complex Gabor filters (*comF25A0*, *comF25A45*, *comF25A90*, *comF25A135* and *comF50A45*). The constellations of the different filter banks can be found in 2.2.3 on page 12.

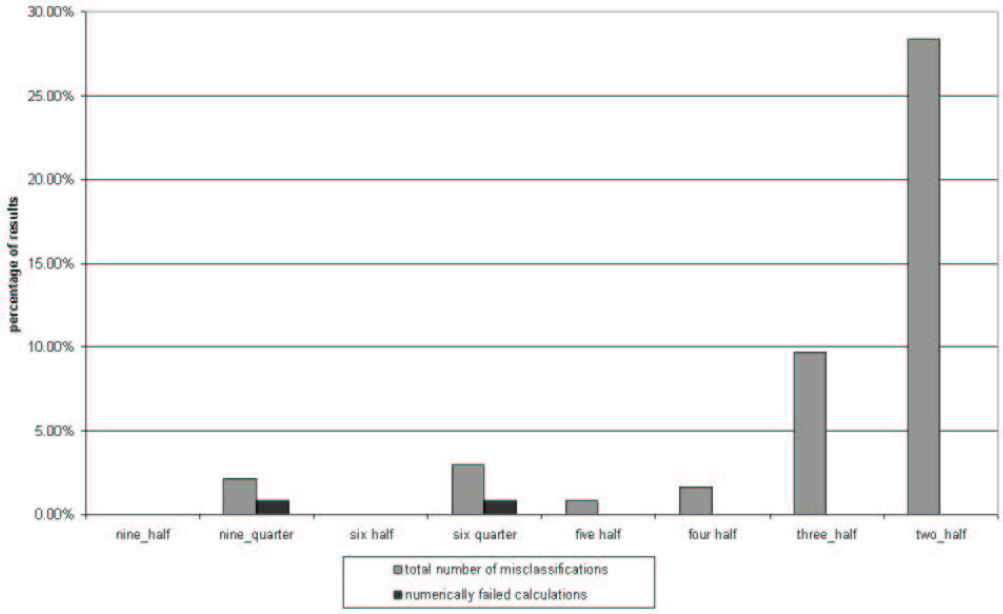


Figure 6.6: Percentage of misclassified images as a function of filters and training values for database 1

Figure 6.6 shows the total percentage of classification errors as well as the percentage of failed calculations using different filter banks for feature extraction and different sizes of the training dataset. A failed calculation occures, when the discrimination algorithm is unable to calculate a ML estimator for the correct surface that means that the roots of the polynomial where not found. If only three or two filters are used, the classification accuracy is bad. For three filters nearly 10% of the images are misclassified. For practical use normally a classification error less than three or four percent is tolerable so that a minimum of four filters are needed to classify database 1. The result is also dependent on the size of the training dataset. The term *"four half"* means that for this classification four filters are used, half of the database is used for training and half for classification. *"Six quater"* means that six filters are used and only one quater of the database is used for training and three quaters are used for classification. If only a quarter of the images are used for training the misclassification error rises rapidely. While there where no misclassifications if only six filters are used and half of the images are used to train the classifier, we got a misclassification rate of more than 3% if only a quater of the images are used for training. We recognized, that the Laws' masks that are only used in the mixed filter bank 9 contain no further information compared to the other filters to get a better classification accu-

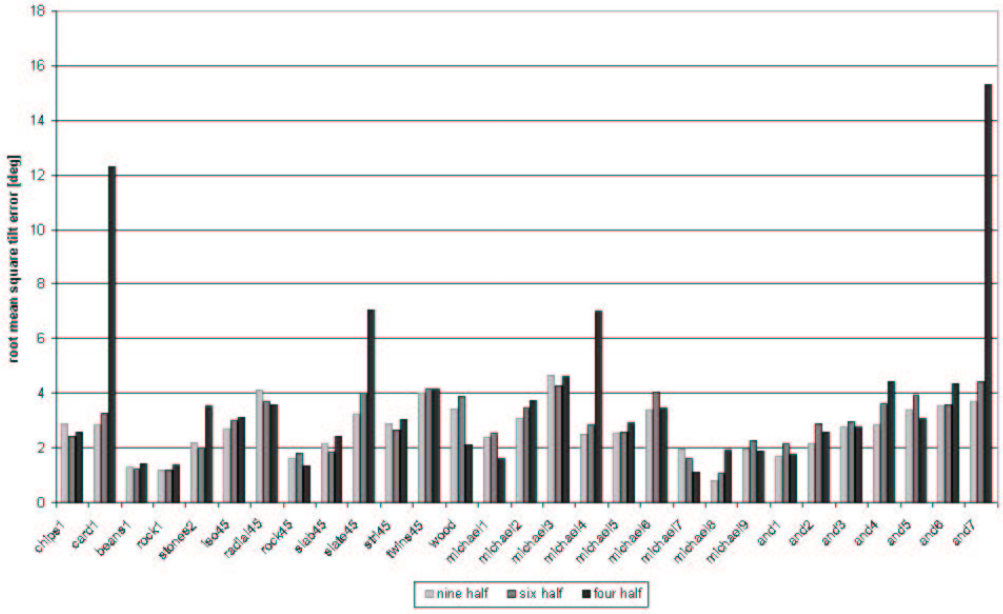


Figure 6.7: The rms error of tilt for classification using four, six and nine filters.

racy. So for further investigations we used filter banks that only contains Gabor filters. We want to point out that the mixed filter banks we used, are not optimal for classification problems, because of overlappings of filters in frequency domain. Plots of the scanned frequency range of the filter banks can be found in appendix 2.2.3 on page 12.

The tilt error is exemplified for nine, six and four filters using half of the database for training. Figure 6.7 shows the rms error of tilt for all surfaces in database 1. Except *and7* and *card1* the rms tilt error is less than 10°. The surface *card1* is highly directional as Sand ripples so that all information of the image is concentrated in one frequency. Using nine filters for feature extraction, we get an average rms error of 2.69°. For six filters the average rms error rises slightly to 2.88° whereas we get an average rms error of 3.82° if only four filters are used for feature extraction.
A great disadvantage of the used ML-approach is, that when only one of the extracted features helds completely wrong information of the image, the value of the ML estimator gets low, which can cause misclassifications and calculation failures. If a feature helds no information about the image the effect of noise controls its behaviour due to changes in illumination direction, which is assumed as gaussian distributed. The model we use is able

to describe this effect, but if only a few feature values are used for training, the algorithm predicts a feature curve, which fits the feature values used for training very well so that a small variance is calculated instead of a much higher variance we get, if more feature values are used. If we classify highly directional surfaces illuminated with a certain direction some features held no information of the image so that noise dominates. This is the case for *card1* and also for *and7* if only four filters are used.

Because of the lack that database 1 contains only images with an illumination slant of 45° so that it cannot be used for classification with unknown tilt and slant, we had to introduce the new texture set database 2. This database does not represent a complete set of new textures, it rather uses old textures, imaged with slant angles of 45° and 60° and tilt angles between 0° and 180°. The tilt distance between two cosecutive images is 30°. We use the tilt angles 0°, 60° and 120° for training and 30°, 90° and 150° for classification. The feature value we got for an illuminant tilt angle of 180° is neither used for training nor for classification, because using it for training would rise the influence of the best fit curve at the illumination tilt angle 0°, which is equal to 180° compared to the other feature values. We did not use this feature values for classification, because we assumed a π-periodic sinosodial function of the feature values in feature space so that the similar feature value is already used for training. For less training values the accuracy of the estimated best fit feature curves to the feature model gets lower so that the estimated tilt error rises. Because we assume an ergodic process, we calculate the variance using the tilt deviations between the feature curve and the extracted features. This causes the problem that for less training values the feature curve gets closer to the feature values and the variance gets lower. The resulting process is no longer ergodic. Although we know this problem in our case we assume the process as ergodic to keep the size of the image database small. So beside the less accurate estimated feature curves the Gaussian distribution function gets narrower. For this reason we assume a larger tilt error for database 2 than for database 1.

Figure 6.8 on the next page shows the misclassification rates dependent on the number of filters used for feature extraction. We got resonable results only for using more than six texture features (for every feature a filter is needed). For eight filters we got a misclassification rate, which is smaller than 4%. Failed calculations where detected for the classifications using six and eight filters.

Figure 6.9 on the following page shows the rms tilt error using four and

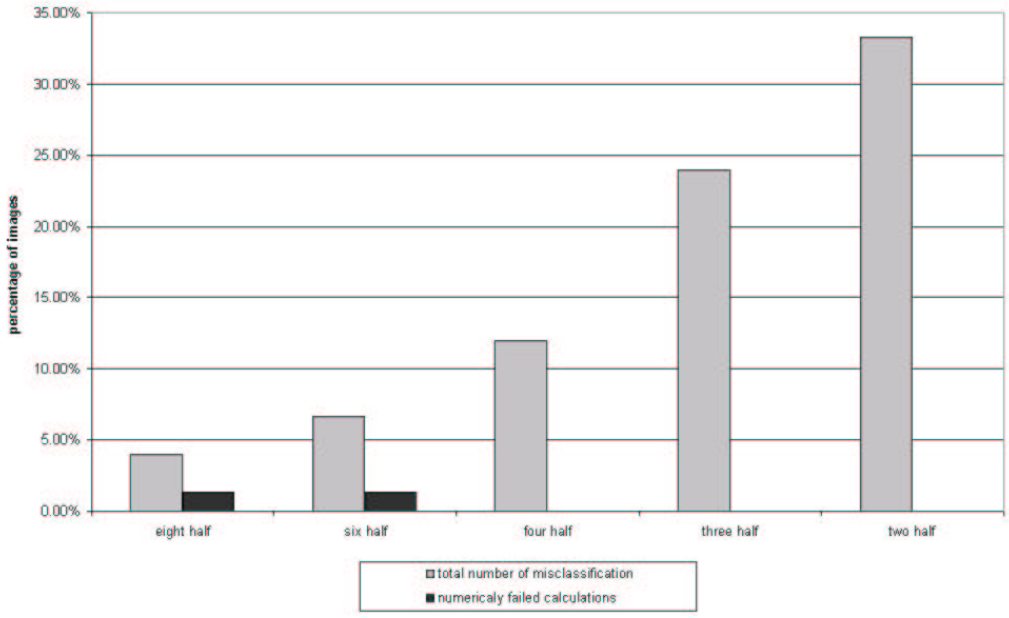


Figure 6.8: Percentage of misclassified images as a function of filters and of number of training values for database 2

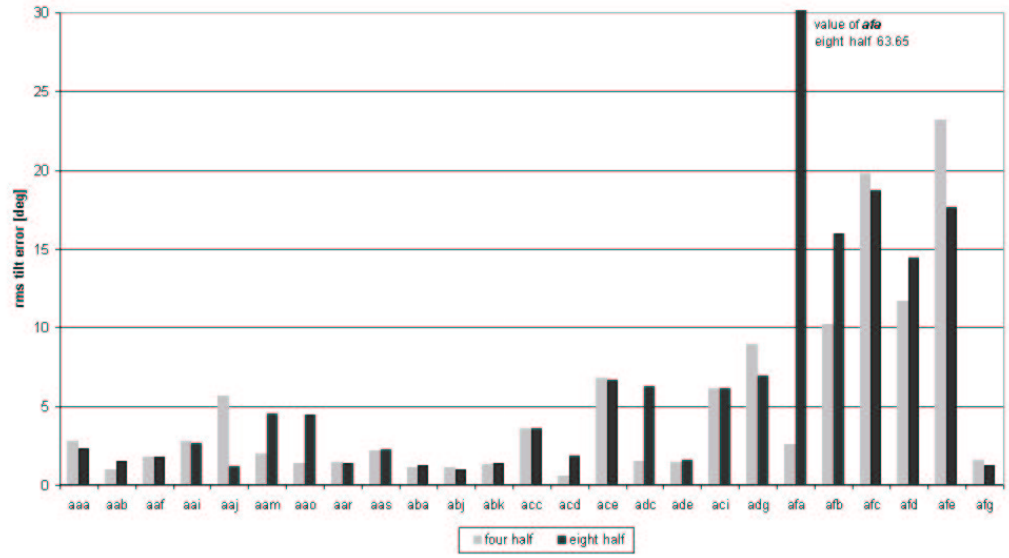


Figure 6.9: The rms tilt error dependent on the slant angle for classification using 8 and 4 filters (database 2).

eight filter for classification. Comparing the results of the two identical textures *and1* and *afa* (captured with different resolutions), for the same classification using four features confirms our prediction. Especially for the images we grabbed from textiles large rms tilt errors occured. While for *and1* to *and6* (database 1) the rms tilt error was below 5° the rms tilt error for the same textiles imaged with a different resolution rised up to more than 63°. Tables which show the misclassifications we got for classification with different filter banks are shown in appendix A.1 on page 73.

6.5 Classification of Illuminated Surfaces with unknown Tilt and Slant

The classifier presented in the last section is able to detect surfaces and estimate the illumination tilt angle, but changing illumination slant between training and classification can also cause complete misclassification. In this section we modify the discrimination algorithm, so that the classifier is able to classify the surface and estimate both the illumination tilt and illumination slant angle. Because of nonlinearities (self- or cast-shadowing) of real surfaces, we reduce the range of illuminant slant between 30° and 60° for real surfaces and 30° to 75° for syntetic surfaces. Tables which contains the misclassifications and the rms tilt error for using Gabor filter bank 8 are shown in appendix A.2 on page 77.

6.5.1 Illumination direction independent classification model

Starting from the best fit feature equation 6.1 on page 34, we get:

$$f(\tau,\sigma) = a\sin^2(\sigma) + b\cos(2\tau)\sin^2(\sigma) + c\sin(2\tau)\sin^2(\sigma) \tag{6.11}$$

where the parameters a, b and c are only functions of the surface height function and the linear filter of the texture feature. Assuming Gaussian Variation to approximize the effects of noise.

$$g_i(\tau,\sigma,y_i) = \frac{1}{s\sqrt{2\pi}}\, exp[-\frac{(y_i - \sin^2(\sigma)(a_i + b_i cos(2\tau) + c_i sin(2\tau)))^2}{2s^2}] \tag{6.12}$$

where $g_i(\tau,\sigma,y_i)$ is the best fit feature estimation function of a filtered image (filter i) for the feature value y_i dependent on the illumination tilt and slant angle.

The Maximum Likelihood Function is the product of the estimation functions and can be written as:

$$L(\tau,\sigma) = \prod_i \frac{1}{s\sqrt{2\pi}}\, exp[-\frac{(y_i - sin^2(\sigma)(a_i + b_i cos(2\tau) + c_i sin(2\tau)))^2}{2s^2}] \tag{6.13}$$

Taking the natural logs we get the Log Likelihood function:

$$\ln f(\tau) = \ln \prod_i (\frac{1}{s_i\sqrt{2\pi}}) + \sum_i \frac{(y_i - d_i - e_i cos(2\tau) - f_i sin(2\tau))^2}{2s_i^2}$$

The resulting equation is a function of illumination tilt (τ) and slant angle (σ). The vector $\vec{v}_{est} = (\tau(y_0, y_1, ..., y_N), \sigma(y_0, y_1, ..., y_N))^T$, the Maximum Likelihood estimator, can be found by calculating the Maximum of $\ln L(\tau, \sigma)$. The necessary conditions for extremas of two dimensional functions can be found in [2].
They are:

$$g_\tau(\tau, \sigma) = 0 \quad (6.14)$$
$$g_\sigma(\tau, \sigma) = 0 \quad (6.15)$$

where $g_\tau(\tau, \sigma) = \frac{\delta}{\delta\tau} g(\tau, \sigma)$, $g_\sigma(\tau, \sigma) = \frac{\delta}{\delta\sigma} g(\tau, \sigma)$

The sufficient conditions for extrema can be found by calculating the determinant:

$$\Delta = \begin{vmatrix} g_{\tau\tau}(\tau, \sigma) & g_{\tau\sigma}(\tau, \sigma) \\ g_{\tau\sigma}(\tau, \sigma) & g_{\sigma\sigma}(\tau, \sigma) \end{vmatrix}$$

The type of the extrema is dependent on the result of Δ:

$$\begin{aligned} \Delta > 0 &\Rightarrow \begin{cases} g_{\tau\tau}(\tau, \sigma) < 0 \text{ maximum} \\ g_{\tau\tau}(\tau, \sigma) > 0 \text{ minimum} \end{cases} \\ \Delta < 0 &\Rightarrow \text{no extremum} \\ \Delta = 0 &\Rightarrow \text{more complex discussion is needed} \end{aligned}$$

Because of the mathematical complexity (associated with a high computal cost) we simply calculate the necessary conditions and check the maximum by inserting the possible values in the Maximum Likelihood function. This is possible, because for the classification of tilt and slant we get a maximum of 24 solutions for the illumination tilt angle (provided that equation 6.14 is solved for σ and inserted into equation 6.15) and 48 possible solutions for the illumination slant. Substituting $\cos^2(2\tau) = 1 - \sin^2(2\tau)$, $X = \sin^2(2\tau)$ and $Y = \sin^2(\sigma)$ for $L_\tau(\tau)$ and $L_\sigma(\sigma)$, we obtain:

$$\begin{aligned}
\frac{\delta}{\delta\tau}L(\tau) &= \sum_i \frac{1}{2\sigma_i} 2(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau))) \\
&\quad \cdot(-\sin^2(\phi)(-2b_i\sin(2\tau) + 2c_i\cos(2\tau))) \qquad (6.16) \\
\frac{\delta}{\delta\sigma}L(\sigma) &= \sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau)))^2 \qquad (6.17)
\end{aligned}$$

Solve equation 6.17 for Y we obtain:

$$\begin{aligned}
Y &= \frac{MAY \pm \sqrt{1-X^2}MBY + XMCY}{(MAA + MBB) + 2XMAC + X^2(MCC - MBB)} \\
&\quad \frac{}{\pm\sqrt{1-X^2}(2XMBC + 2MAB)}
\end{aligned} \qquad (6.18)$$

Inserting this equation into equation (6.16) results to a 12^{th} order polynomial in X. The solutions for X can be inserted into equation(6.17) to get the solutions for Y. The entire derivation of the polynomial can be found in appendix C.2 on page 103.

6.5.2 Simulations and Experiments

This section starts with simulations of four synthetic surfaces imaged with four different slant angles and tilt angles between 0° and 180° in 10° steps. The slant angles are 30°, 45°, 60° and 75°. We use these images to investigate the limits of our linear reflection model.
The second part of this section is an experiment using database 2, which contains 25 textures illuminated with two different slant angles (45° and 60°) and tilt angles between 0° and 180° in 30° steps. As pointed out in section 6.4.2 on page 46, the tilt stepsize causes problems in accuracy. Because of the large amount of results we select only the bad results and inspect them in detail. A table of misclassifications dependent on different Gabor filter banks are shown in appendix A.2 on page 77.

Simulations
As well as in section 6.4.2 on page 42 first we use simulations to validate the classification algorithm. In section 4 on page 19 we investigated the behaviour of four synthetic surfaces due to changing illumination tilt direction. We found out, that the surfaces Ogilvy, Fractal and Malfin fitted the linear model well whereas for Sand ripples large differences between the best fit curves and the feature values occured. The best fit parameters we estimated, where independent of illumination tilt, but dependent on illumination slant. In order to get the best fit parameters for illumination independent classification we need to estimate them from feature sets that are dependent on tilt and slant. The result is a curve that describes the behaviour of the surface due to changes in illumination direction in the feature space.

To get a feeling of the accuracy of the feature values to its model, we can simply plot the estimated feature curve and the "slant compensated" feature values (the measured feature values are divided by its known slant factor $\sin^2(\sigma)$). Figure 6.10 on the next page exemplifies this for Sand ripples in the one dimensional feature space. If the compensated feature values fits the curve well, we expect good results for the estimation of the tilt and slant. Note, that this is only valid for the inspection of the cuvature and the mean value of the best fit curves, an investigation of the distribution with this model is not possible. Large deviations between the best fit curve and the feature values (as seen in figure 6.10 on the following page show that the surface does not fit the model well.

We choose a more illustrative way to show the accuracy of the linear model is simply to take the estimated best fit curves we got in section 4 on

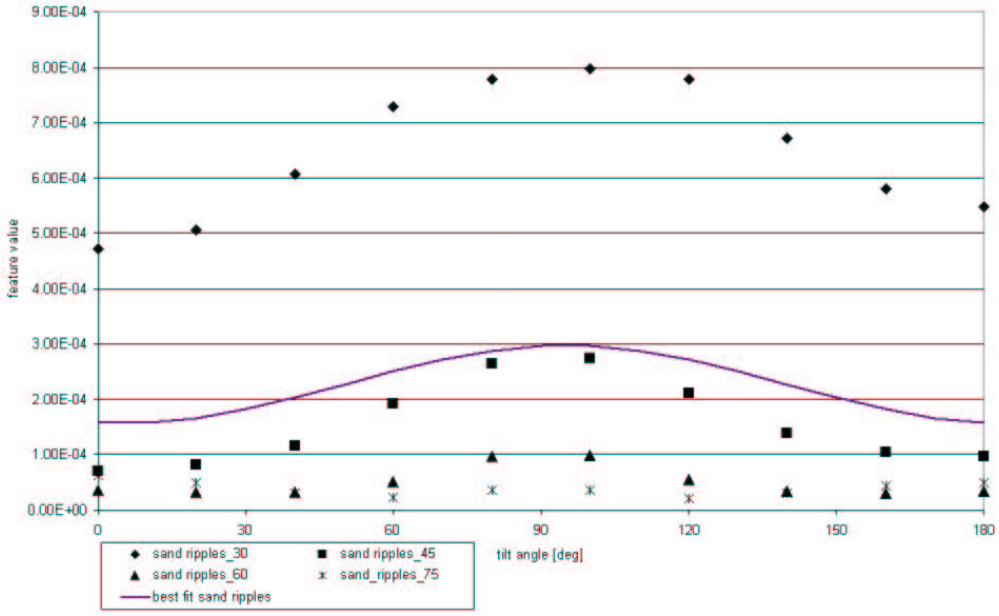


Figure 6.10: Slant compensated feature values and its best fit curve for Sand ripples filtered with the complex Gabor filter comF20A45.

page 19 and divide them by its slant factor $\sin^2(\sigma)$ to provide "slant compensated" best fit curves (eg. figure 6.11 on the following page). The resulting curves approximate the illumination tilt and slant independent best fit curve. Differences between two curves result from nonlinearities of the surface in feature space. While differences of the mean values show a bad fit of the slant angle, differences of the curvature show a bad approximation of tilt. We expect good results for the classification, if all the slant compensated best fit curves describe the same illumination independent best fit curve. Differences cause errors in illumination tilt and slant and can cause misclassifications.

Figure 6.11 on the next page shows the feature values of an Ogilvy surface divided by its slant factor $\sin^2(\sigma)$. All feature curves describe nearly the same best fit curve so that Ogilvy fits well to our model. We expect good classification results and also good results for the estimation of tilt and slant.

For a Mulvaney surface the model is no longer optimal. In figure 6.12 on the following page we detect differences in the mean values and slightly differences in the shapes of the curves. The best fit curve which we use for the tilt and slant invariant classification can roughly be estimated as the curve we get if we average the values of the four curves for every tilt angle. For this reason we expect a higher estimated tilt and slant error than for Ogilvy and possibly misclassifications.

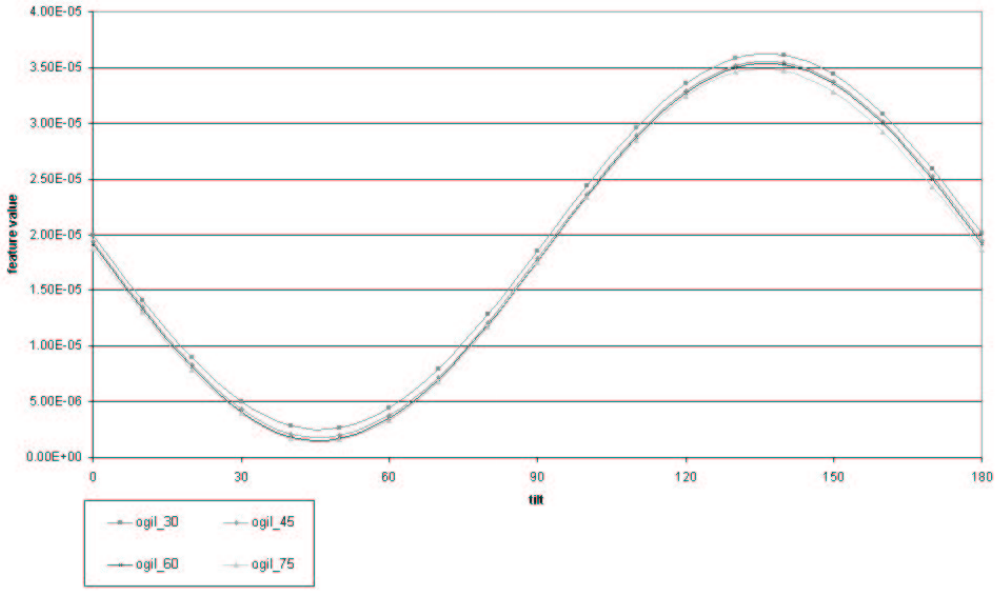


Figure 6.11: Illumination independent feature values of an Ogilvy surface filtered with the complex Gabor filter comF20A45.

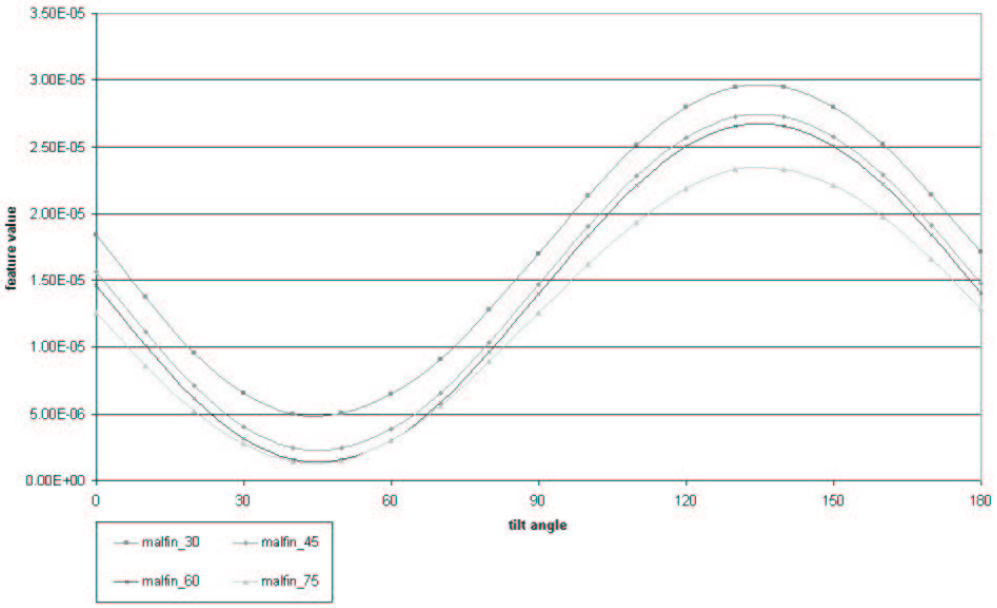


Figure 6.12: Illumination independent feature values of a Mulvaney surface filtered with the complex Gabor filter comF20A45.

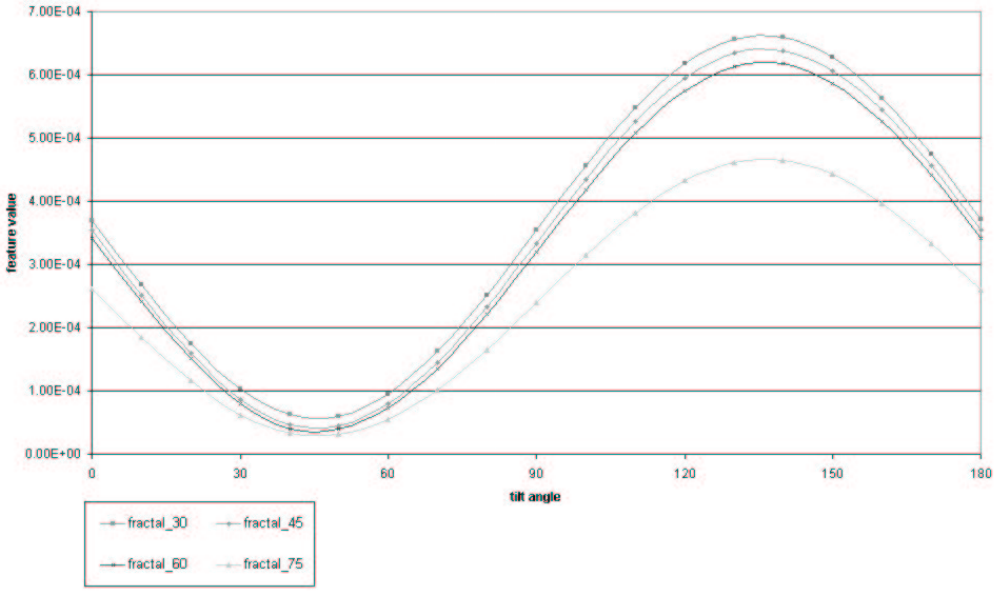


Figure 6.13: Illumination independent feature values of Fractal filtered with the complex Gabor filter comF20A45.

For Fractal the difference of the curves rises compared to the the Mulvaney or Ogilvy surfaces we analysed before. Figure 6.13 shows the slant dependent best fit curves of Fractal divided by its slant factor. The best fit curve we get for a slant angle of 75° differs from the other curves so that the estimation of images illuminated with a high slant angle will cause estimation errors and misclassifications. If we limit the illumination slant angle we would expect good results.

Figure 6.14 on the following page shows the behaviour of Sand ripples filtered with the Gabor filter *comF20A45* in feature space. Beside a very high difference of the mean values of the curves also the shape differs for an illumination slant angle of 75°. The feature curve shows the predicted behaviour of Sand ripples in feature space. For an image illuminated with a tilt angle of 90° and a slant angle of 30° the feature value is more than 1.5 times the predicted value. For classification we expect a high misclassification rate and tilt and slant errors up to 90°.

Summarizing the predictions we expect very good results for Ogilvy, good results for Mulvaney surfaces, relatively bad results for Fractal and extremly bad results for Sand ripples.

The next step is to validate our predictions by classifications using dif-

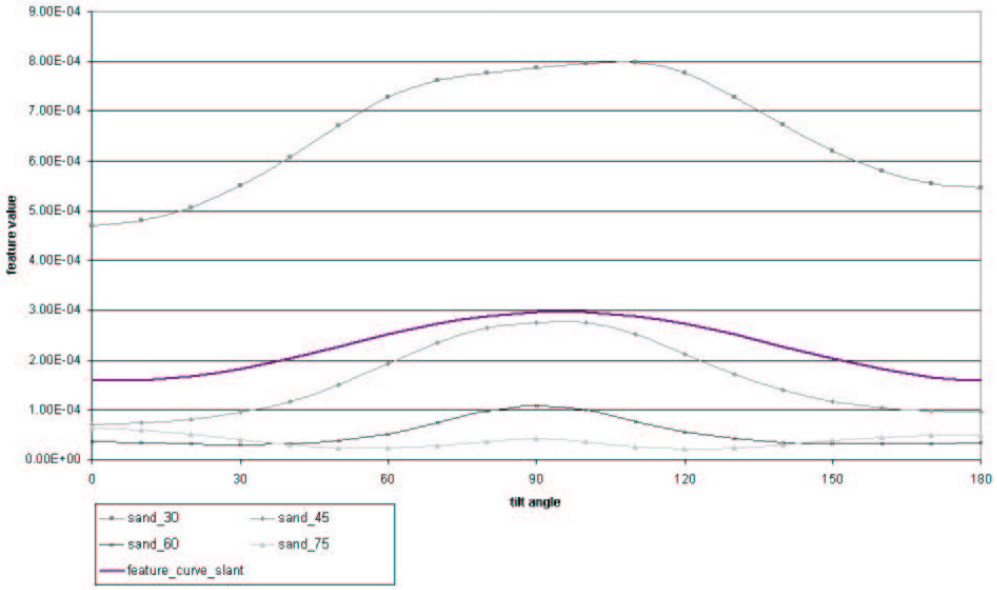


Figure 6.14: Illumination independent feature values of Sand ripples filtered with the complex Gabor filter comF20A45.

ferent feature sets to describe the surface in the feature space. All Gabor filter banks where used for illuminated images with slant angles of 30°, 45°, 60° and 75°. The classifier is trained on illuminated images with the same range of illumination slant angles and with illumination tilt angles between 0° and 180° in 10° steps. For classification the illumination tilt angles $10°, 30°, 50°, \ldots, 170°$ in combination with the illumination slant angles of $30°, 45°, 60°$ and 75° are used.

Figure 6.15 on the next page shows the percentage of misclassifications dependent on the number of filters used to extract the features out of the illuminated images. To classify all the surfaces (except Sand ripples) only the Gabor filter bank 4 is needed. For the maximum percentage of misclassification we get less than 2%. A classification of Sand ripples is not possible even when 12 filters are used. Misclassification errors for four selected classifications can be found in table A.3 on page 77.

We exemplify the estimated tilt and slant error for the worst case where a classification is reasonable for the Gabor filter bank 4. Further investigations showed that increasing the number of feature values improves the estimated errors only slightly.

Figure 6.16 on page 61 shows the estimated tilt errors (figure at the top) and the estimated slant errors (figure at the bottom) that occured when filter

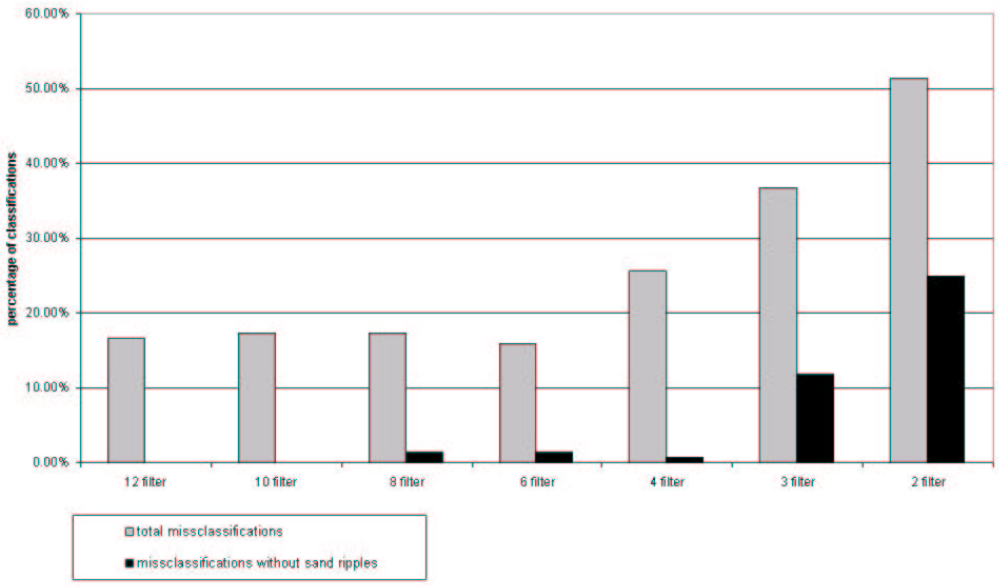


Figure 6.15: Percentage of misclassification for illuminant direction invariant classification of synthetic surfaces. The percentage of misclassification of the entire classification and for the classification of all surfaces except Sand ripples is shown.

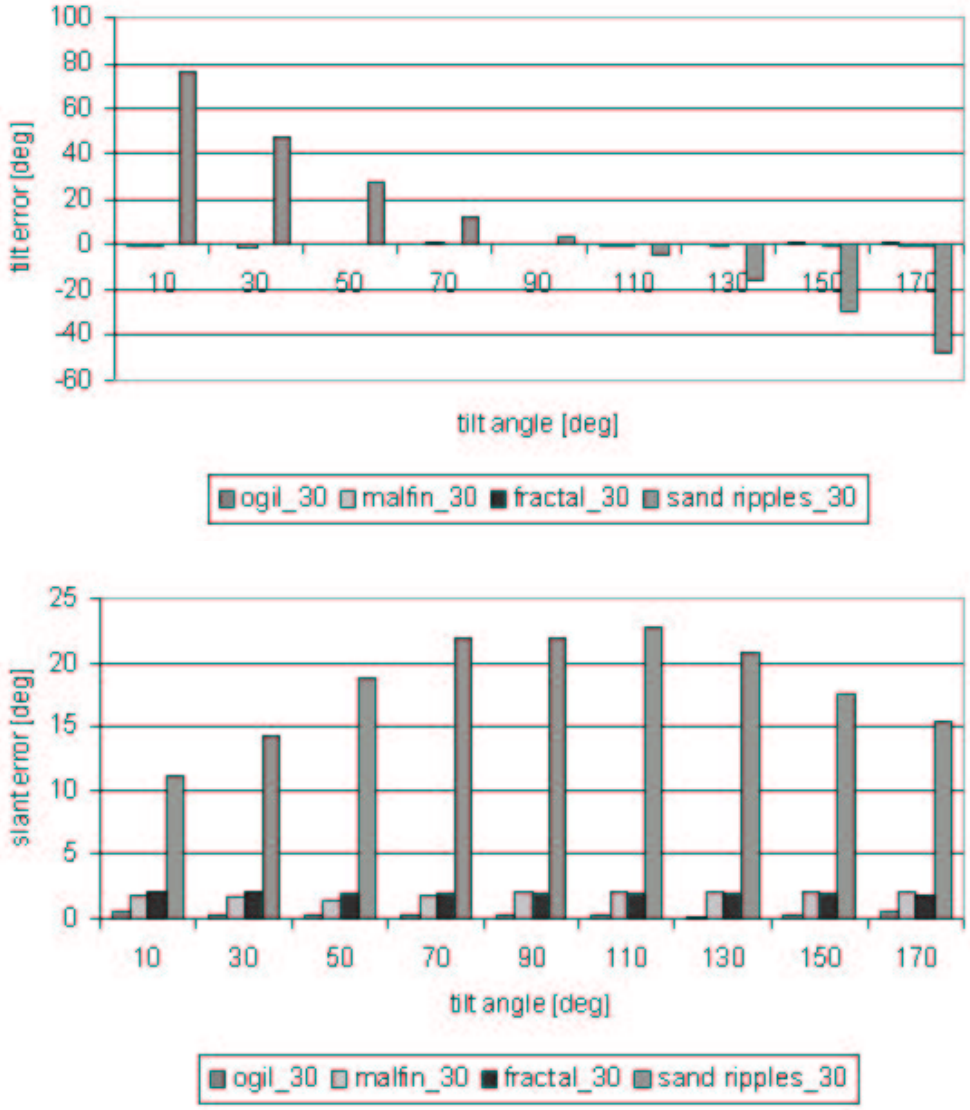


Figure 6.16: Tilt slant classification of synthetic surfaces illuminated with a slant angle of 30°

bank Gabor 4 was used and images with an illumination slant angle of 30° where used. For all surfaces except Sand ripples the tilt error is less than 1.5°. Also the illumination slant angle for these surfaces is detected well. The maximum of the estimated slant errors is less than 2.5°. For Sand ripples as expected high tilt and slant errors with a misclassification rate of 50% occured.

The result using images illuminated with a slant angle of 75° is shown in figure 6.17 on the following page. The estimated tilt error remains almost constant less than 3°, but the slant error rises compared to figure 6.16 on the page before. As shown in the mathematical model in appendix C.2 on page 103 the estimated slant angle is a function of the estimated tilt angle. For this reason a rise in the tilt error causes also a higher slant error. Only for the Ogilvy surface the maximum slant error remains low below 3°.

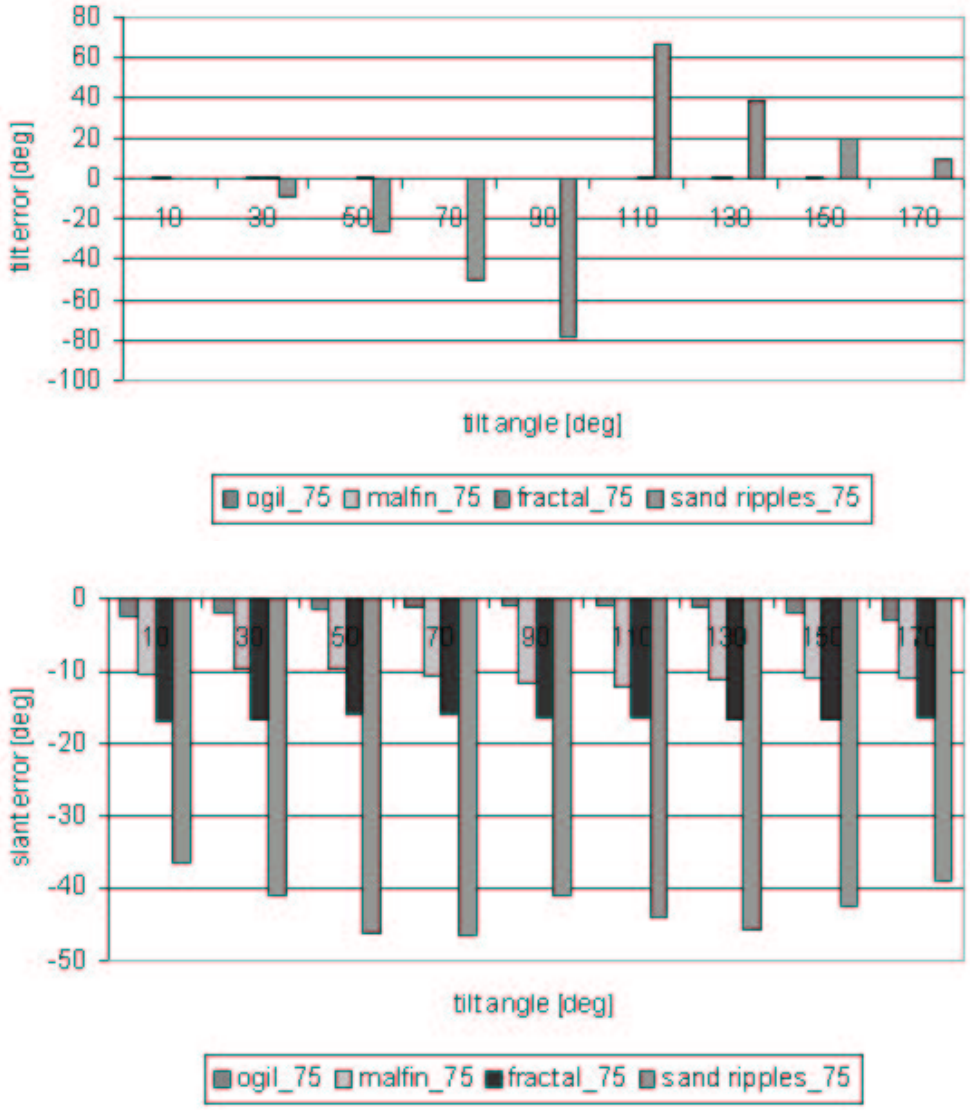


Figure 6.17: Tilt slant classification of synthetic surfaces illuminated with a slant angle of 75°

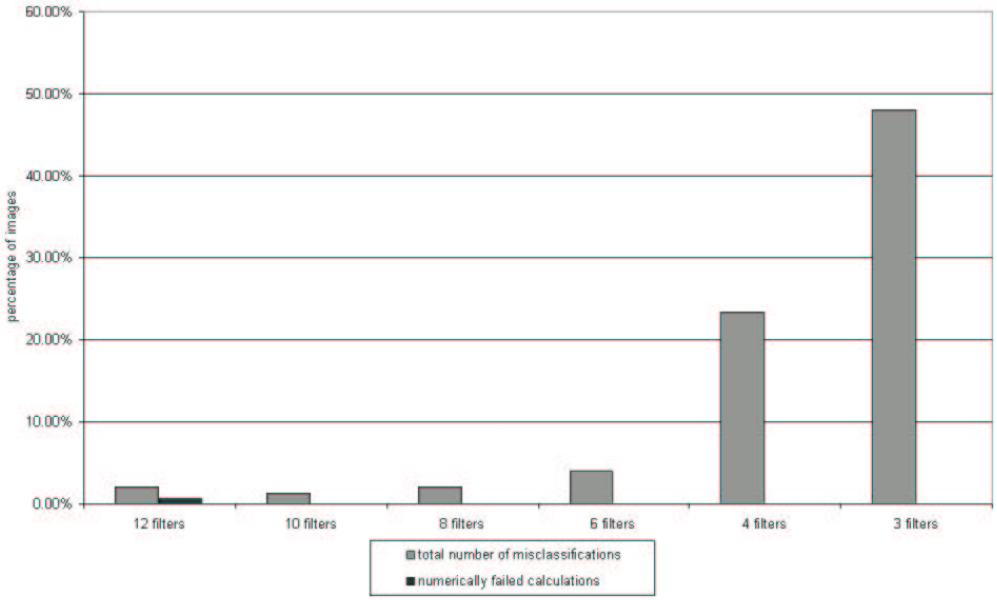


Figure 6.18: Percentage of misclassified images for illumination direction independent classification dependent on the used Gabor filter bank

Experiments

The final step in this report is the classification of real illuminated surfaces with unknown illumination direction. 25 textures are imaged with two different illumination slant angles and seven different illumination tilt angles. They are classified with seven different Gabor filter banks. Detailed information about filter banks can be found in section 2.2 on page 6.

The accuracy of the illumination direction independent classifier is shown in figure 6.18. It can be seen that for classification using Gabor filter bank 4 or 3 the misclassification rate is large. In order to get results for practical use we need to use Gabor filter bank 6 or above. For a classification with Gabor filter bank 10 we get a misclassification rate below 2%. Comparing the results with the tilt classification (shown in figure 6.8 on page 50), we recognize more detection errors for the classification of tilt and slant. This is obvious, because for classification of tilt and slant we have three unknown values, whereas for tilt classification the number of unknown values is two. The next step is to investigate tilt and slant errors that occured. As already described before, the illuminant slant error is a function of the estimated illuminant tilt angle and therefore a function of the tilt error. If the difference between the estimated tilt and the real tilt of an image is large we also get a large deviation between the estimated slant and the real slant angle.

In the following we investigate the classification filtered illuminated images in the eight dimensional feature space in detail. Eight dimensional feature space means that the textures are filtered with eight different Gabor filters so that the illuminated images are described by eight feature values. Figure 6.20 on page 67 shows the rms tilt error (figure at the top) and the rms slant error (figure at the bottom) that occured. The dependency between tilt and slant error is obvious. To our surprise for most of the treated textures the rms tilt error is less than 5°. For these textures the rms slant error is also low. We recognize that for nearly all textures the rms slant error rises when images illuminated with a higher slant angle are classified. For the textiles we got bad estimation results for both, for tilt and slant so that we investigate them in detail (except *afe_60* and *afg_60*), because for them we got relatively good results.

.

The results we got for the estimation of tilt and slant are shown in figure 6.20 on page 67. Nearly all viewed textures have good tilt estimation results for an illumination tilt angle of 30°. Only for *afc_60* the tilt error is more than 15°. It was also recognized that the tilt error rises if the textures are illuminated with large tilt angles. Especially for *aci* we get tilt errors below 3° for the images illuminated with a tilt angle of 30° and 90°, but for 150° the tilt error rises up to more than 20°!. Figure 6.20 on page 67 at the bottom shows the slant errors we got. It can be seen that nearly all errors are negativ, which means that the estimated slant angle is smaller than the real illumination slant angle.

In order to find the cause of the tilt and slant angles we studied the feature generation of these textures. The following figures present the predicted best fit curves, which are independent of the illumination direction and the feature values for the viewed textures illuminated with 60° of slant. The feature values are slant compensated, which means that the feature values are divided by the slant factor $\sin^2(\sigma)$. The features for a tilt angle of 30°, 90° and 150° where used for classification, whereas the remaining together with the corresponding equivalents for the slant angle of 45° where used for training. We selected the worst feature curves to discuss the errors (every image is described by eight features), because the result of the Maximum Likelihood function is proportional to the squared error so that the maximum error is responsible for the accuracy of the estimation. Figure 6.21 on page 68 shows the best fit curve and the slant compensated feature values of *aci_60* filtered with the Gabor filter *comF30A135*. The term *aci_60* describes the images we get when the texture *aci* is illuminated with a slant angle of 60°. The curves above and below the best fit curve are the margins of the area

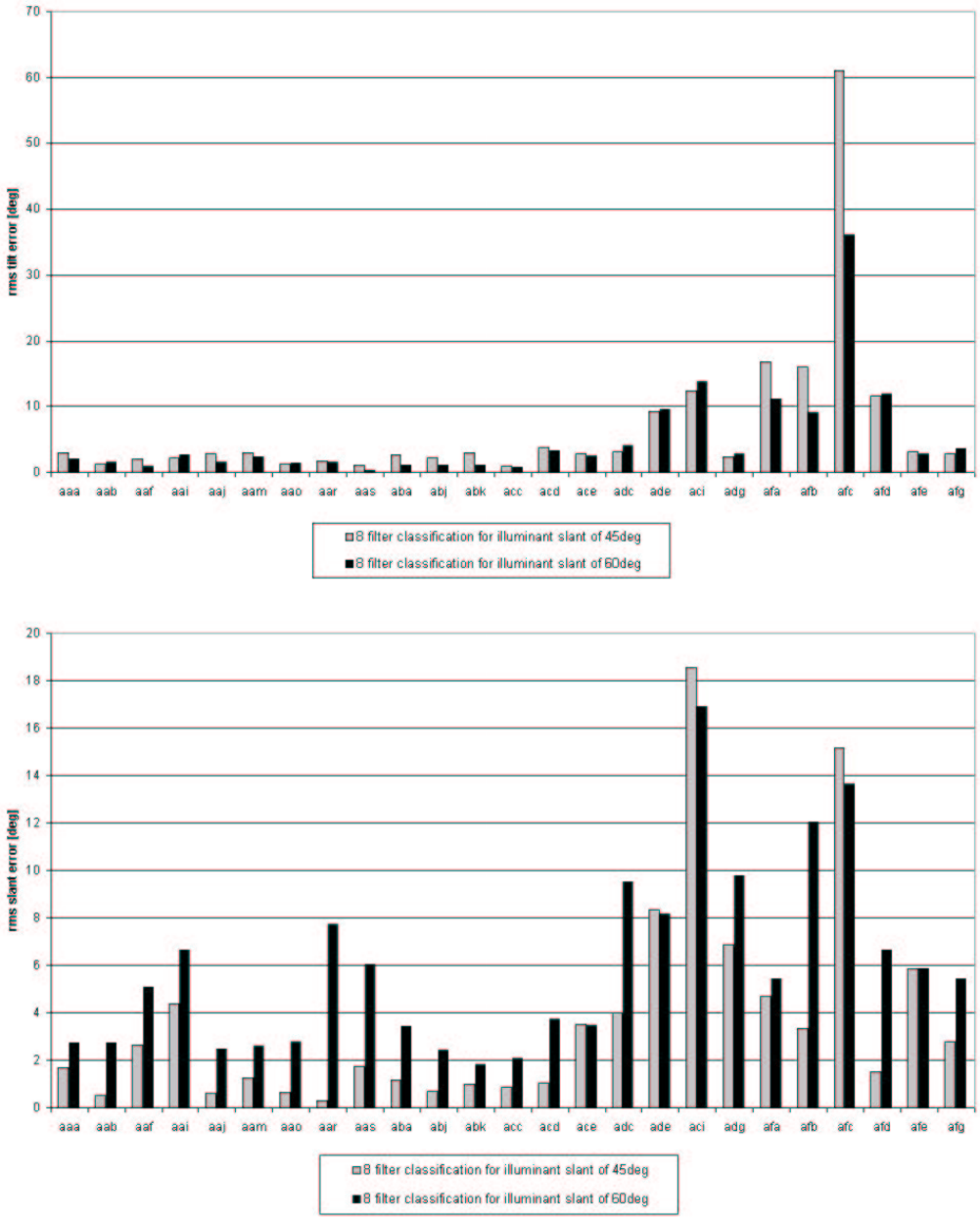


Figure 6.19: Root mean square tilt error (figure at the top) and rms slant error (figure at the bottom) for the tested surfaces of database 2

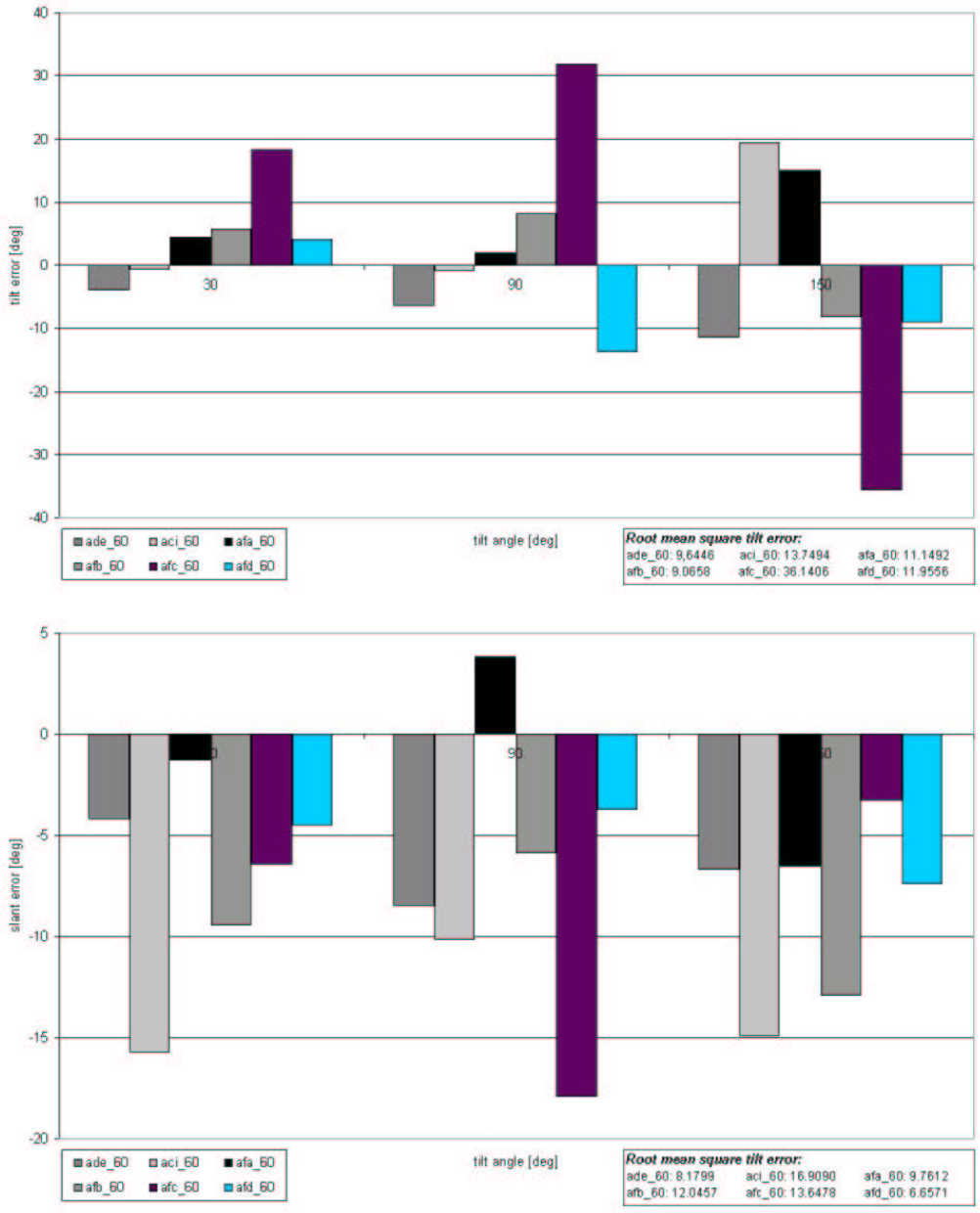


Figure 6.20: Tilt error (figure at the top) and slant error (figure at the bottom) for selected textures as a function of the tilt angle. The illuminated slant angle of the used textures is 60°

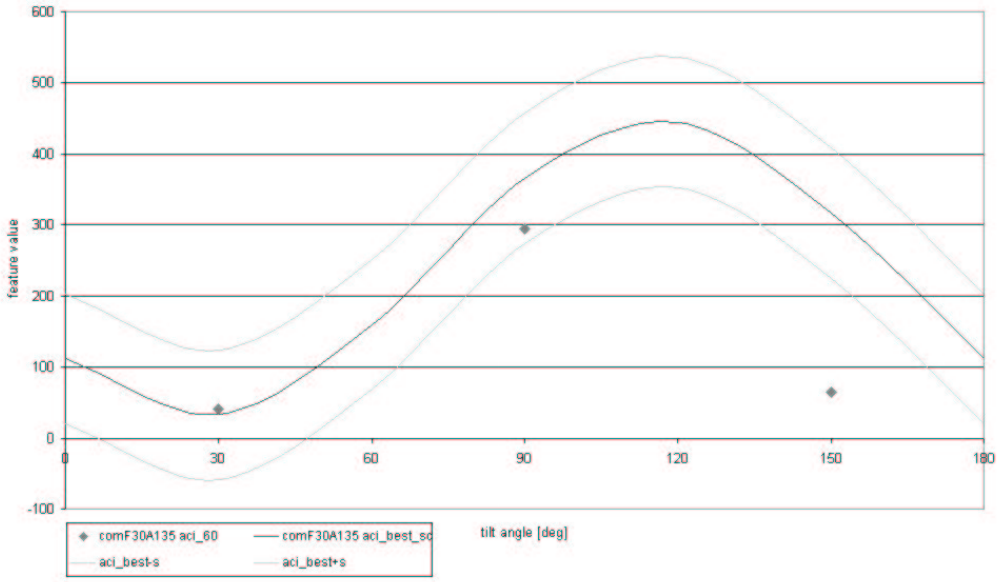


Figure 6.21: The behaviour of the texture *aci* filtered with *comF30A135* in feature space is shown.

where 68% of the feature values should be found. For the tilt angles 30° and 90° the feature values are in this area whereas the difference between the feature value for 150° of slant and the aequivalent best fit value is nearly two σ. We expect that the large difference of this feature value is the reason for the bad estimation of the illumination direction.

Figure 6.22 on the following page shows *afa_60* filtered with the Gabor filter *comF20A135*. We recognize that the best fit curve represents the feature values very bad. This is due to the lack that only six feature values are used to estimate the best fit curve (three feature values for every slant angle) and the feature value we got for 180° of tilt was neglegted. The problem that occures if this feature value is also used for the estimation of the best fit curve is that the influence of the best fit curve for 0° is twice compared to the other feature values.
The bad estimation results we got for *afa* causes to two different problems. The first is the number of training values. A rise in training values to estimate the best fit curve would rise the fit of the feature values to the predicted feature curve. The second problem is the noise of the image. In this case most of the information that is extracted by Gabor filter *F20A135*, is noise, so that the best fit curve is not able to describe the texture.

The same result was found for *afc*. The feature values we got for the

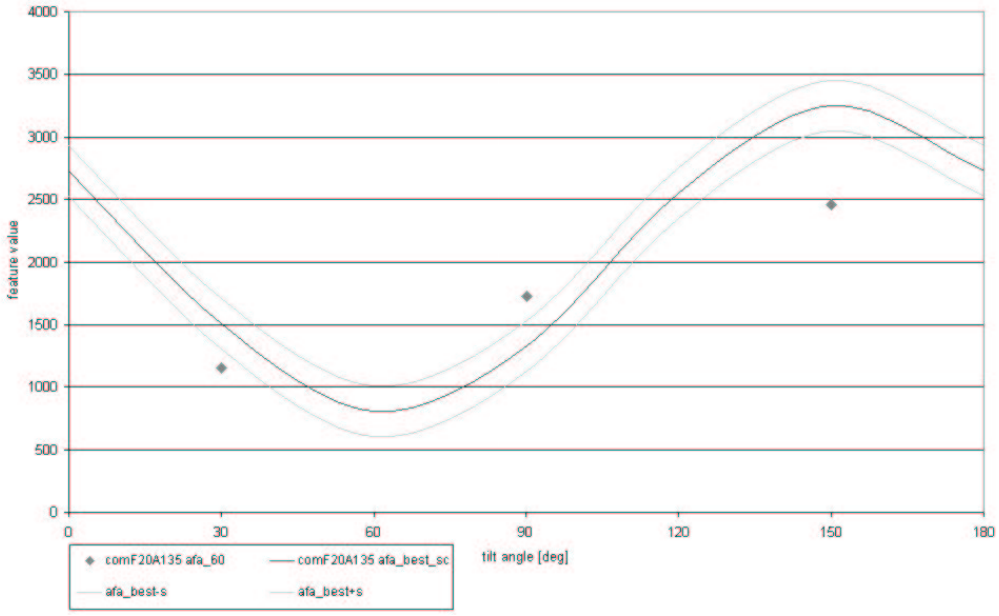


Figure 6.22: The behaviour of the texture *afa* filtered with *comF20A135* in feature space is shown.

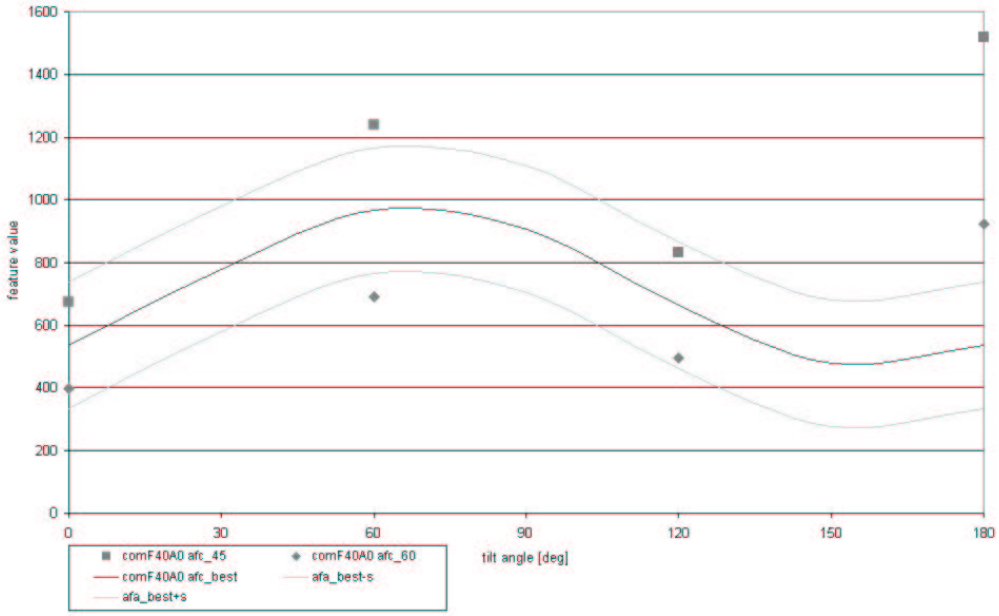


Figure 6.23: Feature values which where used for training and the predicted feature curve of *afc* is shown.

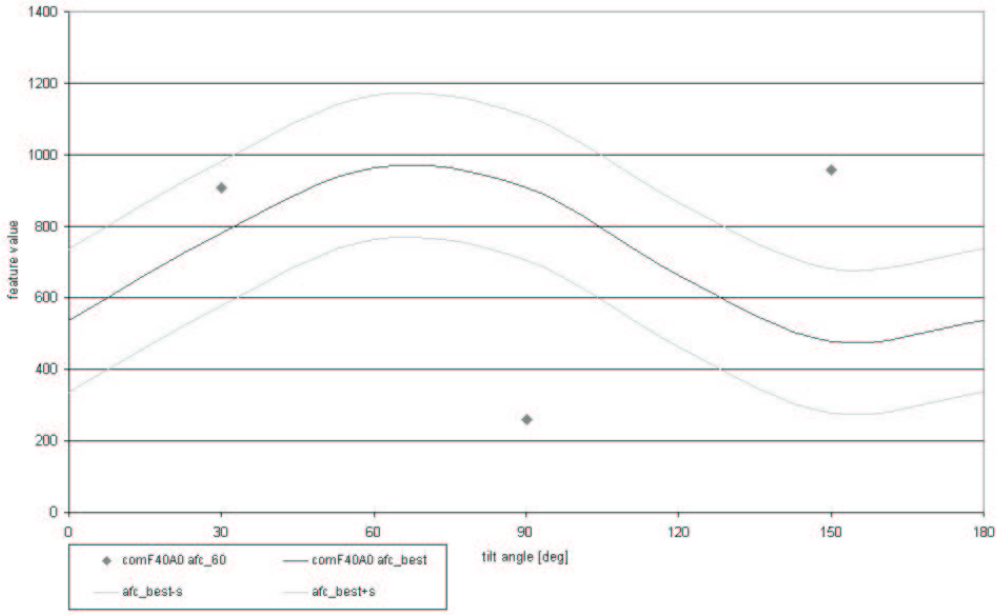


Figure 6.24: The behaviour of the texture *afc* filtered with *comF40A0* in feature space is shown.

tilt angles 0°, 60° and 120° and the slant angles 45° and 60° where used to estimate the best fit curve. Figure 6.23 on the page before shows the feature values and the corresponding feature curve in feature space. The values differ extremly so that the best fit curve cannot describe the behaviour of *afc* filtered with the Gabor filter *comF40A0* in feature space. We recognize a large difference between the feature values for a tilt angle of 0° and 180°. If the feature values would behave like a π-periodic sinosodial curve the feature values for 0° and 180° of tilt should be the same.

The feature values we used for classification are shown in figure 6.24. It can be seen that the prediction of the feature curve fails completely, so that the classifier is unable to classify the image with is illuminated with a illuminant tilt angle of 90°.

Chapter 7

Summary and Further Research

7.1 Summary

We have presented a new tilt invariant 3D surface texture classifier that exploits the hyper-elliptical behaviour of texture features. Using a Maximum Likelihood approach for discrimination the ML function could be transformed to a 4^{th} order polynomial in tilt (shown in C.1 on page 87). The classifier has been shown to perform well on 29 real images out of Database 1. We obtain a high classification accuracy and an accurate estimate of the tilt angle when the complete image is used.
Also an illumination direction invariant 3D texture classifier based on this method is presented. We investigated the illumination accuracy for 25 different real world textures in Database 2. Its discrimination function caused to a 12^{th} order polynomial in tilt and a 6^{th} order polynomial in slant. As expected the accuracy is lower than for tilt invariant classification, but if the range of the illumination slant angle is restricted the classifier also provides a high classification accuracy and quite good estimations of illumination tilt and slant. Finally, we investigated six textures in detail, which where inaccurately classified. Most of the misclassifications occured because of a very bad fit of a feature values to the corresponding best fit feature curves in at least one dimension of the feature space. If the feature value is heavily dependent on the noise in the image, a prediction of best fit curve fails, so that an accurate classification of this texture is no longer possible. We found that these textures contained shadows, which is responsible for the high value of the noise.

The requirements we made for the textures where

- Lambertian reflection,
- no self- or cast-shadowing,
- no interreflection,
- small slope angles and
- orthographic projection

If the filter bank is choosen for a set of textures, which comply with the requirements the classifier is able to classify illuminated textures with unknown illumination direction.

7.2 Further Research

This work invested a classification algorithm that is able to estimate the surface type and the position of the illumination source. Varying illumination slant angle reduced the accuracy of the tilt and slant estimates. For some textures we found big differences in the feature values we extracted for illumination tilt angles of 0° and 180°.
Is it possible to extend this classifier so that it is able to classify tilt angles greater than 180°? We think it is worthwhile to investigate this problem to get a classifier that is robust in illumination tilt and illumination slant angle. Another question which comes into mind is whether this approach can be used for pixel by pixel classification, and hence segmentation.

Appendix A

Results

A.1 Classification with known Illumination Slant Angle

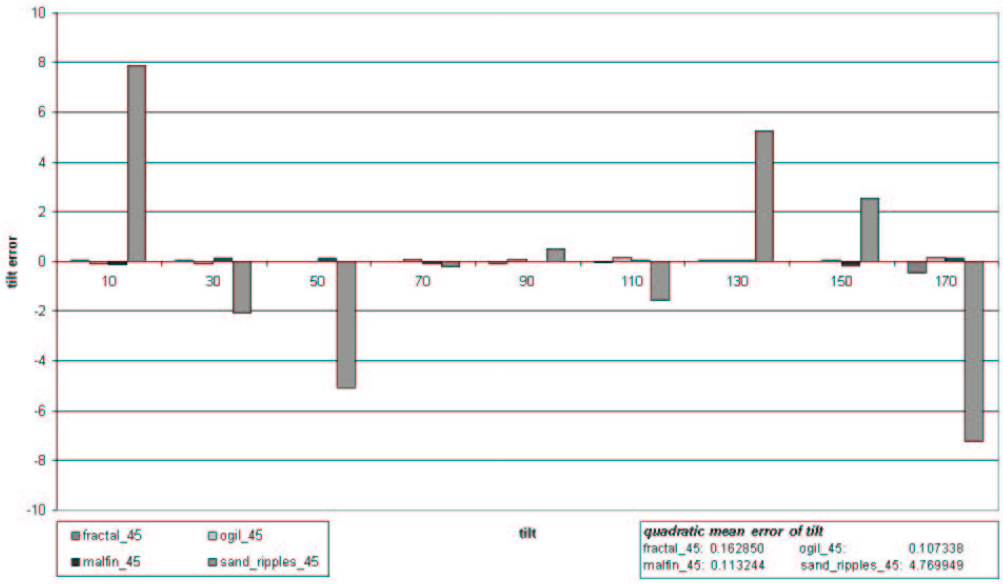


Figure A.1: Tilt error for four synthetic surfaces illuminated with a slant angle of 45° filtered with four different complex Gabor filters.

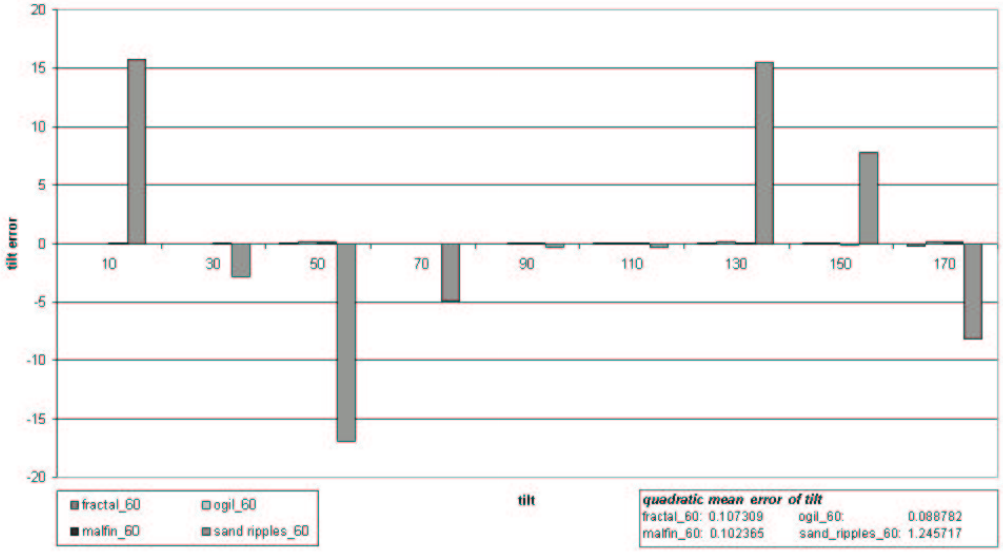


Figure A.2: Tilt error for four synthetic surfaces illuminated with a slant angle of 60° filtered with four different complex Gabor filters.

Input		Misclassification									
		9 filter		*6 filter*		*4 filter*		*3 filter*		*2 filter*	
surface	tilt	surface	tilt	surface	tilt	surface	tilt	surface	tilt	surface	tilt
chips1	30									stones2	138.30
chips1	90									and1	44.21
chips1	110									stones2	103.86
chips1	130							michael7	135.07	michael7	33.63
chips1	150							stones2	25.59	stones2	166.46
card1	50									wood	90.80
card1	90									radial45	87.95
beans1	130									and1	170.52
beans1	170									michael6	154.02
rock1	130									slab45	148.27
stones2	170					michael7	2.21				
iso45	90									michael5	19.67
iso45	110									wood	149.89
rock45	50							and4	51.036		
rock45	150									and5	52.60
slab45	70					rock1	18.56	rock1	6.13		
stri45	10									slab45	19.14
stri45	50									rock45	140.98
stri45	70									and5	161.60
stri45	90							and5	170.18	and5	170.06
stri45	110									and5	166.92
stri45	130									and5	155.66
stri45	150									and5	144.23
stri45	170							rock45	159.75	and5	59.50
twins45	90					iso45	89.88	iso45	95.87	iso45	100.44
wood	30									twins45	117.95
wood	150									iso45	113.29
michael1	10									michael2	14.25
michael1	90									michael2	104.88
michael1	110									michael2	113.46
michael1	170									michael2	22.82
michael2	30									michael1	33.27
michael2	50									michael1	44.41
michael2	70									michael1	122.25
michael2	130									michael1	127.88
michael2	170					michael8	179.69	michael8	162.35	michael8	15.87
michael4	70									and3	31.73
michael4	90									michael7	107.97
michael4	110									and3	28.49
michael6	50							and2	137.75	and2	41.74
michael6	70									and2	76.22
michael7	10							chips1	163.68	stones2	21.03
michael7	30									and3	17.79
michael7	50									and3	140.74
michael7	90									chips1	85.43
michael7	130									michael4	49.24
michael8	50									michael2	136.68
michael8	90									and3	100.98
michael8	150							michael2	158.31		
michael9	10									and2	154.48
michael9	90									and2	111.70
michael9	170									and2	156.01
and1	15									beans1	133.30
and1	165									beans1	129.72
and2	135									michael9	134.05
and3	165									chips1	16.16
and4	45									rock45	123.73
and4	75							rock45	26.03	rock45	25.95
and4	135									rock45	27.10
and4	165									rock45	56.85
and5	45									rock45	38.21
and5	165									michael5	46.78
and6	15							and2	35.90	michael9	138.06
and6	45							and2	43.76	michael9	133.43
and6	75							and2	54.43	and2	55.46
and6	105							and2	61.89	and2	62.02
and6	135							michael6	45.31	and2	42.75
and6	165							and2	20.65	and2	156.95
and7	15							and1	36.67	michael6	123.38
and7	45							and1	42.77	michael6	106.55
and7	75							and1	47.55	michael6	99.17
and7	105							chips1	46.44	beans1	117.89
and7	135							chips1	43.57	beans1	121.67
and7	165							michael6	67.74	michael6	123.71

Table A.1: Detection failures for tilt-classification of real textures (Database 1)

Input		Misclassification									
		8 filter		*6 filter*		*4 filter*		*3 filter*		*2 filter*	
surface	tilt	surface	tilt	surface	tilt	surface	tilt	surface	tilt	surface	tilt
aaa	150							aao	149.77		
aab	90									afb	70.32
aaf	30									aab	150.65
aaf	90									afb	81.56
aaf	150							aam	135.09	aab	46.25
aai	90									afd	137.92
aai	150									acc	83.047
aaj	90					ace	61.62	ace	61.61		
aam	90									afb	129.40
aar	150	aaf	157.67			aai	160.04				
aba	30									abk	155.56
abk	30					afa	173.87				
abk	90							aaj	100.92	afc	81.19
abk	150			aci	21.54	aci	21.54				
acc	90							aci	58.58		
acc	150			ace	153.12			ace	153.07	ace	153.07
acd	90							afa	128.87		
acd	150							aab	146.08		
ace	90					aci	72.92	afa	168.62	aaj	142.81
ade	30									acd	46.72
ade	90									acd	119.03
aci	30									aai	20.83
aci	90									afd	149.76
aci	150									aam	167.56
adc	90			aaa	103.51						
adc	150					aaa	122.48	aaa	133.12		
ade	30							aaa	155.88		
ade	90					afa	103.29	afa	103.42		
ade	150					afa	151.25	aab	134.85		
adg	30									aab	38.10
adg	90									aam	64.57
adg	150	aaf	3.74	afa	146.17	afa	164.83	afa	164.75		
afa	30									aam	149.66
afa	90	aaf	83.46	aaf	83.44						
afa	150							aam	137.86	afb	39.12
afb	30									aaf	54.11783
afb	150							aai	115.95	aaf	68.19
afc	30							afg	148.36	afg	25.97
afc	150									afg	149.05
afg	150							afc	54.11	afc	53.72

Table A.2: Misclassifications for classification of real textures with known tilt angle (Database 2)

A.2 Classification with unknown Illumination Direction

Input			Misclassification								
			8 filter			*4 filter*			*3 filter*		
surface	slant	tilt	surface	tilt	slant	surface	tilt	slant	surface	tilt	slant
ogil	75	50	malfin	34.47	67.32						
		130	malfin	143.31	74.69						
malfin	30	10							ogil	165.44	22.94
		50							ogil	59.76	29.96
		70							ogil	77.42	31.44
		90							ogil	97.04	32.54
		110				ogil	110.18	34.51	ogil	113.74	33.90
		170							ogil	148.45	28.67
malfin	45	10							ogil	169.12	32.18
		50							ogil	58.27	41.46
		70							ogil	75.81	44.67
		90							ogil	96.42	46.60
malfin	60	10							ogil	169.61	40.05
		50							ogil	57.81	52.20
		70							ogil	75.10	57.71
		90							ogil	95.97	60.90
malfin	75	10							ogil	168.42	42.28
		50							ogil	57.84	55.35
		70							ogil	74.34	62.28
		90							ogil	95.40	65.24
sand	30	10	fractal	2.88	37.11	fractal	80.87	30.82	fractal	53.08	38.47
		30	fractal	7.38	37.00	fractal	78.79	33.10	fractal	53.46	41.74
		50	fractal	31.82	36.87	fractal	77.99	36.75	fractal	53.73	47.38
		70				fractal	79.31	39.26	fractal	53.74	51.72
		90				fractal	83.12	39.51	fractal	53.86	52.49
		110				fractal	87.76	39.37	fractal	53.58	52.78
		130				fractal	91.37	37.13	fractal	52.79	49.55
		150	fractal	178.44	38.19	fractal	93.01	33.79	fractal	51.46	44.58
		170	fractal	179.27	38.56	fractal	91.01	31.77	fractal	50.21	41.67
sand	45	10				fractal	28.07	19.77	fractal	140.75	18.55
		30				fractal	53.85	21.38	fractal	134.36	19.87
		50	fractal	8.84	30.88	fractal	69.46	25.62	fractal	123.24	23.63
		70	fractal	60.05	30.88	fractal	75.31	31.37	fractal	55.43	37.83
		90	fractal	79.55	31.98	fractal	82.16	33.51	fractal	55.83	41.27
		110	fractal	66.23	30.16	fractal	91.92	31.48	fractal	55.28	38.93
		130	fractal	177.52	31.66	fractal	102.69	25.99	fractal	128.74	25.87
		150				fractal	147.86	22.41	fractal	137.26	22.02
		170				fractal	174.33	21.04	fractal	143.21	20.14
sand	60	10				fractal	10.61	18.24	fractal	147.11	16.79
		30				fractal	29.58	17.70	fractal	143.48	15.75
		50	fractal	5.24	26.49	fractal	54.65	18.26	fractal	131.79	16.15
		70	fractal	63.04	24.18	fractal	72.50	22.66	fractal	119.83	20.72
		90	fractal	83.80	26.70	fractal	82.57	26.53	fractal	117.74	25.08
		110	fractal	120.14	23.53	fractal	97.33	22.95	fractal	125.93	22.53
		130	fractal	177.43	26.61	fractal	126.06	18.43	fractal	133.58	18.34
		150				fractal	153.48	17.34	fractal	140.24	16.92
		170				fractal	175.84	17.65	fractal	144.69	16.73
sand	75	10				fractal	40.40	21.48	fractal	138.35	20.75
		30	fractal	2.92	30.85	fractal	43.25	19.71	fractal	138.20	17.82
		50	fractal	6.95	24.25	fractal	54.33	17.01	fractal	130.05	14.56
		70	fractal	71.04	19.11	fractal	73.96	16.83	fractal	119.87	15.22
		90	fractal	85.25	20.98	fractal	83.63	19.81	fractal	120.08	18.72
		110	fractal	103.95	18.29	fractal	102.62	17.36	fractal	125.81	17.00
		130	fractal	175.19	23.56	fractal	122.35	16.95	fractal	130.94	16.85
		150	fractal	178.43	29.78	fractal	123.60	18.52	fractal	133.43	18.46
		170				fractal	81.84	19.72	fractal	134.22	19.69

Table A.3: Misclassifications for synthetic surfaces with unknown illumination direction

Input			**Misclassification**								
			8 filter			*6 filter*			*4 filter*		
surface	slant	tilt	surface	tilt	slant	surface	tilt	slant	surface	tilt	slant
aaf	45	30							aab	34.54	44.57
aaf		90							aab	92.87	48.33
aaf	60	30							aab	34.87	50.88
aaf		90							aab	95.78	59.06
aai	45	30							ace	70.89	53.50
aai		90							aab	100.81	28.49
aai	60	90							aab	102.67	41.57
aaj	45	30							abk	72.54	62.77
aaj	60	30				acc	74.85	32.29	acc	72.67	31.80
aas	45	30							aaj	70.63	60.73
aas		150							aar	141.44	32.99
aas	60	30							aar	38.20	42.93
aas		150							aar	143.62	44.17
abk	45	150							abj	137.00	55.01
acc	60	150							ace	153.47	72.10
ade	60	90							aaa	91.12	34.83
aci	45	30				aao	25.56	21.80	aam	26.96	35.78
aci		90							aar	83.83	46.15
aci		150							aab	160.05	39.27
aci	60	30							acc	60.54	28.30
aci		90							aar	84.47	38.25
aci		150				aaa	160.10	25.80	aab	161.77	31.16
adg	45	30							aaa	17.74	32.91
adg		150							aaa	166.30	32.86
adg	60	30	aaa	23.31	33.30				aaa	20.99	33.77
adg		90							aaf	92.20	38.36
adg		150							aaa	163.13	32.64
afa	45	30							aai	37.93	57.17
afa		90							aaa	91.49	35.97
afa		150							aab	124.89	34.24
afa	60	90							aaa	91.15	39.78
afa		150				adg	138.17	59.91	aab	126.20	44.38
afb	45	30							acd	95.30	42.51
afb	60	30							acd	100.24	47.06
afc	45	30							afd	11.99	53.32
afc		90							aaj	0.40	16.03
afc		150							aaj	176.54	20.14
afc	60	30							aaj	11.48	20.00
afc		90							afg	74.55	27.88
afc		150							aaj	174.61	19.53
afg	45	90				aaj	95.49	28.40	aaj	95.44	28.44
afg	60	90	aab	75.77	16.37	aaj	95.84	34.65	aaj	95.74	34.78
afg		150	aab	164.01	15.21				aaj	161.20	28.70

Table A.4: Misclassifications for real textures with unknown illumination direction (Database 2)

Surface	rms tilt error for a slant angle of 45° in [°]									
	12 filter		*10 filter*		*8 filter*		*6 filter*		*4filter*	
	tilt	slant	tilt	slant	tilt	slant	tilt	slant	tilt	slant
aaa	2.5058	1.6977	2.3745	1.7016	2.9349	1.6943	3.0691	1.9960	2.9348	2.2799
aab	1.4625	0.4717	1.4196	0.5670	1.3187	0.5355	1.3449	0.3582	1.6039	0.9100
aaf	2.1585	2.6816	2.3556	2.7367	2.0229	2.6396	2.0358	2.4940	2.1140	2.9011
aai	1.8148	4.1489	1.9329	4.2058	2.2008	4.3838	1.9732	4.1665	1.1913	5.1940
aaj	3.0362	0.6945	3.2045	0.6789	2.8185	0.5931	3.1311	0.6012	3.5663	0.5242
aam	2.8111	1.2596	3.0850	1.3221	3.0445	1.2567	2.7157	1.4237	2.9459	1.4896
aao	1.4440	0.7245	1.3474	0.6799	1.2984	0.6349	1.3637	0.9470	1.3149	0.9511
aar	0.9698	0.2816	1.2620	0.4169	1.6842	0.3033	1.7161	0.2037	1.8477	0.6023
aas	1.0194	1.6533	1.1069	1.6514	1.0277	1.7442	0.6760	1.7832	1.2040	1.5169
aba	2.2890	1.0749	2.1834	1.1556	2.6982	1.1608	2.8698	1.1276	2.5682	1.0740
abj	2.0018	0.6167	2.1475	0.6195	2.2033	0.6866	2.2913	0.6756	2.0235	0.7482
abk	2.8439	0.9935	2.8532	1.0310	2.9776	0.9822	3.1596	1.1135	2.8643	1.1357
acc	1.0184	0.6457	1.0175	0.8211	0.9046	0.8815	1.9089	0.8040	1.4359	1.0861
acd	2.2726	1.1774	3.9341	1.2380	3.8135	1.0649	3.2974	0.9262	3.2019	0.8680
ace	2.8489	3.6902	3.0392	3.7293	2.8752	3.5384	3.0044	3.4922	3.8365	3.4050
adc	3.1598	4.2440	2.9426	4.1829	3.1065	3.9856	3.3137	3.9568	5.0669	4.0649
ade	8.6625	8.7405	8.8085	8.3377	9.3090	8.3283	10.1771	9.4639	10.0490	8.2476
aci	11.4665	18.3652	11.5176	18.1480	12.3333	18.5694	13.7327	18.6390	10.3734	18.6923
adg	3.4272	7.1020	3.6540	7.2620	2.4213	6.8824	2.8224	6.9640	2.2109	7.5875
afa	11.8456	3.2650	14.1289	4.3732	16.8266	4.7017	6.2162	2.6347	5.4636	6.0080
afb	16.7096	3.5492	14.2708	3.3542	15.9853	3.3301	17.2819	3.5958	15.9369	3.1811
afc	49.3845	13.2968	26.7390	12.5114	61.0358	15.1828	62.2004	14.1883	32.6011	14.6886
afd	10.5053	1.1356	11.0395	1.3060	11.6106	1.5115	11.2811	1.4667	11.1093	2.5070
afe	2.7722	5.7677	1.8605	5.7926	3.1202	5.8723	19.6186	5.3687	23.1347	6.5506
afg	3.6296	2.4082	2.6740	2.9617	2.7829	2.7988	2.3464	1.9072	1.8374	2.1462

Table A.5:
Root mean square tilt error for real textures imaged with illuminant slant of 45°

Surface	rms error for a slant angle of 60° in [°]									
	12 filter		*10 filter*		*8 filter*		*6 filter*		*4filter*	
	tilt	slant	tilt	slant	tilt	slant	tilt	slant	tilt	slant
aaa	2.2389	3.0129	1.9664	3.0427	1.9833	2.7580	2.2244	3.1658	1.9564	2.7873
aab	1.4056	2.2852	1.6452	2.9447	1.6447	2.7237	1.4432	2.4361	1.7639	3.9293
aaf	1.0166	4.9549	1.1328	4.9153	1.0122	5.0739	1.0797	4.9715	1.5361	4.9053
aai	2.4483	6.2362	2.4670	6.6181	2.6021	6.6721	2.2024	5.9881	3.4651	7.3465
aaj	1.7778	2.5846	1.8116	2.5684	1.5742	2.4583	1.8230	2.5649	1.8239	2.4117
aam	2.1146	2.9561	2.3723	2.7485	2.2798	2.6261	2.0810	3.9551	1.9293	3.4807
aao	1.7552	3.0040	1.4509	2.5762	1.4686	2.7838	2.0235	2.7773	1.9014	2.4774
aar	0.9543	7.1845	0.7532	6.3866	1.5130	7.7588	1.4019	7.6745	1.0653	7.8667
aas	0.9530	4.5405	0.9129	4.4165	0.3655	6.0396	0.5214	6.0411	0.8075	5.1948
aba	1.1243	3.1594	1.2499	3.2904	1.0625	3.4183	1.4835	3.3120	1.0586	4.1122
abj	1.1988	2.6938	1.0187	2.5167	1.0418	2.4515	0.9860	2.2538	1.1850	2.1707
abk	1.0835	1.7657	1.1435	1.7106	1.0658	1.8179	1.1118	1.9719	1.1333	1.8887
acc	7.1894	6.1409	0.9045	2.1876	0.7569	2.0789	0.4872	1.3132	1.2426	2.0174
acd	1.9338	3.1933	3.2455	3.7328	3.3344	3.7447	2.9040	4.1871	2.1373	3.9324
ace	2.5805	3.0429	2.3222	3.1368	2.5654	3.4782	2.1095	3.1145	2.5616	3.5818
adc	3.9993	9.7878	3.9454	9.7115	4.1468	9.5012	4.3247	9.3073	4.6972	11.1921
ade	9.3253	7.7012	10.2892	7.3599	9.6446	8.1799	9.7794	7.6298	9.8940	7.5061
aci	13.0791	16.6148	12.8807	16.8343	13.7494	16.9091	15.1953	16.5688	12.2683	15.7164
adg	2.1677	9.5084	2.0568	9.3895	2.7976	9.7612	2.2507	9.6844	3.5784	10.0291
afa	8.8818	3.4974	10.2194	5.8342	11.1492	5.4316	7.2850	3.1405	5.1460	6.3683
afb	64.0338	43.9560	8.8722	12.3348	9.0658	12.0457	9.2015	12.1997	7.9007	11.2342
afc	28.0386	12.7437	26.5639	12.7386	36.1406	13.6478	36.5367	12.9718	27.4462	15.4636
afd	10.1824	6.6437	10.6965	6.8109	11.9556	6.6571	11.1152	7.3170	12.0337	7.6198
afe	2.5942	5.8429	2.9652	5.5169	2.8115	5.8529	1.6260	6.2477	3.9036	5.5781
afg	4.8767	4.8010	3.6128	5.9253	3.5509	5.4142	3.3213	4.1502	3.7644	4.2111

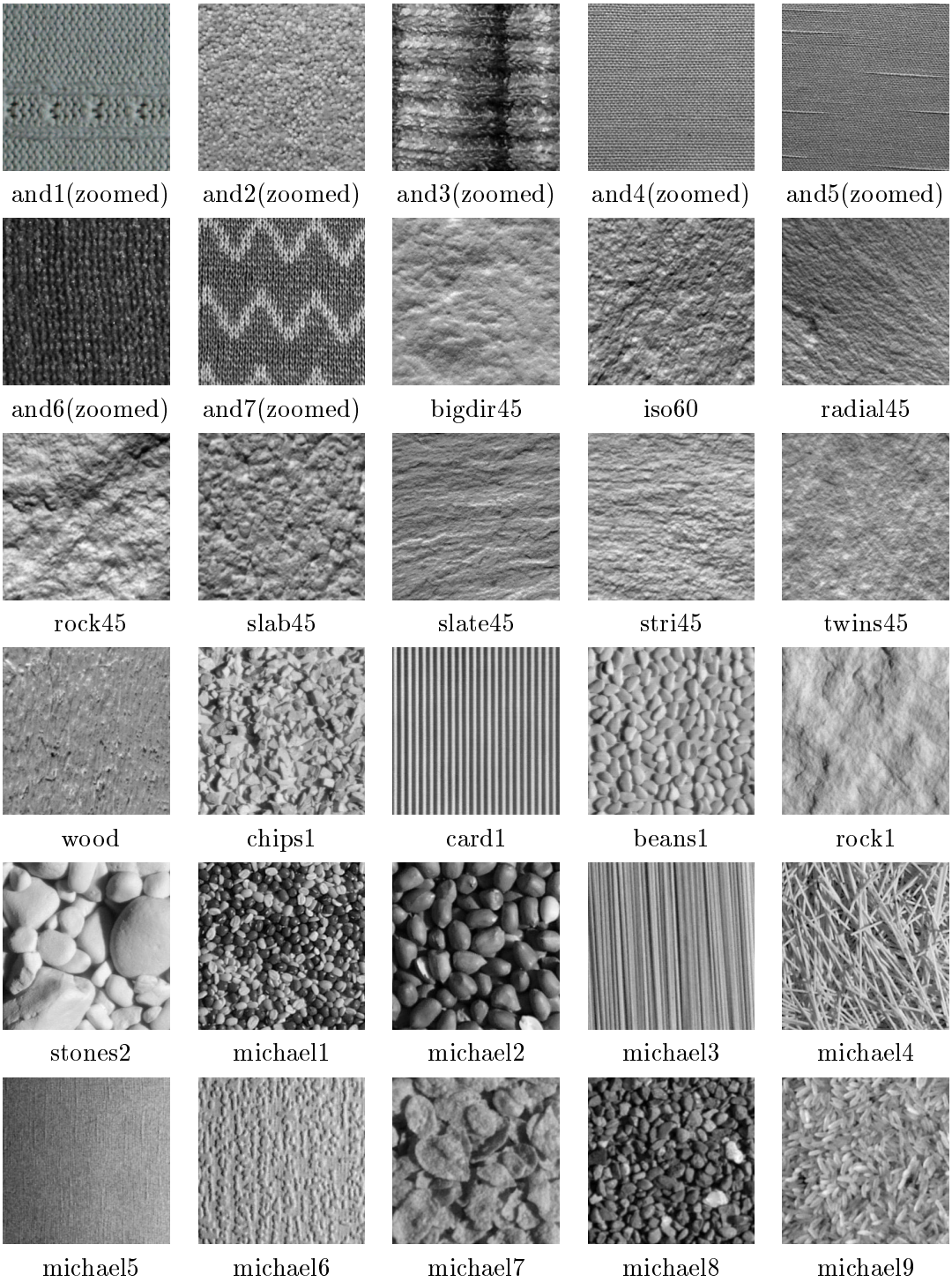
Table A.6: Root mean square tilt error for real textures imaged with illuminant slant of 60°

Appendix B

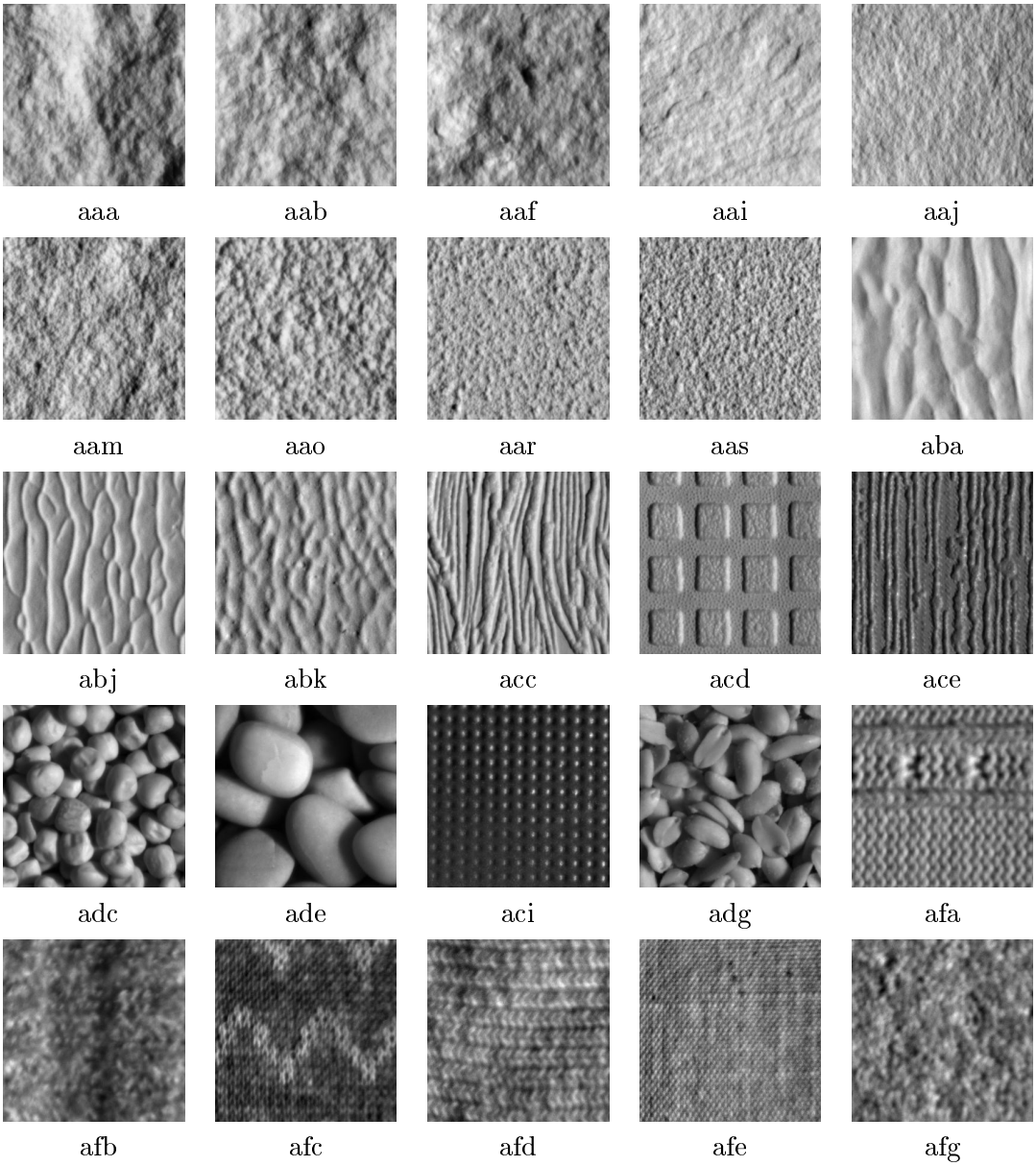
Texture Databases

For this student research project two different texture databases are used. The first database (database 1) is used for classification of tilt. 29 different textures are imaged with an illumination tilt angle between 0° and 180° and an illumination slant angle of 45°. The difference of the illumination tilt angle of two consecutive images is 10° except the data sets *and1* to *and7*, which are imaged in 15° steps. Because of the fixed slant angle this database can't be used for classification of tilt and slant.
The second database (database 2) contains 25 different textures imaged with an illumination tilt angle between 0° and 180° and illumination slant angles of 45° and 60°. The difference of the illumination tilt angle of two consecutive images is 30°.

B.1 Database 1

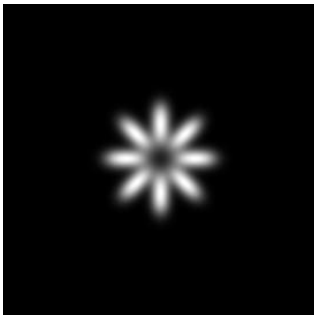


B.2 Database 2

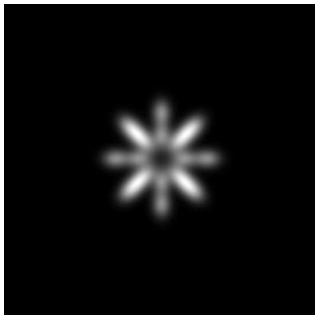


B.3 Filter Bank

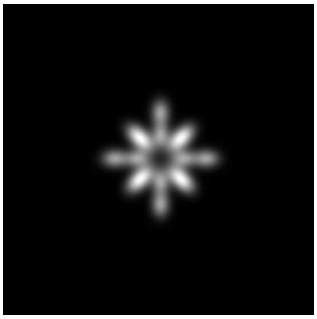
The filtered area in frequency domain dependent on the filter bank is shown (zoomed). The filters used for a specific filterbank can be found in the tables 2.1 on page 12 and 2.2 on page 13 where the particular filters are discussed in detail.



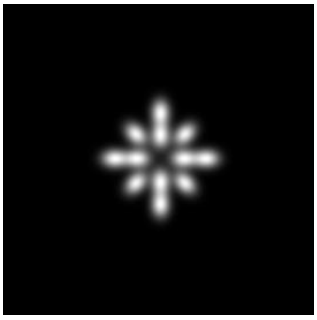
12 Gabor filters



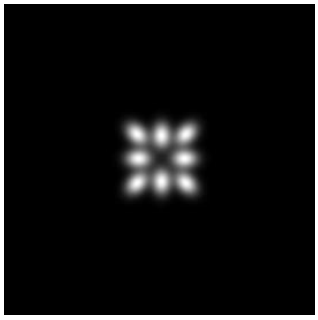
10 Gabor filters



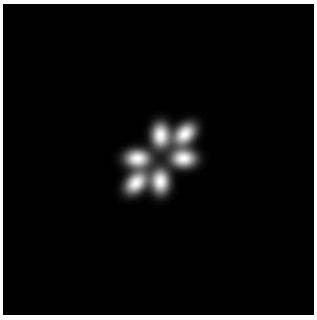
8 Gabor filters



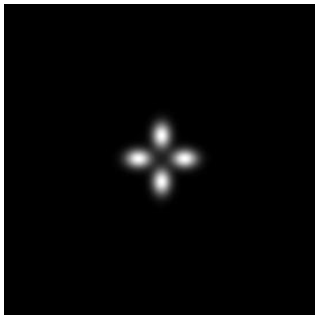
6 Gabor filters



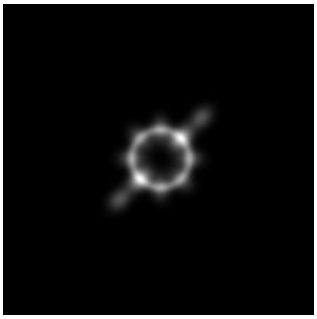
Gabor 4 filters



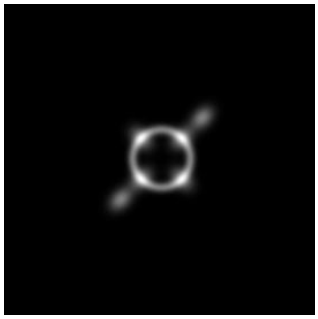
3 Gabor filters



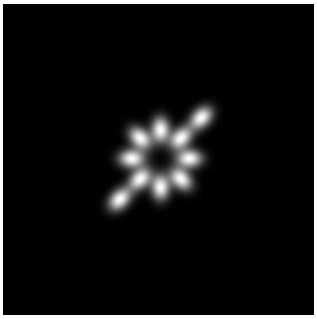
2 Gabor filters



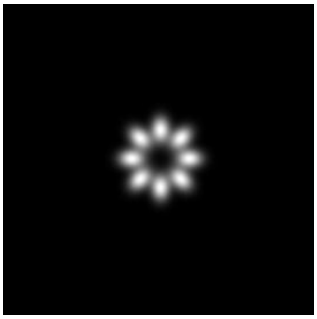
9 Mixed filters



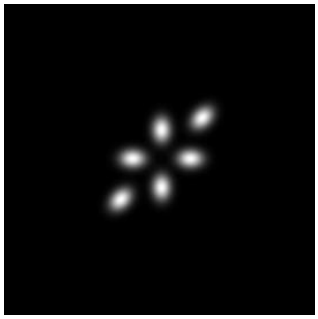
6 Mixed filters



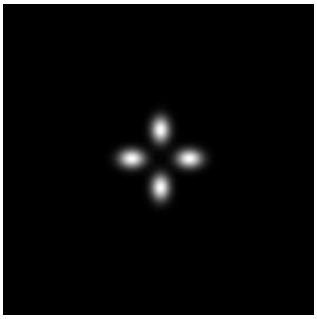
5 Mixed filters



4 Mixed filters



3 Mixed filters



2 Mixed filters

Appendix C

Mathematical Models

The discrimination functions we use for classification are exponential. In general the extrema of nonlinear functions are solved numerically. In this section we proof, that the maximum of the functions we use can be found solving polimomial functions. We transform the nonlinear problem to polynomial functions, where the extrema can simply calculated by finding its roots. The disadvantage of the transformation is that because of separating the sums the numerical error rises. The advantage is the reduced computal cost (no explicit numerical method like Newton algorithm or Constant Gradient algorithm is used). The next two sections present the transformations of the discrimination functions to polynomials.

C.1 Estimation of Tilt

The estimation of tilt is a simplified problem of estimation of tilt and slant, so that the discrimination function to estimate tilt and slant can also be used for estimation of tilt. Because of its complexity and the rised error in this section we derive the transformation of the tilt estimation function.

Starting with the linear prediction

$$Y_i(\tau) = (d_i + e_i cos(2\tau) + f_i sin(2\tau)) \tag{C.1}$$

assuming Gaussian variation of the sum of errors

$$f(t) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2s^2}}$$

The Maximum Likelihood method is used to estimate the most probably surface. We get the following equation:

$$f(\tau,\theta) = \prod_i \frac{1}{s\sqrt{2\pi}} e^{-\frac{(y_i-(d_i+e_i cos(2\tau)+f_i sin(2\tau)))^2}{2s^2}}$$

Finding the Maximum Likelihood estimator means that we need to find the maximum of $f(\tau,\theta)$. Because ln is a linear operator, in the same way we can maximize $\ln f(\tau,\theta)$.

$$\ln f(\tau) = \ln \prod_i (\frac{1}{s_i\sqrt{2\pi}}) + \sum_i \frac{(y_i - d_i - e_i cos(2\tau) - f_i sin(2\tau))^2}{2s_i^2}$$

Calculating the first derivation of $\ln f_i(\tau)$ gives:

$$\begin{aligned}
&\frac{\delta}{\delta\tau}\ln f_i(\tau) = \\
&= \sum_i \frac{1}{2s_i^2}(2(y_i - d_i - e_i cos(2\tau) - f_i sin(2\tau)) \cdot 2(e_i \sin(2\tau) - f_i \cos(2\tau)) \\
&= \sum_i \frac{1}{2s_i^2}[2e_i(y_i - d_i)\sin(2\tau) - 2f_i(y_i - d_i)\cos(2\tau) \\
&\quad -(2e_i^2 - 2f_i^2)\sin(2\tau)\cos(2\tau) - 2e_i f_i sin^2(2\tau) + 2e_i f_i \cos^2(2\tau)]
\end{aligned}$$

Using the following abbrevations:

$$\begin{aligned}
A &= \sum_i \frac{1}{s_i^2}[2e_i(y_i - d_i)] \\
B &= \sum_i \frac{1}{s_i^2}[2f_i(d_i - y_i)] \\
C &= \sum_i \frac{1}{s_i^2}[2(f_i^2 - e_i^2)] \\
D &= \sum_i \frac{1}{s_i^2}[-4e_i\, f_i] \\
E &= \sum_i \frac{1}{s_i^2}[2e_i\, f_i]
\end{aligned}$$

Substitution of $\cos^2(2\tau) = 1 - \sin^2(2\tau)$, we get:

$$0 = A\sin(2\tau) + B\cos(2\tau) + C\sin(2\tau)\cos(2\tau) + D\sin^2(2\tau) + E$$

The term $\cos(2\tau)$ can be substituted by:

$$\begin{aligned}
\sin^2(2\tau) &= X \\
\cos(2\tau) &= \pm\sqrt{1-X^2} \\
&\quad - \textbf{if } -\frac{\pi}{2} \leq 2\tau \leq \frac{\pi}{2} \\
&\quad + \textbf{if } +\frac{\pi}{2} \leq 2\tau \leq \frac{3\pi}{2}
\end{aligned}$$

So we get:

$$\pm\sqrt{1-X^2}(B + C\,X) = -A\,X - D\,X^2 - E$$

Calculating the square of the equation we can derive the left side of the formula as:

$$\begin{aligned}
&(\pm\sqrt{1-X^2})^2(B + C\,X)^2 = \\
&\quad = (1-X^2)(B^2 + 2B\,C\,X + C^2X^2) \\
&\quad = B^2 + 2B\,C\,X + C^2X^2 - B^2X^2 - 2B\,C\,X^3 - C^2X^4
\end{aligned}$$

For the right side of the formula we get:

$$\begin{aligned}
&(-A\,X - D\,X^2 - E)^2 = \\
&\quad = A^2X^2 + 2A\,X(D\,X^2 + E) + (D\,X^2 + E)^2 \\
&\quad = A^2X^2 + 2A\,D\,X^3 + 2A\,E\,X + D^2X^4 + 2D\,E\,X^2 + E^2
\end{aligned}$$

Arranging the whole equation in X we get the result:

$$\begin{aligned}
0 \;=\; & X^4(C^2 - D^2) + X^3(2A\,D + 2B\,C) \\
& + X^2(A^2 + 2D\,E + B^2 - C^2) + X(2A\,E - 2B\,C) + (E^2 - B^2)
\end{aligned}$$

Calculating the roots of this function and reinserting the possible solutions in the linear prediction, we get the maximum choosing the solution, which maximizes the function.

C.2 Estimation of Tilt and Slant

This section shows one possible solution of the two dimensional ML function, which is used to classify textures and estimate the illumination tilt and slant angle. The derivation of equation 6.12 on page 52 can be found in chapter 6. This section shows the transformation of of the ML function to a 12^{th} order polinom. The advantage of the polynomial function is its simplicity of evaluate the maximum. For the Maximum Likelihood function of our tilt-slant classifier we found the following equation:

$$f(\tau,\phi) = \prod_i \frac{1}{\sqrt{2\pi}\sigma_i} \exp - \frac{(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\phi)))^2}{2\sigma_i^2}$$

In order to find its maximum, we can maximize the natural log of $f(\tau,\phi)$.

$$\ln f(\tau,\phi) = \textstyle\sum_i \ln(\frac{1}{\sqrt{2\pi}\sigma_i}) - \sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\phi))^2$$

$$\begin{aligned} \ln f(\tau,\phi) - \sum_i \ln(\frac{1}{\sqrt{2\pi}\sigma_i}) &= -\sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\phi))^2 \\ &= \rightarrow max \end{aligned}$$

Using the abbrevation:

$$\ln f(\tau,\phi) - \sum_i \ln(\frac{1}{\sqrt{2\pi}\sigma_i}) = g(\tau,\phi)$$

gives:

$$g(\tau,\phi) = \sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau))^2$$

In order to find the maximum of $g(\tau,\phi)$ as a function of tilt(τ) and slant(σ) we need to calculate the first and second partial derivates. The necessary conditions for extrema are:

$$\begin{aligned} g_\tau(\tau,\sigma) &= 0 \\ g_\sigma(\tau,\sigma) &= 0 \end{aligned}$$

The sufficient conditions for extrema can be found by calculating the following determinant:

$$\Delta = \begin{vmatrix} g_{\tau\tau}(\tau,\sigma) & g_{\tau\sigma}(\tau,\sigma) \\ g_{\tau\sigma}(\tau,\sigma) & g_{\sigma\sigma}(\tau,\sigma) \end{vmatrix}$$

Calculating the determinant, we get the type of the extrema:

$$\begin{array}{lcl} \Delta > 0 & \Rightarrow & \begin{cases} g_{\tau\tau}(\tau,\sigma) < 0 \text{ maximum} \\ g_{\tau\tau}(\tau,\sigma) > 0 \text{ minimum} \end{cases} \\ \Delta < 0 & \Rightarrow & \text{no extremum} \\ \Delta = 0 & \Rightarrow & \text{more complex discussion is needed} \end{array}$$

To reduce the computal cost we calculate the zero points of the partial derivates $g_\tau(\tau,\sigma)$ and $g_\sigma(\tau,\sigma)$ and check the maximum by inserting the possible solutions for tilt and slant into $g(\tau,\sigma)$.

Calculation of $g_\tau(\tau,\sigma) = 0$:

$$\begin{aligned} g_\tau(\tau,\sigma) &= 0 \\ \frac{\delta g(\tau,\phi)}{\delta\tau} &= \frac{\delta}{\delta\tau}\sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau)))^2 \\ &= \sum_i \frac{1}{2\sigma_i} 2(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau))) \\ &\quad \cdot(-\sin^2(\phi)(-2b_i\sin(2\tau) + 2c_i\cos(2\tau))) \\ &= \sum_i \frac{1}{\sigma_i}[2b_i y_i \sin^2(\phi)\sin(2\tau) - 2c_i yi\sin^2(\phi)\cos(2\tau) \\ &\quad -2b_i\sin^4(\phi)\sin(2\tau)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau)) \\ &\quad +2c_i\sin^4(\phi)\cos(2\tau)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau)) \\ &= \sum_i \frac{1}{\sigma_i}[2b_i y_i \sin^2(\phi)\sin(2\tau) - 2c_i y_i\sin^2(\phi)\cos(2\tau) \\ &\quad -2a_i b_i\sin^4(\phi)\sin(2\tau) - 2b_i^2\sin^4(\phi)\sin(2\tau)\cos(2\tau) \\ &\quad -2b_i c_i\sin^4(\phi)\sin^2(2\tau) + 2a_i c_i\sin^4(\phi)\cos(2\tau) \\ &\quad +2b_i c_i\sin^4(\phi)\cos^2(2\tau) \\ &\quad +2c_i^2\sin^4(\phi)\cos(2\tau)sin(2\tau)] \\ &= \sum_i \frac{1}{\sigma_i}[2b_i\sin^2(\phi)\sin(2\tau)(y_i - a_i\sin^2(\phi)) \\ &\quad +2c_i\sin^2(\phi)(a_i\sin^2(\phi) - y_i)\cos(2\tau) \\ &\quad +2\sin^4(\phi)\sin(2\tau)\cos(2\tau)(c_i^2 - b_i^2) \\ &\quad +(-4b_i c_i\sin^4(\phi)\sin^2(2\tau) + 2b_i c_i\sin^4(\phi))] \end{aligned}$$

Substitution of

$$\begin{aligned} Y &= \sin^2(\phi) \\ X &= \sin(2\tau) \\ \pm\sqrt{1-X^2} &= \cos(2\tau) \end{aligned}$$

gives:

$$\pm\sqrt{1-X^2}\sum_i \frac{1}{\sigma_i}[2Yc_iy_i - 2Y^2a_ic_i + 2X\,Y^2b_i^2 - 2X\,Y^2c_i^2] =$$

$$= \sum_i \frac{1}{\sigma_i}[2X\,Y\,b_iy_i - 2X\,Y^2a_ib_i + 2Y^2b_ic_i - 4X^2Y^2b_ic_i]$$

The equation can be simplified using the following abbrevations:

$$\begin{aligned} MAA &= \sum_i \frac{1}{\sigma_i}a_i^2 \\ MAB &= \sum_i \frac{1}{\sigma_i}a_ib_i \\ MAC &= \sum_i \frac{1}{\sigma_i}a_ic_i \\ MAY &= \sum_i \frac{1}{\sigma_i}a_iy_i \\ MBB &= \sum_i \frac{1}{\sigma_i}b_i^2 \\ MBC &= \sum_i \frac{1}{\sigma_i}b_ic_i \\ MBY &= \sum_i \frac{1}{\sigma_i}b_iy_i \\ MCC &= \sum_i \frac{1}{\sigma_i}c_i^2 \\ MCY &= \sum_i \frac{1}{\sigma_i}c_iy_i \\ MYY &= \sum_i \frac{1}{\sigma_i}y_i^2 \end{aligned}$$

$$\pm\sqrt{1-X^2}\cdot[2Y\,MCY - 2Y^2MAC + 2XY^2MBB - 2X\,Y^2MCC]$$

$$= [2X\,Y\,MBY - 2XY^2MAB + 2Y^2MBC - 4X^2Y^2MBC]$$

$$\begin{aligned} & \pm\sqrt{1-X^2}\cdot[Y(2MCY) + Y^2(X(2MBB - 2MCC)) - 2MAC)] \\ = \; & [Y(2X\,MBY) + Y^2(X^2(-4MBC) + X(-2MAB) + 2MBC)] \end{aligned}$$

Calculating the square of the first part of the formula gives:

$$\begin{aligned} & (\pm\sqrt{1-X^2}\cdot[Y(2MCY) + Y^2(X(2MBB - 2MCC)) - 2MAC)])^2 \\ = \; & (1-X^2)[4Y^2MCY^2 + 2Y^3(2MCY)(X(2MBB - 2MCC) - 2MAC) \\ & +Y^4(X(2MBB - 2MCC) - 2MAC)^2] \\ = \; & Y^4[(1-X^2)(4X^2(MBB - MCC)^2 \\ & -8X\,MAC(MBB - MCC) + 4MAC^2)] \\ & +Y^3[(1-X^2)(8X\,MCY(MBB - MCC) - 8MCY\,MAC)] \\ & +Y^2[(1-X^2)(4MCY^2)] \\ = \; & Y^4[-4X^4(MBB - MCC)^2 + 8X^3MAC(MBB - MCC) \\ & +X^2(4(MBB - MCC)^2 - 4MAC^2) \\ & -8X\,MAC(MBB - MCC) + 4MAC^2] \\ & +Y^3[-8X^3MCY(MBB - MCC) + 8X^2MCY\,MAC \\ & +8X\,MCY(MBB - MCC) \\ & -8MCY\,MAC] \\ & +Y^2[-4X^2MCY^2 + 4MCY^2] \\ = \; & X^4[-4Y^4(MBB - MCC)^2 + X^3(8Y^4MAC(MBB - MCC) \\ & -8Y^3MCY(MBB - MCC)) + X^2(4Y^4((MBB - MCC)^2 - MAC^2) \\ & +8Y^3MCY\,MAC - 4Y^2MCY^2) + X(-8Y^4MAC(MBB - MCC) \\ & +8Y^3MCY(MBB - MCC)) + (4Y^4MAC^2 \\ & -8Y^3MCY\,MAC + 4Y^2MCY^2)] \end{aligned}$$

$$\begin{aligned} & [Y(2X\,MBY) + Y^2(X^2(-4MBC) + X(-2MAB) + 2MBC)]^2 \\ = \; & 4X^2Y^2MBY^2 - 16X^3Y^3MBY\,MBC - 8X^2Y^3MBY\,MAB \\ & +8X\,Y^3MBY\,MBC + 16X^4Y^4MBC^2 + 16X^3Y^4MBC\,MAB \end{aligned}$$

$$
\begin{aligned}
& -16X^2Y^4MBC^2 + 4X^2Y^4MAB^2 - 8XY^4MAB\,MBC + 4Y^4MBC^2 \\
= \; & X^4[16Y^4MBC^2] + X^3[-16Y^3MBY\,MBC + 16Y^4MBC\,MAB] \\
& + X^2[4Y^2MBY^2 - 8Y^3MBY\,MAB - 16Y^4MBC^2 + 4Y^4MAB^2] \\
& + X[8Y^3MBY\,MBC - 8Y^4MAB\,MBC] \\
& + 4Y^4MBC^2
\end{aligned}
$$

Recalcuate the equation above gives:

$$
\begin{aligned}
0 \; = \; & X^4Y^4[16MBC^2 + 4(MBB - MCC)^2] \\
& + X^3[Y^4(16MBC\,MAB - 8MAC(MBB - MCC)) \\
& + Y^3(8MCY(MBB - MCC) \\
& - 16MBY\,MBC)] \\
& + X^2[Y^4(-16MBC^2 + 4MAB^2 + 4MAC^2 - 4(MBB - MCC)^2) \\
& + Y^3(-8MBY\,MAB - 8MCY\,MAC) \\
& + Y^2(4MBY^2 + 4MCY^2)] \\
& + X[Y^4(-8MAB\,MBC + 8MAC(MBB - MCC)) \\
& + Y^3(8MBY\,MBC - 8MCY(MBB - MCC))] \\
& + [Y^4(4MBC^2 - 4MAC^2) + Y^3(8MAC\,MCY) + Y^2(-4MCY^2)]
\end{aligned}
$$

Substitution of

$$
\begin{aligned}
VA &= 16MBC^2 + 4(MBB - MCC)^2 \\
VB &= 16MBC\,MAB - 8MAC(MBB - MCC) \\
VC &= 8MCY(MBB - MCC) - 16MBY\,MBC \\
VD &= -16MBC^2 + 4MAB^2 + 4MAC^2 - 4(MBB - MCC)^2 \\
VE &= -8MBY\,MAB - 8MCY\,MAC \\
VF &= 4MBY^2 + 4MCY^2 \\
VG &= -8MAB\,MBC + 8MAC(MBB - MCC) \\
VH &= 8MBY\,MBC - 8MCY(MBB - MCC) \\
VI &= 4MBC^2 - 4MAC^2 \\
VK &= 8MAC\,MCY \\
VL &= -4MCY^2
\end{aligned}
$$

gives:

$$\begin{aligned} 0 &= X^4(Y^4VA) + X^3(Y^4VB + Y^3VC) \\ &\quad + X^2(Y^4VD + Y^3VE + Y^2VF) + X(Y^4VG + Y^3VH) \\ &\quad + (Y^4VI + Y^3VK + Y^2VL) \\ &= Y^4(X^4VA + X^3VB + X^2VD + XVG + VI) \\ &\quad + Y^3(X^3VC + X^2VE + XVH + VK) + Y^2(X^2VF + VL) \end{aligned} \tag{C.2}$$

divide by Y^2 $\quad (\sin^2(\phi) \neq 0 \quad \forall \phi \in]0, \frac{\pi}{2}[\ )$

$$\begin{aligned} 0 &= Y^2(X^4VA + X^3VB + X^2VD + XVG + VI) \\ &\quad + Y(X^3VC + X^2VE + XVH + VK) + (X^2VF + VL) \end{aligned}$$

Calculation of $g_\sigma(\tau, \sigma) = 0$:

$$\begin{aligned} g_\phi(\tau, \phi) &= 0 \\ &= \sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)(a_i + b_i\cos(2\tau) + c_i\sin(2\tau)))^2 \end{aligned}$$

Substitution of

$$H_i(\tau) = (a_i + b_i\cos(2\tau) + c_i\sin(2\tau))$$

gives:

$$\begin{aligned} g_\phi(\tau, \phi) &= \sum_i \frac{1}{2\sigma_i^2}(y_i - \sin^2(\phi)H_i(\tau))^2 \\ \frac{\delta g(\tau, \phi)}{\delta \phi} &= \sum_i \frac{1}{2\sigma_i^2}(2(y_i - \sin^2(\phi)H_i(\tau))(-2\sin(\phi)\cos(\phi)H_i(\tau))) \\ &= 2\sin(\phi)\cos(\phi)\sum_i \frac{1}{\sigma_i^2}(\sin^2(\phi)H_i^2(\tau) - y_iH_i(\tau)) \end{aligned}$$

Because $2 \cdot \sin(\phi)\cos(\phi) = 0 \quad \forall \phi = n\frac{\pi}{2} \quad n \in N$
we get fixed points at $\phi = 0°$ and $\phi = 90°$
with:

$$\begin{aligned} H_i(\tau) &= (a_i + b_i\cos(2\tau) + c_i\sin(2\tau)) \\ \sum_i \frac{1}{\sigma_i^2} y_i H_i(\tau) &= MAY + \cos(2\tau)MBY + \sin(2\tau)MCY \end{aligned}$$

$$\begin{aligned} H_i^2(\tau) &= a_i^2 + 2a_ib_i\cos(2\tau) + 2a_ic_i\sin(2\tau) \\ &\quad + b_i^2\cos^2(2\tau) + 2b_ic_i\cos(2\tau)\sin(2\tau) + c_i^2\sin^2(2\tau) \\ \sum_i \frac{1}{\sigma_i^2} H_i^2(\tau) &= MAA + 2\cos(2\tau)MAB + 2\sin(2\tau)MAC + \cos^2(2\tau)MBB \\ &= +2\cos(2\tau)\sin(2\tau)MBC + \sin^2(2\tau)MCC \end{aligned}$$

$$\begin{aligned} \sin^2(\phi) &= \frac{\sum_i \frac{1}{\sigma_i^2} y_i H_i(\tau)}{\sum_i \frac{1}{\sigma_i^2} H_i^2(\tau)} \\ &= \frac{MAY + \cos(2\tau)MBY + \sin(2\tau)MCY}{\overline{MAA + 2\cos(2\tau)MAB + 2\sin(2\tau)MAC + \cos^2(2\tau)MBB \\ +2\cos(2\tau)\sin(2\tau)MBC + \sin^2(2\tau)MCC}} \end{aligned}$$

Subst: $X = sin(2\tau)$ and $\pm\sqrt{1-X^2} = \cos(2\tau)$

$$\begin{aligned} \sin^2(\phi) &= \\ &= \frac{MAY \pm \sqrt{1-X^2}MBY + X\,MCY}{\overline{MAA \pm 2\sqrt{1-X^2}MAB + 2X\,MAC + (1-X^2)MBB \\ \pm 2X\sqrt{1-X^2}MBC + X^2MCC}} \\ &= \frac{MAY \pm \sqrt{1-X^2}MBY + XMCY}{\overline{(MAA + MBB) + 2X\,MAC + X^2(MCC - MBB) \\ \pm\sqrt{1-X^2}(2X\,MBC + 2MAB)}} \end{aligned} \tag{C.3}$$

Substitution of

$$Y = \sin^2(2\tau)$$

$$\begin{aligned} Y_n &= MAY \pm \sqrt{1-X^2}MBY + X\,MCY \\ Y_d &= (MAA + MBB) + 2X\,MAC + X^2(MCC - MBB) \\ &\quad \pm\sqrt{1-X^2}(2X\,MBC + 2MAB) \\ Y &= \frac{Y_n}{Y_d} \end{aligned}$$

and inserting equation C.3 into equation C.2

$$\begin{aligned} 0 &= \frac{Y_n^2}{Y_d^2}(X^4VA + X^3VB + X^2VD + X\,VG + VI) \\ &\quad + \frac{Y_n}{Y_d}(X^3VC + X^2VE + X\,VH + VK) \\ &\quad + (X^2VF + VL) \end{aligned}$$

Multiply by Y_d gives:

$$\begin{aligned} 0 &= Y_n^2(X^4VA + X^3VB + X^2VD + X\,VG + VI) \\ &\quad + Y_nY_d(X^3VC + X^2VE + X\,VH + VK) \\ &\quad + Y_d^2(X^2VF + VL) \end{aligned} \tag{C.4}$$

Calculation of Y_n^2:

$$\begin{aligned} Y_n^2 &= (MAY \pm \sqrt{1-X^2}MBY + X\,MCY)^2 \\ &= MAY^2 \pm 2MAY\sqrt{1-X^2}MBY + 2XMAY\,MCY \\ &\quad + (1-X^2)MBY^2 \pm 2\sqrt{1-X^2}X\,MBY\,MCY + X^2MCY^2 \\ &= X^2[-MBY^2 + MCY^2] + X[2MAY\,MCY] \\ &\quad \pm\sqrt{1-X^2}[2MAY\,MBY + 2X\,MBY\,MCY] + [MAY^2 + MBY^2] \end{aligned}$$

Substitution of

$$\begin{aligned} WA &= -MBY^2 + MCY^2 \\ WB &= 2MAY\,MCY \\ WC &= MAY^2 + MBY^2 \end{aligned}$$

gives:

$$Y_n^2 = X^2 WA + X\,WB + WC \pm \sqrt{1-X^2}[2MAY\,MBY + 2X\,MBY\,MCY]$$

Calculation of $Y_n Y_d$:

$$\begin{aligned}
Y_n Y_d &= (MAY \pm \sqrt{1-X^2}MBY + X\,MCY)((MAA+MBB) + 2X\,MAC \\
&\quad +X^2(MCC-MBB) \pm \sqrt{1-X^2}(2X\,MBC + 2MAB)) \\
&= \pm\sqrt{1-X^2}[2MAY\,MAB + 2X\,MAY\,MBC + MBY\,MAA \\
&\quad +2X\,MBY\,MAC + X^2 MBY\,MCC + (1-X^2)MBY\,MBB \\
&\quad +2X\,MCY(MAB + XMBC)] \\
&\quad +X^2[MAY\,MCC - MAY\,MBB - 2MBY\,MAB + 2MAC\,MCY] \\
&\quad +X^3[-2MBY\,MBC + MCC\,MCY - MBB\,MCY] \\
&\quad +X[2MAY\,MAC + 2MBY\,MBC + MAA\,MCY + MBB\,MCY] \\
&\quad +[MAA\,MAY + MAY\,MBB + 2MBY\,MAB]
\end{aligned}$$

Subst:

$$\begin{aligned}
WD &= MAY\,MCC - MAY\,MBB - 2MBY\,MAB + 2MAC\,MCY \\
WE &= -2MBY\,MBC + MCC\,MCY - MBB\,MCY \\
WF &= 2MAY\,MAC + 2MBY\,MBC + MAA\,MCY + MBB\,MCY \\
WG &= MAA\,MAY + MAY\,MBB + 2MBY\,MAB
\end{aligned}$$

gives:

$$\begin{aligned}
Y_n Y_d &= X^3 WE + X^2 WD + X\,WF + WG \pm \sqrt{1-X^2}[2MAY\,MAB \\
&\quad +2X\,MAY\,MBC + MBY\,MAA + 2X\,MBY\,MAC \\
&\quad +X^2 MBY\,MCC + (1-X^2)MBY\,MBB \\
&\quad +2X\,MCY(MAB + X\,MBC)]
\end{aligned}$$

Calculation of Y_d^2:

$$\begin{aligned}
Y_d^2 &= [(MAA+MBB) + 2X\,MAC + X^2(MCC-MBB) \\
&\quad \pm\sqrt{1-X^2}(2X\,MBC + 2MAB)]^2
\end{aligned}$$

$$
\begin{aligned}
= \; & \pm\sqrt{1-X^2}[2(MAA+MBB)(2MAB+2X\,MBC) \\
& +4X\,MAC(2MAB+2X\,MBC) \\
& +2X^2(MCC-MBB)(2MAB+2X\,MBC)] \\
& +X^4[(MCC-MBB)^2-4MBC^2] \\
& +X^3[4MAC(MCC-MBB)-8MAB\,MBC] \\
& +X^2[2(MAA+MBB)(MCC-MBB) \\
& +4MAC^2+4MBC^2-4MAB^2] \\
& +X[4MAC(MAA+MBB)+8MAB\,MBC] \\
& +[(MAA+MBB)^2+4MAB^2]
\end{aligned}
$$

Subst:

$$
\begin{aligned}
WH &= (MCC-MBB)^2-4MBC^2 \\
WI &= 4MAC(MCC-MBB)-8MAB\,MBC \\
WK &= 2(MAA+MBB)(MCC-MBB)+4MAC^2 \\
&\quad +4MBC^2-4MAB^2] \\
WL &= 4MAC(MAA+MBB)+8MAB\,MBC \\
WM &= (MAA+MBB)^2+4MAB^2
\end{aligned}
$$

gives:

$$
\begin{aligned}
Y_d^2 = \; & X^4WH+X^3WI+X^2WK+X\,WL+WM \\
& \pm\sqrt{1-X^2}[X^3(4MBC(MCC-MBB)) \\
& +X^2(8MAC\,MBC+4MAB(MCC-MBB)) \\
& +X(4MBC(MAA+MBB)+8MAC\,MAB) \\
& +4MAB(MAA+MBB)]
\end{aligned}
$$

Rewrite equation C.4 using Y_n^2 , Y_nY_d and Y_d^2 gives:

$$
\begin{aligned}
0 = \; & (X^2WA+XWB \\
& +WC\pm\sqrt{1-X^2}[2MAY\,MBY+2X\,MBY\,MCY] \\
& \cdot[X^4VA+X^3VB+X^2VD+X\,VG+VI] \\
& +(X^3WE+X^2WD+X\,WF+WG\pm\sqrt{1-X^2} \\
& \cdot(X^2(MBY\,MCC-MBY\,MBB+2MCY\,MBC)
\end{aligned}
$$

$$
\begin{aligned}
&+X(2MAY\,MBC + 2MBY\,MAC + 2MCY\,MAB)\\
&+(2MAY\,MAB + MBY\,MAA + MBY\,MBB)))\\
&\cdot[X^3VC + X^2VE + X\,VH + VK]\\
&+(X^4WH + X^3WI + X^2WK + X\,WL + WM\\
&\pm\sqrt{1-X^2}(X^3(4MBC(MCC - MBB))\\
&+X^2(8MAC\,MBC + 4MAB(MCC - MBB))\\
&+X(4MBC(MAA + MBB) + 8MAC\,MAB)\\
&+4MAB(MAA + MBB)))[X^2VF + VL]
\end{aligned}
$$

Subst:

$$
\begin{aligned}
WN &= MBY\,MCC - MBY\,MBB + 2MCY\,MBC\\
WO &= 2MAY\,MBC + 2MBY\,MAC + 2MCY\,MAB\\
WP &= 2MAY\,MAB + MBY\,MAA + MBY\,MBB\\
WR &= 4MBC(MCC - MBB)\\
WS &= 8MAC\,MBC + 4MAB(MCC - MBB)\\
WT &= 4MBC(MAA + MBB) + 8MAC\,MAB\\
WU &= 4MAB(MAA + MBB)\\
WV &= 2MAY\,MBY\\
WW &= 2MBY\,MCY
\end{aligned}
$$

gives:

$$
\begin{aligned}
0 = {} & (X^2WA + X\,WB + WC \pm \sqrt{1-X^2}(WV + X\,WW)\\
&\cdot[X^4VA + X^3VB + X^2VD + X\,VG + VI]\\
&+(X^3WE + X^2WD + X\,WF + WG \pm \sqrt{1-X^2}\\
&\cdot(X^2WN + X\,WO + WP))[X^3VC + X^2VE + X\,VH + VK]\\
&+(X^4WH + X^3WI + X^2WK + X\,WL + WM\\
&\pm\sqrt{1-X^2}(X^3WR + X^2WS + X\,WT + WU))[X^2VF + VL]
\end{aligned}
$$

Separating gives:

$$
\begin{aligned}
& \pm\sqrt{1-X^2}[(WV+XWW)(X^4VA+X^3VB+X^2VD+XVG+VI) \\
& +(X^2WN+XWO+WP)(X^3VC+X^2VE+XVH+VK) \\
& +(X^3WR+X^2WS+XWT+WU)(X^2VF+VL)] \\
= \; & [(X^2WA+XWB+WC)(X^4VA+X^3VB+X^2VD+XVG+VI) \\
& +(X^3WE+X^2WD+XWF+WG)(X^3VC+X^2VE+XVH+VK) \\
& +(X^4WH+X^3WI+X^2WK+XWL+WM)(X^2VF+VL)]
\end{aligned}
$$

Extracting X of the left side of the equation gives:

$$
\begin{aligned}
& \pm\sqrt{1-X^2}[(WV+XWW)(X^4VA+X^3VB+X^2VD+XVG+VI) \\
& +(X^2WN+XWO+WP)(X^3VC+X^2VE+XVH+VK) \\
& +(X^3WR+X^2WS+XWT+WU)(X^2VF+VL)] \\
= \; & \pm\sqrt{1-X^2}[X^5(VAWW+VCWN+VFWR) \\
& +X^4(VAWV+VBWW+VEWN+VCWO+VFWS) \\
& +X^3(VBWV+VDWW+VHWN+VEWO \\
& +VCWP+VFWT+VLWR) \\
& +X^2(VDWV+VGWW+VKWN+VHWO \\
& +VEWP+VFWU+VLWS) \\
& +X(VGWV+VIWW+VKWO+VHWP+VLWT) \\
& +(VIWV+VKWP+VLWU)]
\end{aligned}
$$

Subst:

$$
\begin{aligned}
DA &= VAWW+VCWN+VFWR \\
DB &= VAWV+VBWW+VEWN+VCWO+VFWS \\
DC &= VBWV+VDWW+VHWN+VEWO+VCWP \\
&\quad +VFWT+VLWR \\
DD &= VDWV+VGWW+VKWN+VHWO+VEWP \\
&\quad +VFWU+VLWS \\
DE &= VGWV+VIWW+VKWO+VHWP+VLWT \\
DF &= VIWV+VKWP+VLWU
\end{aligned}
$$

gives:

$$= \pm\sqrt{1-X^2} \cdot [X^5 DA + X^4 DB + X^3 DC + X^2 DD + X\,DE + DF]$$

Calculating the square:

$$\begin{aligned}
&= (1-X^2)(X^5 DA + X^4 DB + X^3 DC + X^2 DD + X\,DE + DF)^2 \\
&= (1-X^2)[X^{10}(DA^2) + X^9(2DA\,DB) + X^8(2DA\,DC + DB^2) \\
&\quad + X^7(2DA\,DD + 2DB\,DC) + X^6(2DA\,DE + 2DB\,DD + DC^2) \\
&\quad + X^5(2DA\,DF + 2DB\,DE + 2DC\,DD) \\
&\quad + X^4(2DB\,DF + 2DC\,DE + DD^2) + X^3(2DC\,DF + 2DD\,DE) \\
&\quad + X^2(2DD\,DF + DE^2) + X(2DE\,DF) + DF^2]
\end{aligned}$$

Subst:

$$\begin{aligned}
EA &= DA^2 \\
EB &= 2DA\,DB \\
EC &= 2DA\,DC + DB^2 \\
ED &= 2DA\,DD + 2DB\,DC \\
EE &= 2DA\,DE + 2DB\,DD + DC^2 \\
EF &= 2DA\,DF + 2DB\,DE + 2DC\,DD \\
EG &= 2DB\,DF + 2DC\,DE + DD^2 \\
EH &= 2DC\,DF + 2DD\,DE \\
EI &= 2DD\,DF + DE^2 \\
EK &= 2DE\,DF \\
EL &= DF^2
\end{aligned}$$

gives:

$$\begin{aligned}
&= X^{12}(-EA) + X^{11}(-EB) + X^{10}(EA - EC) \\
&\quad + X^9(EB - ED) + X^8(EC - EE) + X^7(ED - EF) \\
&\quad + X^6(EE - EG) + X^5(EF - EH) + X^4(EG - EI) \\
&\quad + X^3(EH - EK) + X^2(EI - EL) + X(EK) + EL
\end{aligned}$$

Calculating the second part:

$$
\begin{aligned}
& (X^2WA + XWB + WC)(X^4VA + X^3VB + X^2VD + XVG + VI) \\
& +(X^3WE + X^2WD + XWF + WG)(X^3VC + X^2VE + XVH + VK) \\
& +(X^4WH + X^3WI + X^2WK + XWL + WM)(X^2VF + VL) \\
= \; & X^6(WAVA + WEVC + WHVF) \\
& +X^5(WAVB + WBVA + WEVE + WDVC + WIVF) \\
& +X^4(WAVD + WBVB + WCVA + WEVH + WDVE + WFVC \\
& +WKVF + WHVL) \\
& +X^3(WAVG + WBVD + WCVB + WEVK + WDVH + WFVE \\
& +WGVC + WLVF + WIVL) \\
& +X^2(WAVI + WBVG + WCVD + WDVK + WFVH + WGVE \\
& +WMVF + WKVL) \\
& +X(WBVI + WCVG + WFVK + WGVH + WLVL) \\
& +(WCVI + WGVK + WMVL)
\end{aligned}
$$

Subst:

$$
\begin{aligned}
GA &= WAVA + WEVC + WHVF \\
GB &= WAVB + WBVA + WEVE + WDVC + WIVF \\
GC &= WAVD + WBVB + WCVA + WEVH + WDVE \\
&\quad +WFVC + WKVF + WHVL \\
GD &= WAVG + WBVD + WCVB + WEVK + WDVH \\
&\quad +WFVE + WGVC + WLVF + WIVL) \\
GE &= WAVI + WBVG + WCVD + WDVK + WFVH \\
&\quad +WGVE + WMVF + WKVL \\
GF &= WBVI + WCVG + WFVK + WGVH + WLVL \\
GG &= WCVI + WGVK + WMVL
\end{aligned}
$$

gives:

$$= X^6GA + X^5GB + X^4GC + X^3GD + X^2GE + XGF + GG$$

Calculating the square:

$$
\begin{aligned}
& (X^6 GA + X^5 GB + X^4 GC + X^3 GD + X^2 GE + X\,GF + GG)^2 \\
= \;& X^{12}(GA^2) + X^{11}(2GA\,GB) + x^{10}(2GA\,GC + GB^2) \\
& + X^9(2GA\,GD + 2GB\,GC) + X^8(2GA\,GE + 2GB\,GD + GC^2) \\
& + X^7(2GA\,GF + 2GB\,GE + 2GC\,GD) \\
& + X^6(2GA\,GG + 2GB\,GF + 2GC\,GE + GD^2) \\
& + X^5(2GB\,GG + 2GC\,GF + 2GD\,GE) \\
& + X^4(2GC\,GG + 2GD\,GF + GE^2) \\
& + X^3(2GD\,GG + 2GE\,GF) \\
& + X^2(2GE\,GG + GF^2) \\
& + X(2GF\,GG) \\
& + GG^2
\end{aligned}
$$

Subst:

$$
\begin{aligned}
HA &= GA^2 \\
HB &= 2GA\,GB \\
HC &= 2GA\,GC + GB^2 \\
HD &= 2GA\,GD + 2GB\,GC \\
HE &= 2GA\,GE + 2GB\,GD + GC^2 \\
HF &= 2GA\,GF + 2GB\,GE + 2GC\,GD \\
HG &= 2GA\,GG + 2GB\,GF + 2GC\,GE + GD^2 \\
HH &= 2GB\,GG + 2GC\,GF + 2GD\,GE \\
HI &= 2GC\,GG + 2GD\,GF + GE^2 \\
HK &= 2GD\,GG + 2GE\,GF \\
HL &= 2GE\,GG + GF^2 \\
HM &= 2GF\,GG \\
HN &= GG^2
\end{aligned}
$$

gives:

$$
\begin{aligned}
= \;& X^{12}HA + X^{11}HB + X^{10}HC + X^9 HD + X^8 HE + X^7 HF + X^6 HG \\
& + X^5 HH + X^4 HI + X^3 HK + X^2 HL + X\,HM + HN \qquad \text{(C.5)} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(C.6)}
\end{aligned}
$$

Rewriting equations C.5 on page 100 and C.5 into one equation

$$\begin{aligned} 0 \;=\; & X^{12}(HA+EA)+X^{11}(HB+EB) \\ & +X^{10}(HC+EC-EA)+X^{9}(HD+ED-EB) \\ & +X^{8}(HE+EE-EC)+X^{7}(HF+EF-ED) \\ & +X^{6}(HG+EG-EE)+X^{5}(HH+EH-EF) \\ & +X^{4}(HI+EI-EG)+X^{3}(HK+EK-EH) \\ & +X^{2}(HL+EL-EI)+X(HM-EK)+(HN-EL) \end{aligned}$$

Appendix D

Scripts

D.1 C-Programs

C-Programs in /u1/ee3/ceeap/cpplinux

best_fit_dbase

Calculates the parameters a, b, c and s of the function

$$a + b * \cos(x) + c * \sin(x)$$

which fits the function given by the input values best. Where a is the mean, b and c are the coefficients and s is the variance of the best fit curve. The input values must cover exactly one period.

```
/*
 * NAME
 *
 *      best_fit_dbase
 *
 * SYNOPSIS
 *      Calculates the parameters a, b ,c and s of the
 *      function a+b*cos(x)+c*sin(x) which fits the slant
 *      corrected functions given by the input values best.
 *      a is the mean, b and c are the coefficients and s
 *      is the standard-deviation of the best fit curve.
 *      The input values must cover exactly one period.
 *
```

```
 * AUTHOR
 *      Andreas Penirschke
 */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define PI 3.141592654

/* FAILURE HANDLING NOT DISPLAYED */

/* USAGE: */
/* [VALUES 1]...[VALUES N][SLANT 1]...[SLANT N][LENGTH OF VALUES I]
main(argc,argv)
char *argv[];
int argc;
{
   /* DEFINE VARIABLES*/
   int i,j, fun_val, test, slant_num;
   float a=0.0, b=0.0, c=0.0, s=0.0, tmp;

   /* NUMBER OF FEATURE VALUES PER SLANT ANGLE */
   fun_val=atoi(argv[argc-1]);

   /* NUMBER OF DIFFERENT SLANT ANGLES */
   slant_num=(int) ((argc-2)/(fun_val+1));

   /* CALCULATE THE BEST FIT PARAMETERS */
   for(i=0;i<(fun_val-1);i++)
   {
      for(j=0;j<slant_num;j++)
      {
         tmp=atof(argv[i+1+(j*fun_val)]);
         tmp=tmp/(sin(PI/180*atof(argv[argc-1-slant_num+j]))\
              *sin(PI/180*atof(argv[argc-1-slant_num+j])));
         a+=tmp;
         b+=tmp*cos(2*PI*(double) i/(double) (fun_val-1));
         c+=tmp*sin(2*PI*(double) i/(double) (fun_val-1));
      }
   }
   a=a/(double)(argc-2-(2*slant_num));
   b=2*b/(double)(argc-2-(2*slant_num));
   c=2*c/(double)(argc-2-(2*slant_num));
```

```
   /* CALCULATE THE STANDART DEVIATION S: */
   for(i=0;i<(fun_val-1);i++)
   {
      for(j=0;j<slant_num;j++)
      {
         tmp=atof(argv[i+1+(j*fun_val)]);
         tmp=tmp/(sin(PI/180*atof(argv[argc-1-slant_num+j]))\
              *sin(PI/180*atof(argv[argc-1-slant_num+j])));
         tmp=(tmp-(a+b*cos(2*PI*(double) i/(double) (fun_val-1))\
              +c*sin(2*PI*(double) i/(double) (fun_val-1))));
         s+=tmp*tmp;
      }
   }
   s=sqrt(s/(double)(argc-2-(2*slant_num)));

   /* WRITE RESULTS BACK */
   printf("%.10f %.10f %.10f %.10f ", a, b, c, s);
   exit(0);
}
```

D.2 Shell-Scripts

Shellscripts in /u1/ee3/ceeap/shellscripts

feature_generation

For every surface the features of images taken under different tilt angles are computed. Every image is filtered with 12 different complex Gabor filters in the frequency domain, then transformed back into spatial domain. The mean of the variance of each filtered image is calculated and stored as feature vectors for each surface.

```
set -x # DEBUG VERSION

# SET PATHS
a=/net/delos/usr/local/imaging/bin
c=/net/lxtexture1/spare/6/Invest6
d=~ceeap/experiments/slant_angle/gabor_filters
output=dbase_real # NAME OF THE OUTPUT FILE

# INIT THE OUTPUT FILE WHICH IS READ BY EXCEL
echo \"`date`\" > ${name} #quotes are needed for Excel

# WRITE THE HEADER FOR THE OUTPUT FILE
echo surface slant filter 0 30 60 90 120 150 180 >> ${name}

# CREATING GABOR FILTERS IN THE FREQUENCY DOMAIN
for phase in 0 45 90 135
do
   gab4vinnie3-4 -f20 -a$phase -p1 -n512 gaborF20A${phase}_re
   gab4vinnie3-4 -f20 -a$phase -p-1 -n512 gaborF20A${phase}_im
   gab4vinnie3-4 -f30 -a$phase -p1 -n512 gaborF30A${phase}_re
   gab4vinnie3-4 -f30 -a$phase -p-1 -n512 gaborF30A${phase}_im
   gab4vinnie3-4 -f40 -a$phase -p1 -n512 gaborF40A${phase}_re
   gab4vinnie3-4 -f40 -a$phase -p-1 -n512 gaborF40A${phase}_im
done


# CALCULATING THE FILTERED IMAGES
for text in afa afb afc afd afe afg
do # TEXT
```

```
# LOOP OVER DIFFERENT SLANT ANGLES
for slant in 45 60
do # SLANT

   # EMPTY FEATURE ARRAYS
   for freq in 20 30 40
   do # FREQ
      for phase in 0 45 90 135
      do # PHASE
         eval featurecomF${freq}A${phase}=\"\"
      done # PHASE
   done # FREQ

   # LOOP OVER DIFFERENT TILT ANGLES
   for tilt in 0 30 60 90 120 150 180
   do # TILT

      # CUT THE SIZE OF THE IMAGES
      $a/cutim -X512 -Y512 -x104 -y0 \
      $c/${text}/6.${text}.0.${slant}.${tilt} tmp

      # TRANSFORM THE IMAGE INTO FREQUENCY DOMAIN
      $a/fft3 tmp fft_re fft_im
      $a/swapshop fft_re fft_re fft_re
      $a/swapshop fft_im fft_im fft_im

      # FILTER IMAGE WHITH DIFFERENT COMPLEX GABOR FILTERS
      # AND COMPUTE FEATURES
      for freq in 20 30 40
      do # FREQ
         for phase in 0 45 90 135
         do # PHASE
            com_gabor fft_re fft_im $d/gaborF${freq}A${phase}_re \
                      $d/gaborF${freq}A${phase}_im final
            eval featurecomF${freq}A${phase}=\"\
                 \$featurecomF${freq}A${phase} \`variance_m final\`\"
         done # PHASE
      done # FREQ
   done # TILT

   # STORE RESULTS IN OUTPUT FILE
   for freq in 20 30 40
```

```
      do # FREQ
         for phase in 0 45 90 135
         do # PHASE
            eval echo \$text \$slant comF\${freq}A\${phase}\
                 \$featurecomF${freq}A${phase} \>\> ${name}
         done # PHASE
      done # FREQ
   done # SLANT
done # TEXT

echo \"`date`\" >> ${name}

# DELETE UNIMPORTANT FILES
rm fft_*
rm final
rm gaborF*
```

data_generation

This shellscript separates the input database into training and classification. For classification the odd tilt angles are used and to calculate the training parameters the even tilt angles are used.

```
set -x # debug version

# SYNOPSYS
# Generates for each surface in input a database train (odd
# tild angles) and a database class to test the classifier.
# The output of calculation is stored in the file features.
#
# AUTHOR
# Andreas Penirschke
#
# MODIFICATION
# 09/01/02
#
# MODULES REQUIRED
# Input to create the feature_vector
# [ comF20A0 comF20A45 comF20A90 comF20A135\
#   comF30S0 comF30A45 comF30A90 comf30A135 \
#   comF40A0 comF40A45 comF40A90 comF40A135 ]
```

```
# number of used Filters in the database filter_num

#SET CONSTANTS

#NUMBER OF USED FILTERS AND SLANT ANGLES IN DATABASE
filter_num=12
slant_num=4

#SET PATHS AND DEFINE INPUT AND OUTPUT FILE
a=/u1/ee3/ceeap/cpplinux
b=/u1/ee3/ceeap/experiments/dbase/dbase_real_10deg #DATABASE
input=dbase_synth_new # DATABASE

# BEST_FIT_APPROXXIMATION USING EVEN TILD ANGLE‘S
output_train=result_12/data_train
output_train_matrix=result_12/data_train_m.txt

# REDUCED DATABASE USING ODD TILD ANGLE’S
output_class=result_12/data_class
output_class_matrix=result_12/data_class_m.txt

#DELETING UNUSED FILTERS
row=‘cut -d’ ’ -f1 $b/$input | wc -w‘
eval sed -n \’1p\’ $b/$input \| tr \’\\11\’ \’ \’ \> dbase
i=2

while [ "$i" -le "$row" ]
do #i
   eval sed -n \’${i}p\’ $b/$input \| tr \’\\11\’ \’ \’ \>\> dbase
   i=‘expr $i + 1‘
   eval sed -n \’${i}p\’ $b/$input \| tr \’\\11\’ \’ \’ \>\> dbase
   i=‘expr $i + 3‘
   eval sed -n \’${i}p\’ $b/$input \| tr \’\\11\’ \’ \’ \>\> dbase
   i=‘expr $i + 2‘
   eval sed -n \’${i}p\’ $b/$input \| tr \’\\11\’ \’ \’ \>\> dbase
   i=‘expr $i + 6‘
done #i

# SEPARATE INPUT DATA INTO DATA_CLASS AND DATA_TEST
# STORE VARIABLES IN OUTPUT_CLASS AND TEMPORARY FILE TRAIN_DATA
column=‘sed -n ’1p’ dbase | wc -w‘
row=‘cut -d’ ’ -f1 dbase | wc -w‘
train_v=""
```

```
class_v=""
rm train_data
rm $b/$output_class
i=1
while [ "$i" -le "$row" ]
do
   eval sed -n \'${i}p\' dbase \> feature
   eval train_v=\"\$train_v\`cut -d\' \' -f1-3 feature \
                \| tr \' \' \'\\t\'\`\"
   eval class_v=\"\$class_v\`cut -d\' \' -f1-3 feature  \
                \| tr \' \' \'\\t\'\`\"

   j=4
   while [ "$j" -le "$column" ]
   do
      echo $i $j
      eval train_v=\"$train_v \`cut -d\' \' -f\$j feature\`\"
      j=`expr $j + 1`
      if [ "$j" -le "$column" ]
      then
         eval class_v=\"$class_v \`cut -d\' \' -f\$j feature\`\"
      fi
      j=`expr $j + 1`
   done
   i=`expr $i + 1`
   echo $train_v >> train_data
   echo $class_v >> $b/$output_class
   train_v=""
   class_v=""
done

#GENERATE MATRIX FILE WHICH CAN BE READ BY MATLAB
cat $b/$output_class | tr ' ' '\11' > tmp
cut -f{4-$column} > $b/$output_class_matrix

#CALCULATION OF THE BEST_FIT PARAMETERS OF TRAIN_DATA [A B C S]
#BEST_FIT_PARAMETERS ARE STORED IN FILE OUTPUT_BEST
column=`sed -n '1p' train_data | wc -w`
row=`cut -d' ' -f1 train_data | wc -w`
tilt_val=`expr $column - 3`
surf_row=`expr $filter_num \* $slant_num`
echo "surface filter a b c s" > $b/$output_train
i=2
```

```
while [ "$i" -le "$row" ]
do
   echo $i # LINECOUNTER

   j=$i
   tmp_filt=‘expr $i + $filter_num‘
   while [ "$j" -lt "$tmp_filt" ]
   do
      k=0
      train_v_slant=""
      train_val=""
      while [ "$k" -lt "$surf_row" ]
      do
         tmp1=‘expr $k + $j‘
         eval sed -n \’${tmp1}p\’ train_data > train_v
         eval train_v_slant=\"$train_v_slant
               \‘cut -d\’ \’ -f2 train_v\‘\"
         eval train_val=\"$train_val \‘cut -d\’ \’ -f4- train_v \
               \| tr \’ \’ \’\\t\’\‘\"
         k=‘expr $k + $filter_num‘
      done

      eval train_v_desc=\"\‘cut -d\’ \’ -f1,3 train_v \‘\"
      tmp="$train_val $train_v_slant $tilt_val"
      eval train_v_best=\"\‘$a/best_fit_dbase $tmp\‘\"
      echo "$train_v_desc $train_v_best" >> $b/$output_train
      j=‘expr $j + 1‘
   done
   i=‘expr $i + $surf_row‘
done

#GENERATE PARAMETER MATRIX FILE WHICH CAN BE READ BY
#MATLAB
sed -1d $b/output_train | tr ’ ’ ’\11’ > tmp
cut -f{3-$column} > $b/output_train_matrix

# DELETE UNIMPORTANT FILES
rm feature*
rm train_v*
rm dbase
```

class_matrix_slant

This shellscript classifies a given result matrix which is generated by tilt_slant_calc.m. by finding the maximum ML-value in every line of the matrix.

```
#set -x # debug version
# SYNOPSYS
#       DISCRIMINATION OF THE ML VALUES GIVEN
#       IN MATRIX GENERATED BY
#       TILT_SLANT_CALC.M
# AUTHOR
#       Andreas Penirschke
#
# MODIFICATION
#       20/01/02
#
# MODULES REQUIRED
#       MAX
#
# INPUT
#       RESULT.TXT
#
# OUTPUT
#       RESULT_CLASS.TXT

#SET PATHS
a=/u1/ee3/ceeap/cpp
b=/u1/ee3/ceeap/experiments/slant_angle/synth_surf/result_12

#SET INPUT AND OUTPUT FILES
input=result.txt
output=result_class.txt

#RESULT VECTOR
result_vector=""
```

```
#CALCULATE COLUMN AND ROWS OF INPUT FILE
column=‘sed -n ’1p’ $b/$input | wc -w‘
row=‘cut -f1 $b/$input | wc -l‘

#SEARCHING THE TEXTURE WITH MAXIMUM ML VALUE
#AND STORE IN OUTPUT FILE
i=1
while [ "$i" -le "$row" ]
do
   eval sed -n \’${i}p\’ $b/$input > image_res
   eval result_vector=\"\$result_vector\‘cut -f1 image_res\‘ \"
   data=""
   j=4
   while [ "$j" -le "$column" ]
   do
      echo "$i $j"
      eval data=\"\$data\‘cut -f\$j image_res\‘ \"
      j=‘expr $j + 3‘
   done
   best=‘$a/max $data‘
   start1=‘expr $best \* 3‘
   start=‘expr $start1 - 1‘
   end=‘expr $start + 2‘
   eval result_vector=\"\$result_vector \$best \‘cut \
                      -f\$start-\$end image_res\‘\\n\"
   i=‘expr $i + 1‘
done
echo $result_vector | tr ’ ’ ’\t’ > $b/$output

#DELETE UNIMPORTANT FILES
rm image_res
```

D.3 $Matlab^{TM}$-Scripts

Matlab-scripts in /u1/ee3/ceeap/matlab_tools

tilt_calc.m
MATLAB is a registered trademark. This script solves the discrimination function that is used for tilt estimation with known slant. The mathematical background can be found in C.1 on page 87. The input is the training database (vectors $\vec{a}$, $\vec{b}$, $\vec{c}$ and $\vec{s}$ for each of the n surfaces calculated by the shell script D.2 on page 107) and the feature matrix which contains m unknown surfaces (m unknown feature vectors $\vec{y}$). The output is a matrix $m \times n$, which contains the results of the Maximum Likelihood estimator.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% PROGRAM NAME:       tilt_calc_polinom.m                               %
%                                                                       %
% DESCRIPTION:        discrimination algorithm to                       %
%                     classify textures with unknown                    %
%                     illumination tilt direction                       %
%                                                                       %
% INPUT               data_train.txt                                    %
%                     data_class.txt                                    %
%                                                                       %
% OUTPUT              result.txt                                        %
%                                                                       %
% Author:             Andreas Penirschke                                %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%DEFINITION OF GLOBAL VARIABLES

global y A B C D E F s

%DEFINITION OF LOCAL VARIABLES

filter_num=12; %NUMBER OF USED FILTERS
train_surf=25; %NUMBER OF TRAINING TEXTURES
class_surf=50; %NUMBER OF TEXTURES TO CLASSIFY

%OPEN TEXT FILES

[train_dat,message]=fopen('tilt_angle/data_train.txt','rt');
```

```
if  train_dat == -1
    disp(message)
end

[class_dat,message]=fopen('tilt_angle/data_class.txt','rt');

if  class_dat == -1
    disp(message)
end

result=fopen('tilt_angle/result','wt');

%READ THE SIZE OF THE FILE DATA_CLASS

[tmp,size]=fscanf(class_dat,'%s');
status=fclose(class_dat);

%REOPEN DATA_CLASS TO READ FEATURE VECTORS OF UNKNOWN
%IMAGES

[class_dat,message]=fopen('tilt_angle/data_class.txt','rt');

if  class_dat == -1
    disp(message)
end

%CALCULATE THE NUMBER OF IMAGES ILLUMINATED WITH
%DIFFERENT TILT ANGLES OF EVERY TEXTURE IN DAT_CLASS
class_column=size/(class_surf*filter_num+1);
class_angle=fscanf(class_dat,'%f',class_column);

train=fscanf(train_dat,'%f');

%LOOP OVER DIFFERENT TEXTURES IN DATA_CLASS
for i = 1:1:class_surf
    surf=[(fscanf(class_dat,'%f',class_column))'];

    %READ FEATURE VALUES FOR UNKNOWN TEXTURE IN DATA_CLASS
    %(FEATURE VALUES FOR ALL IMAGES FROM A TEXTURE)
    for j = 2:1:filter_num
        surf=[surf;(fscanf(class_dat,'%f',class_column))'];
    end
```

```
%LOOP OVER NUMBER OF DIFFERENT IMAGES OR EVERY TEXTURE
for j = 1:1:class_column
    angle=class_angle(j);

    %PRINT ILLUMINATION TILT ANGLE OF THE IMAGE
    fprintf(result,'%d\t',angle);
    y=(surf(:,j))';

    %LOOP OVER TEXTURES IN DATA_CLASS
    for k = 0:(4*filter_num):(4*filter_num)*(train_surf-1)
        a=[]; b=[]; c=[]; s=[];

        %READ BEST FIT PARAMETERS FOR ONE TEXTURE IN DATA_TRAIN
        for m = 1:4:(4*(filter_num-1)+1)
            a=[a train(k+m)];
            b=[b train(k+m+1)];
            c=[c train(k+m+2)];
            s=[s train(k+m+3)];
        end

        %CALCULATE SUBSTITUDES

        %GENERAL

        A=((y-a)./(2.*s)*((y-a)./s)');
        B=((a-y)./s)*(b./s)';
        C=((a-y)./s)*(c./s)';
        D=(b./(2.*s)*(b./s)');
        E=((b./s)*(c./s)');
        F=((c./(2.*s))*(c./s)');
        AA=(2*b./s)*((y-a)./s)';
        BB=(2*c./s)*((a-y)./s)';
        CC=(2*c./s)*(c./s)'-(2*b./s)*(b./s)';
        DD=-(4*b./s)*(c./s)';
        EE=(2*b./s)*(c./s)';

        %CALCULATE THE ROOTS OF THE POLYNOMIAL Q
        Q=[ (CC^2+DD^2) , (2*AA*DD+2*BB*CC) ,
            (AA^2+2*DD*EE+BB^2-CC^2) ,
            (2*AA*EE-2*BB*CC) , (EE^2-BB^2) ];
        res=roots(Q);
```

```
angle=0;
probt=0;
prob=0.0;
tilt=0.0;

%CALCULATE THE ILLUMINATION TILT ANGLE
%OF THE SURFACE
for n=1:1:length(res)

    if  -0.0001 <= res(n) & res(n) <= 1.001

        %VALUE OF TILT IN [0..PI/2]

        angle=asin(real(res(n)))/2;
        probt=1/(sqrt(2*pi)^length(s)*prod(s))*
              exp(-(A+B*cos(2*angle)+C*sin(2*angle)+
              D*(cos(2*angle))^2+E*cos(2*angle)*
              sin(2*angle)+F*(sin(2*angle))^2));

        if  probt > prob
            prob=probt;
            tilt=abs(angle*180/pi);
        end

        %VALUE OF TILT IN [PI/2..PI]

        angle=(pi-asin(real(res(n))))/2;
        probt=1/(sqrt(2*pi)^length(s)*prod(s))*
              exp(-(A+B*cos(2*angle)+
              C*sin(2*angle)+D*(cos(2*angle))^2+
              E*cos(2*angle)*sin(2*angle)+F*
              (sin(2*angle))^2));

        if  probt > prob
            prob=probt;
            tilt=abs(angle*180/pi);
        end

    end

    if  -1.001 <= res(n) & res(n) <= +0.0001
```

```
                    %VALUE OF TILT IN [PI..3/2*PI]

                    angle=(pi-asin(real(res(n))))/2;
                    probt=1/(sqrt(2*pi)^length(s)*prod(s))*
                          exp(-(A+B*cos(2*angle)+C*sin(2*angle)+
                          D*(cos(2*angle))^2+E*cos(2*angle)*
                          sin(2*angle)+F*(sin(2*angle))^2));

                    if  probt > prob
                        prob=probt;
                        tilt=abs(angle*180/pi);
                    end

                    %VALUE OF TILT IN [3/2*PI..2*PI]

                    angle=(2*pi+asin(real(res(n))))/2;
                    probt=1/(sqrt(2*pi)^length(s)*prod(s))*
                          exp(-(A+B*cos(2*angle)+C*sin(2*angle)+
                          D*(cos(2*angle))^2+E*cos(2*angle)*
                          sin(2*angle)+F*(sin(2*angle))^2));

                    if  probt > prob
                        prob=probt;
                        tilt=abs(angle*180/pi);
                    end

                end

            end

            %WRITE TILT ANGLE AND ML_VALUE IN RESULT
            fprintf(result,'%f\t%e\t',tilt,prob);
        end

        %WRITE CARRIAGE RETURN IN RESULT
        fprintf(result,'\n');
    end

    %WRITE CARRIAGE RETURN IN RESULT
    fprintf(result,'\n');
end

%CLOSE ALL OPEN FILES
```

```
status=fclose(train_dat);
status=fclose(class_dat);
status=fclose(result);
```

tilt_slant_calc.m

This script solves the discrimination function of the tilt_slant classifier. The mathematical background can be found in C.2 on page 103. The input and the output is the same as for *tilt_calc* (D.3 on page 115).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% PROGRAM NAME:      tilt_slant_calc_polinom.m                          %
%                                                                       %
% DESCRIPTION:       discrimination algorithm to                        %
%                    classify textures with                             %
%                    unknown illumination direction                     %
%                                                                       %
% INPUT              data_train.txt                                     %
%                    data_class.txt                                     %
%                                                                       %
% OUTPUT             result.txt                                         %
%                                                                       %
% Author:            Andreas Penirschke                                 %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%DEFINITION OF GLOBAL VARIABLES

global y A B C D E F

%DEFINITION OF LOCAL VARIABLES

filter_num=12; %NUMBER OF USED FILTERS
train_surf=25; %NUMBER OF TRAINING TEXTURES IN DATA_TRAIN
class_surf=50; %NUMBER OF TEXTURES IN DATA_CLASS

%OPEN TEXT FILES
%TRAINING SEQUENCE

[train_dat,message]=fopen('slant_angle/data_train.txt','rt');

if  train_dat == -1
    disp(message)
end

%TEST SEQUENCE
```

```
[class_dat,message]=fopen('slant_angle/data_class.txt','rt');

if  class_dat == -1
    disp(message)
end

result=fopen('slant_angle/result.txt','wt');

%READ THE SIZE OF THE FILE DATA_CLASS

[tmp,size]=fscanf(class_dat,'%s');
status=fclose(class_dat);

%REOPEN DATA_CLASS TO READ FEATURE VECTORS OF UNKNOWN
%ILLUMINATION DIRECTION

[class_dat,message]=fopen('slant_angle/data_class.txt','rt');

if  class_dat == -1
    disp(message)
end

%CALCULATE THE NUMBER OF IMAGES ILLUMINATED WITH
%DIFFERENT TILT ANGLES OF EVERY TEXTURE IN DAT_CLASS
class_column=(size)/(class_surf*filter_num+1);
class_angle=fscanf(class_dat,'%f',class_column);
train=fscanf(train_dat,'%f');

%LOOP OVER DIFFERENT TEXTURES IN DATA_CLASS
for i = 1:1:class_surf
    surf=[(fscanf(class_dat,'%f',class_column))'];

    %READ FEATURE VALUES FOR UNKNOWN TEXTURE IN DATA_CLASS
    %(FEATURE VALUES FOR ALL IMAGES FROM A TEXTURE)
    for j = 2:1:filter_num
        surf=[surf;(fscanf(class_dat,'%f',class_column))'];
    end

    %LOOP OVER NUMBER OF DIFFERENT IMAGES OR EVERY TEXTURE
    for j = 1:1:class_column
        angle=class_angle(j);

        %PRINT ILLUMINATION TILT ANGLE OF THE IMAGE
```

```
    fprintf(result,'%d\t',angle);
    y=(surf(:,j))';

    %LOOP OVER TEXTURES IN DATA_CLASS
for k = 0:(4*filter_num):(4*filter_num)*(train_surf-1)
        a=[]; b=[]; c=[]; s=[];

        %READ BEST FIT PARAMETERS FOR ONE TEXTURE IN DATA_TRAIN
        for m = 1:4:(4*(filter_num-1)+1)
            a=[a train(k+m)];
            b=[b train(k+m+1)];
            c=[c train(k+m+2)];
            s=[s train(k+m+3)];
        end

    %CALCULATE SUBSTITUDES

    %GENERAL

    MAA=a./s*(a./s)';
    MAB=a./s*(b./s)';
    MAC=a./s*(c./s)';
    MAY=a./s*(y./s)';
    MBB=b./s*(b./s)';
    MBC=b./s*(c./s)';
    MBY=b./s*(y./s)';
    MCC=c./s*(c./s)';
    MCY=c./s*(y./s)';
    MYY=y./s*(y./s)';

    %SUBSTITUTIONS USED FOR THE FIRST DERIVATIVES

    VA=16*(MBC)^2+4*(MBB-MCC)^2;
    VB=16*MAB*MBC+8*MAC*MCC-8*MAC*MBB;
    VC=8*MCY*MBB-16*MBY*MBC-8*MCY*MCC;
    VD=4*(MAB)^2-16*(MBC)^2+4*(MAC)^2-4*(MBB-MCC)^2;
    VE=-8*MBY*MAB-8*MAC*MCY;
    VF=4*(MBY)^2+4*(MCY)^2;
    VG=8*MAC*MBB-8*MAB*MBC-8*MAC*MCC;
    VH=8*MBY*MBC+8*MCY*MCC-8*MCY*MBB;
    VI=4*(MBC)^2-4*(MAC)^2;
    VK=8*MAC*MCY;
    VL=-4*(MCY)^2;
```

```
WA=(MCY)^2-(MBY)^2;
WB=2*MAY*MCY;
WC=(MAY)^2+(MBY)^2;
WD=MAY*MCC-MAY*MBB-2*MBY*MAB+2*MAC*MCY;
WE=MCC*MCY-MBB*MCY-2*MBY*MBC;
WF=2*MAY*MAC+2*MBY*MBC+MAA*MCY+MBB*MCY;
WG=MAA*MAY+MAY*MBB+2*MBY*MAB;
WH=(MCC-MBB)^2-4*(MBC)^2;
WI=4*MAC*(MCC-MBB)-8*MAB*MBC;
WK=2*(MAA+MBB)*(MCC-MBB)+4*(MAC)^2+4*(MBC)^2-4*(MAB)^2;
WL=4*MAC*(MAA+MBB)+8*MAB*MBC;
WM=(MAA+MBB)^2+4*(MAB)^2;
WN=MBY*MCC-MBY*MBB+2*MCY*MBC;
WO=2*MAY*MBC+2*MBY*MAC+2*MCY*MAB;
WP=2*MAY*MAB+MBY*MAA+MBY*MBB;
WR=4*MBC*(MCC-MBB);
WS=8*MAC*MBC+4*MAB*(MCC-MBB);
WT=4*MBC*(MAA+MBB)+8*MAC*MAB;
WU=4*MAB*(MAA+MBB);
WV=2*MAY*MBY;
WW=2*MBY*MCY;

DA=WW*VA+WN*VC+WR*VF;
DB=WV*VA+WW*VB+WN*VE+WO*VC+WS*VF;
DC=WV*VB+WW*VD+WN*VH+WO*VE+WP*VC+WT*VF+WR*VL;
DD=WV*VD+WW*VG+WN*VK+WO*VH+WP*VE+WU*VF+WS*VL;
DE=WV*VG+WW*VI+WO*VK+WP*VH+WT*VL;
DF=WV*VI+WP*VK+WU*VL;

EA=(DA)^2;
EB=2*DA*DB;
EC=2*DA*DC+(DB)^2;
ED=2*DA*DD+2*DB*DC;
EE=2*DA*DE+2*DB*DD+(DC)^2;
EF=2*DA*DF+2*DB*DE+2*DC*DD;
EG=2*DB*DF+2*DC*DE+(DD)^2;
EH=2*DC*DF+2*DD*DE;
EI=2*DD*DF+(DE)^2;
EK=2*DE*DF;
EL=(DF)^2;

GA=WA*VA+WE*VC+WH*VF;
```

```
GB=WA*VB+WB*VA+WE*VE+WD*VC+WI*VF;
GC=WA*VD+WB*VB+WC*VA+WE*VH+WD*VE+WF*VC+WK*VF+WH*VL;
GD=WA*VG+WB*VD+WC*VB+WE*VK+WD*VH+WF*VE+WG*VC+WL*VF+WI*VL;
GE=WA*VI+WB*VG+WC*VD+WD*VK+WF*VH+WG*VE+WM*VF+WK*VL;
GF=WB*VI+WC*VG+WF*VK+WG*VH+WL*VL;
GG=WC*VI+WG*VK+WM*VL;

HA=(GA)^2;
HB=2*GA*GB;
HC=2*GA*GC+(GB)^2;
HD=2*GA*GD+2*GB*GC;
HE=2*GA*GE+2*GB*GD+(GC)^2;
HF=2*GA*GF+2*GB*GE+2*GC*GD;
HG=2*GA*GG+2*GB*GF+2*GC*GE+(GD)^2;
HH=2*GB*GG+2*GC*GF+2*GD*GE;
HI=2*GC*GG+2*GD*GF+(GE)^2;
HK=2*GD*GG+2*GE*GF;
HL=2*GE*GG+(GF)^2;
HM=2*GF*GG;
HN=(GG)^2;

%CALCULATE THE ZERO POINTS OF THE POLYNOMIAL
%(ONLY DEPENDENT ON TILT)

KA=(HA+EA);
KB=(HB+EB);
KC=(HC+EC-EA);
KD=(HD+ED-EB);
KE=(HE+EE-EC);
KF=(HF+EF-ED);
KG=(HG+EG-EE);
KH=(HH+EH-EF);
KI=(HI+EI-EG);
KK=(HK+EK-EH);
KL=(HL+EL-EI);
KM=(HM-EK);
KN=(HN-EL);

Q=[ KA , KB , KC , KD , KE , KF , KG , KH , KI , KK  ,
    KL , KM , KN ];
tilt_vec=roots(Q);

%DELETE COMPLEX SOLUTIONS FOR TILT
```

```
tilt_real=[];

for n=1:1:length(tilt_vec)

    if  abs(imag(tilt_vec(n))) < 0.001
        tilt_real=[tilt_real real(tilt_vec(n))];
    end

end

%CALCULATE POSSIBLE SOLUTIONS FOR ILLUMINATION SLANT
slant_vec1=(MAY+sqrt(1-tilt_real.^2).*MBY+tilt_real.*MCY)./
           ((MAA+MBB)+2*sqrt(1-tilt_real.^2).*(MAB+MBC.*tilt_real)+
           2*MAC.*tilt_real+(MCC-MBB).*tilt_real.^2);

slant_vec2=(MAY-sqrt(1-tilt_real.^2).*MBY+tilt_real.*MCY)./
           ((MAA+MBB)-2*sqrt(1-tilt_real.^2).*(MAB+MBC.*tilt_real)+
           2*MAC.*tilt_real+(MCC-MBB).*tilt_real.^2);

tilt=[];
slant=[];

%CALCULATE POSSIBLE SOLUTIONS FOR ILLUMINATION TILT AND THE
%CORRESPONDING ILLUMINATION SLANT VECTORS (2 VALUES)
for n=1:1:length(slant_vec1)

    if  (slant_vec1(n) >= 0 & slant_vec1(n) <= 1 &
        slant_vec2(n) >= 0 & slant_vec2(n) <= 1)

        if  -0.0001 <= tilt_real(n) & tilt_real(n) <= 1.001

            %VALUE OF TILT IN [0..PI/2]

            tilt=[tilt asin(tilt_real(n))/2
                 asin(tilt_real(n))/2];
            slant=[slant asin(sqrt(slant_vec1(n)))
                  asin(sqrt(slant_vec2(n)))];

            %VALUE OF TILT IN [PI/2..PI]

            tilt=[tilt (pi-asin(tilt_real(n)))/2
                 (pi-asin(tilt_real(n)))/2];
```

```
                slant=[slant asin(sqrt(slant_vec1(n)))
                      asin(sqrt(slant_vec2(n)))];
            end

            if  -1.001 <= tilt_real(n) & tilt_real(n) <= +0.0001

                %VALUE OF TILT IN [PI..3/2*PI]

                tilt=[tilt (pi-asin(tilt_real(n)))/2
                     (pi-asin(tilt_real(n)))/2];
                slant=[slant asin(sqrt(slant_vec1(n)))
                      asin(sqrt(slant_vec2(n)))];

                %VALUE OF TILT IN [3/2*PI..2*PI]

                tilt=[tilt (2*pi+asin(tilt_real(n)))/2
                     (2*pi+asin(tilt_real(n)))/2];
                slant=[slant asin(sqrt(slant_vec1(n)))
                      asin(sqrt(slant_vec2(n)))];
            end

        end

    end

    %PRESETTINGS IF NO SOLUTION FOUND
    prob=-1e100;
    tilt_final=-1;
    slant_final=-1;

    %CALCULATE THE ML_VECTOR (2 VALUES DEP. ON SLANT)
    %FOR EVERY POSSIBLE TILT SOLUTION.
    for n=1:1:length(tilt)
        probt=1/(sqrt(2*pi)^length(s)*prod(s))*
              exp(-(MYY/2-(sin(slant(n)))^2*
              (MAY+cos(2*tilt(n))*MBY+sin(2*tilt(MCY)+
              (sin(slant(n)))^4*(MAA/2+cos(2*tilt(n))*
              MAB+sin(2*tilt(n))*MAC+(cos(2*tilt(n)))^2*
              MBB/2+cos(2*tilt(n))*sin(2*tilt(n))*
              MBC+(sin(2*tilt(n)))^2*MCC/2)));

        %HOLD THE MAXIMUM SOLUTION FOR ML_VALUE IN PROB
        %AND TILT AND SLANT IN TILT_ AND SLANT_FINAL
```

```
            if  probt > prob
                prob=probt;
                tilt_final=(tilt(n)*180/pi);
                slant_final=(slant(n)*180/pi);
            end

        end

        %WRITE SOLUTION TO RESULT.TXT
        fprintf(result,'%f\t%f\t%e\t',tilt_final,slant_final,prob);
    end

    %WRITE CARRIAGE RETURN TO RESULT.TXT
    fprintf(result,'\n');
end

%WRITE CARRIAGE RETURN TO RESULT.TXT
fprintf(result,'\n');
end

%CLOSE ALL FILES
status=fclose(train_dat);
status=fclose(class_dat);
status=fclose(result);
```

Bibliography

[1] **Ronald N. Bracewell**, *The Fourier Transform and its Applications 3rd ed.*, McGraw-Hill International Editions, 2000.

[2] **I. N. Bronstein, K. A. Semendjaev**, *Taschenbuch der Mathematik*, Verlag Harry Deutsch Frankfurt am Main 1999, pp 394-395

[3] **Michael J. Chantler**, *The effect of variation in illuminant direction on texture classification*, PhD Theses, Department of Computing and Electrical Engineering, Heriot Watt University, August 1994

[4] **M. J. Chantler, G. McGunnigle**, *The response of texture features to illuminant rotation*, 15th International Converence on Pattern Recognition, Vol. 3 pp. 943 - 946, September 2000

[5] **M. J. Chantler, M. Schmidt, M. Petrou and G. McGunnigle**, *The Effect of Illuminant Rotation on Texture Filters: Lissajous's Ellipses*

[6] **J. G. Daugman**, *Uncertainty relation for resolution in space, spatial-frequency, and orientation optimized by two dimensional visual cortical filters*, J. Opt. Soc Am A Vol. 2 No.7 July 1985, pp1160-1169.

[7] **G. McGunnigle**, *The Classification of Textured Surfaces Under Varying Illuminant Directions*, PhD Theses, Department of Computing and Electrical Engineering, Heriot Watt University, April 1998

[8] **Eberhard Hänsler**, *Statistische Signale, Grundlagen und ihre Anwendungen, 2. Auflage*, SpringerVerlag Berlin Heidelberg New York 1997

[9] **A. K. Jain F. Farrokhina**, *"Unsupervised texture segmentation using Gabor Filters*, IEEE Trans. on Pattern Analysis and Machine Intelligence Vol.10 No.5 pp.317-324, September 1998

[10] **P. Kube & A. Pendland**, *"On the Imaging of Fractal Surfaces*, IEEE Trans. on Pattern Analysis and Machine Intelligence Vol.10 No.5 pp.317-324, September 1998

[11] **K. I. Laws**, *Texture Energy Measures*, Proceedings in Image Understanding Workshop, No. 1979 pp. 47-51.

[12] **Trygve Randen and John H. Husoy**, *Filtering for Texture Classification: A Comparative Study*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 4, April 1999

[13] **Todd R. Reed and J. M. Hans du Buf**, *A Review of Recent Texture Segmentation and Feature Extraction Techniques*, CVGIP: Image Understanding, V57, No.3, pp359–372, May 1993

[14] **Michael Schmidt**, *The Effect of Changing Illumination Tilt on Texture Features*, Study Reasearch Project, Department of Computing and Electrical Engineering, Heriot Watt University, October 2001

[15] **Christian V. Sinn**, *An investigation into the importance of phase in textured images*, Diploma Theses, Department of Computing and Electrical Engineering, Heriot Watt University, July 2000