

EXAMENSARBETE VID NADA, KTH

**Identifying Defective IR Cameras through a
Machine Learning Approach to Image Artifact
Detection**

**Identifiering av defekta IR-kameror med ett
lärande system för detektion av bildartefakter**

Oscar Danielsson

e-postadress vid KTH: osda02@kth.se

Exjobb i: Datalogi

Handledare: Örjan Ekeberg

Examinator: Anders Lansner

Uppdragsgivare: FLIR Systems AB

Abstract

Title: Identifying Defective IR Cameras through a Machine Learning Approach to Image Artifact Detection

This report is about methods for automatic detection of subtle image artifacts. Such methods could be a part of the image quality control procedure in the production process of infrared cameras. Currently image quality control requires manual inspection.

The main goal of the investigation described in this report was to suggest and evaluate different measures that can be used to detect image artifacts and assess image quality. In many cases these measures are attempts to find structure in what should be random white noise and will assume that test images are acquired against a uniform radiation. Often single-valued measures are sufficient for achieving good detection accuracy, but in some cases more complex, vector valued measures (termed features) in combination with a machine learning classifier are needed.

This report conclusively shows that automatic detection of subtle image artifacts with high accuracy is possible and in many cases achievable using single-valued measures.

Sammanfattning

Titel: Identifiering av defekta IR-kameror med ett lärande system för detektion av bildartefakter

Denna rapport handlar om metoder för automatisk detektion av diffusa bildartefakter. Sådana metoder kan bli en del av den bildkvalitetskontroll som utförs vid produktion av värmekameror. I nuläget krävs manuell granskning av varje kamera.

Den utredning som beskrivs har fokuserats på att föreslå och utvärdera olika mått som kan användas för att detektera artefakter i bilder och för att mäta bildkvalité. I många fall är dessa mått försök att finna struktur i bilder som borde bestå av enbart vitt brus och kräver att testbilden tas mot en likformig strållare. Ofta räcker envärda mått för att detektera artefakter med god precision, men i vissa fall krävs mer komplexa vektorvärda mått (som kommer att kallas features) i kombination med en lärande klassificerare.

Denna rapport visar slutgiltigt att automatisk detektion av artefakter med hög precision är möjlig och ofta kan uppnås med envärda mått.

Preface

This thesis project was suggested and sponsored by FLIR Systems AB in Danderyd, Sweden. I would like to thank my supervisors at FLIR, Malin Ingerhed and Lars-Åke Tunell, for the great support and encouragement that I have received during my work on this thesis. I also thank my other colleagues at FLIR for their good spirit and enthusiasm and for their interest in this project.

The academic supervision of the thesis was provided by Örjan Ekeberg and the Computational Biology and Neurocomputing (CBN) group at the School of Computer Science and Communication, KTH. I thank him for his support.

Oscar Danielsson, December 2006, Danderyd, Sweden

Contents

1 Introduction	1
1.1 Background	1
1.1.1 Thermographic Imaging	2
1.1.2 The Infrared Camera	2
1.1.3 Uses and Market for Infrared Cameras	2
1.1.4 Quality Control	2
1.2 Problem Description	3
1.2.1 Observations	4
1.3 Report Outline	5
2 Computer Vision Background	6
2.1 Image Transforms	6
2.2 Filters	7
2.3 Image Statistics	8
2.4 Interest Point Detection and Saliency	8
2.5 Texture Classification	9
2.6 Content-based Image Retrieval	10
2.7 Template Matching	11
2.8 Face Detection	11
3 Machine Learning Background	12
3.1 Nearest Neighbor	12
3.2 Multi-layered Perceptron	12
3.3 Support Vector Machine	13
3.4 Bayesian Classifiers	14
3.5 Parameter Selection and Dimensionality Reduction	14
4 General Architecture	16
4.1 Training the Classifiers	17
4.2 Detecting the Presence of Artifacts	18
4.3 Avoiding False Positives	19
5 Features	20
5.1 Spot Detection	20
5.1.1 Radon Profile	20
5.1.2 Edge Pixel Location	21
5.1.3 Spin Image	23
5.1.4 Gradient Direction Histogram	23
5.1.5 Downsampled Original	25
5.1.6 Distance from Center and Intensity Correlation	25

5.2 Detection of Periodic Artifacts.....	25
5.2.1 Fourier Coefficients	26
5.2.2 Gradient Direction Histogram.....	30
5.3 Detection of “Shadows”	30
5.3.1 Gray-Level Co-Occurrence Matrix (GLCM).....	31
5.3.2 Point-Point Distance Distribution	32
5.3.3 Graininess.....	33
5.4 Detection of Row Noise	34
5.4.1 Variance of Row Intensity Means.....	35
5.4.2 Row Intensity Variance Minimum.....	36
5.4.3 Number of Variance and Mean Outliers	36
5.4.4 Model Conformity.....	37
5.4.5 Maximum χ^2 Histogram Distance	38
5.4.6 Number of χ^2 Distance Outliers	39
5.4.7 Number of Small Entropies.....	39
6 Results.....	41
6.1 Spot Detection.....	41
6.2 Detection of Periodic Artifacts.....	48
6.3 Detection of “Shadows”	49
6.4 Detection of Row noise.....	51
7 Discussion	53
7.1 Results Analysis	53
7.2 Implementation	54
7.2.1 Choice of Features	54
7.2.2 Test Image Acquisition	55
7.2.3 Selection of Training Examples	56
7.2.4 Obtaining a Confidence Measure.....	56
7.3 Future Improvements	57
7.3.1 Artifact Detection on the Camera	57
7.3.2 Global Detection of Local Artifacts	59
7.3.3 Online Learning	59
7.3.4 Letting the System Request Images	60
7.3.5 Detecting Other Artifacts	60
7.3.6 Validation and Tuning.....	63
7.4 Alternative Set of Features.....	64
8 Summary and Conclusions.....	68
8.1 Summary	68

8.2 Conclusions.....	69
Bibliography.....	70
Appendix A – Detailed Results.....	72
Spot Detection.....	72
Detection of Periodic Artifacts.....	74
Detection of “Shadows”	75
Detection of Row noise.....	79

1 Introduction

This chapter is a general introduction, including some background information on infrared (IR) imaging and a detailed problem description.

The purpose of this Master's thesis project is to investigate the possibility of training a machine learning system to automatically detect subtle artifacts in IR images. Such system could be used for automatic image quality control in IR camera production. The subtle artifacts that are to be detected signify camera defects. The final goal (which will not be achieved in this thesis) is illustrated in Figure 1.1. The camera is first placed in some test setup, where it generates a series of test images. The test images are then analyzed by a machine learning system, which has been trained to recognize the characteristic patterns that indicate the presence of artifacts in an image. If artifacts are detected in any of the test images, the camera will be classified as defect. Otherwise the camera will be shipped to the customer.

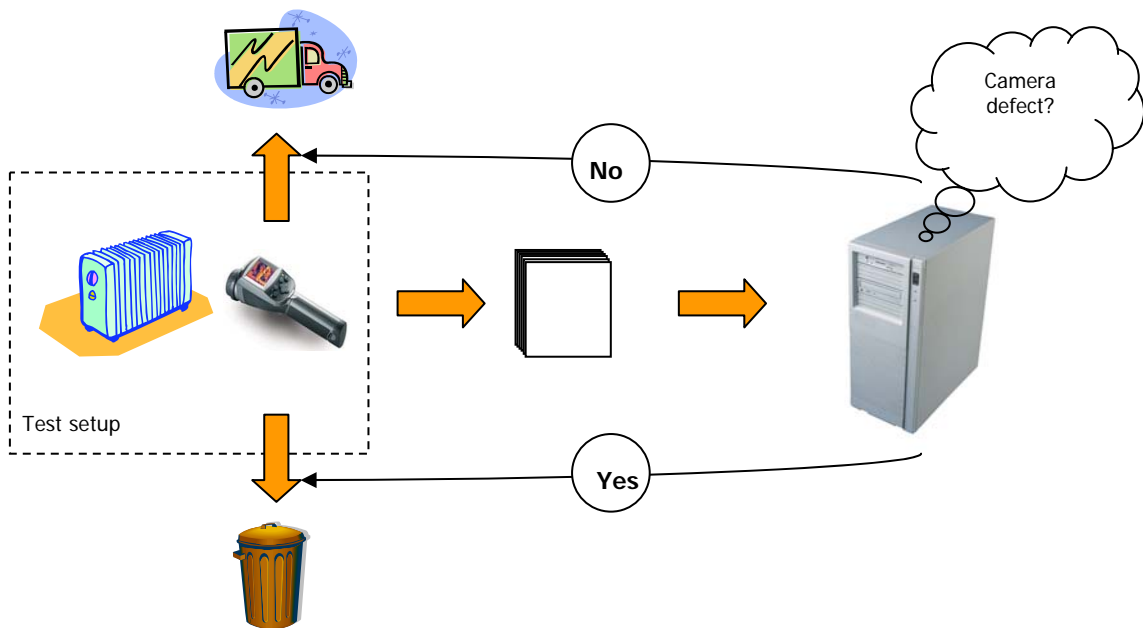


Figure 1.1: Sketch of the test situation

The machine learning system cannot use the raw images to detect artifacts; the images need pre-processing. It is not clear what pre-processing should be applied to the images in order to give the machine learning system optimal conditions for distinguishing artifacts. Therefore this thesis will be dedicated mainly to proposing and evaluating different pre-processing methods. Pre-processing will later be called feature extraction and the result of this process will be called features. So this thesis is about defining good features that allow a machine learning system to detect artifacts in images. As a consequence of evaluating the performance of different features, many insights about the behavior of the chosen machine learning method will also result.

Before continuing with a more comprehensive problem description, a short background on thermographic imaging will be given.

1.1 Background

This section contains a short background on thermographic imaging and IR cameras.

1.1.1 Thermographic Imaging

Normal cameras capture the light reflected of objects (or emitted from light sources) in the scene. In contrast, an IR camera captures the heat radiation emitted from (and reflected of) objects in the scene. Thus the normal camera and the IR camera “see” in different regions of the electromagnetic spectrum. By capturing the heat radiation emitted from an object, the IR camera can measure the temperature of the object. It is very useful in many applications to be able to deduce the temperature of an object without having to touch the object with a thermometer (applications will be discussed further in section 1.1.3). Figure 1.2 shows an example of an IR image depicting a pipe. The color bar to the right of the image gives a visual temperature index. More information on thermographic imaging can be found in [41].

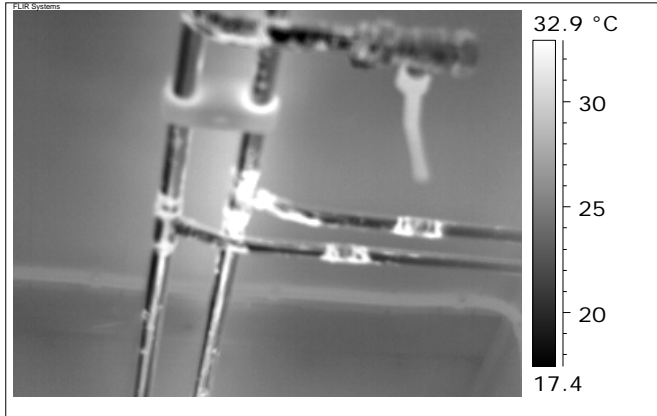


Figure 1.2: An example IR image.

1.1.2 The Infrared Camera

How does the IR camera work? Architecturally it is similar to an ordinary video camera. It has a lens that precedes a detector, which is used to collect infrared radiation. The detector sources a continuous stream of images which can be viewed on a display and stored.

The lenses are commonly made of germanium since glass is impermeable to infrared radiation. Imperfections or dust on the lens might be the cause of image artifacts.

The core of the camera is the detector. The most common type of detector is called microbolometer. One advantage of the microbolometer compared to previous techniques is that the microbolometer does not require cooling, which was necessary in older systems. The detector is a grid (commonly 320 by 240) of temperature sensors that convert the incoming radiation in each pixel to an electrical response. Non-linearity in the response function might be one cause of image artifacts. The detector also has a transparent casing, which might be damaged or scratched, causing other artifacts.

1.1.3 Uses and Market for Infrared Cameras

IR cameras are used in a broad range of applications. One example is condition based maintenance (CM), where IR cameras are used to optimize maintenance (and avoid incidents) by monitoring for example electrical connections, buildings, furnaces and boilers [41]. IR cameras also have applications in quality control and process monitoring, medical and veterinary applications, research and development and non-destructive testing.

1.1.4 Quality Control

Due to the demands for high quality, each camera is subjected to a combination of automatic and manual screening as a part of the production process. The automatic testing routines can detect a number of defects, but manual inspection is needed to find more subtle image artifacts.

As the demand for low end and low price cameras becomes greater and the volume of produced cameras increases, it is desirable to detect as many defects as possible automatically to reduce production costs. That is the motivation for the problem treated in this thesis, which is described in the following section.

1.2 Problem Description

The goal is to automatically detect subtle image artifacts that currently require manual inspection. Often the artifacts targeted in this thesis only appear when an image is captured under extreme conditions, for example when imaging very hot objects using a minimal span setting on the camera. Even then the artifacts will usually influence the pixel intensity values less than the noise in the rest of the image, but might still be disturbing to the user because humans are very effective at noticing regularities. One common class of artifacts is named “rings” and is illustrated as a representative example in Figure 1.3. Another example is “shadows”, which are signified by a shadow-like pattern when looking at a uniform temperature with minimum span. A very clear example of “shadows” is shown in Figure 1.4.

The rings are an example of a spatial artifact, but there are also temporal artifacts. They are typically only noticeable in image sequences and are hardly seen in a single frame. This thesis is also targeting temporal artifacts.

The task is to devise methods that can detect the presence of these artifacts. Initially the focus will lie on a few common artifacts to gain experience and develop strategies for artifact detection, but hopefully these strategies can be extended to detect new types of artifacts. An important part of such strategy will be machine learning methods to learn to recognize an artifact after training on example images containing that artifact. The other main component will be methods from computer vision and signal processing to extract useful information from the images, since it is obvious that the classification of artifacts cannot be done in image space. Both training and the later recognition of image artifacts will thus be a two step process, where the first step is to extract salient, low dimensional and discriminative information from the images and the second step is to use that information to train a classifier (the learning phase) or as input to an already trained classifier (the classification phase). Thus solving the problem of artifact detection will require methods from machine learning and computer vision.

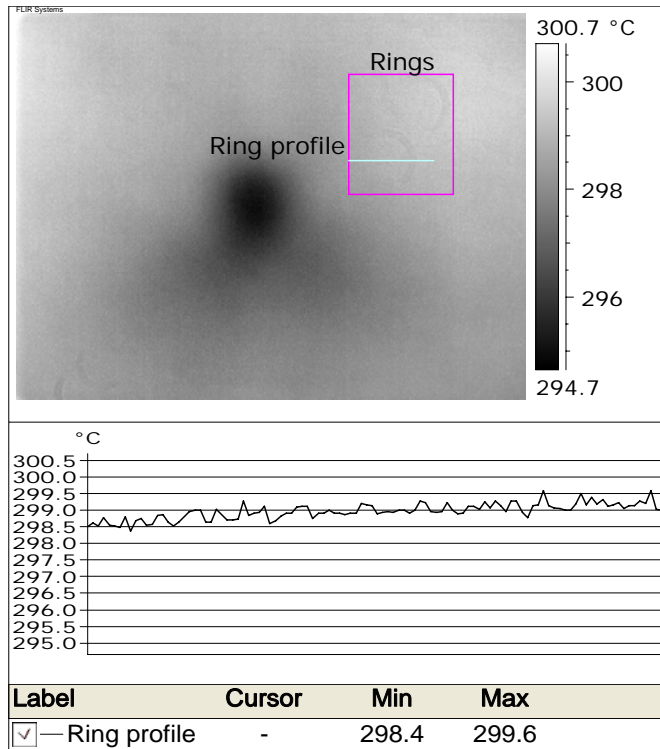


Figure 1.3: Image showing the appearance of "rings", which constitute one class of subtle artifacts. The (purple) square marks the spatial extent of the rings and the (turquoise) line marks the path of the temperature profile shown in the chart. The subtle ring intersected by the profile path is concealed by the noise and not visible in the chart.

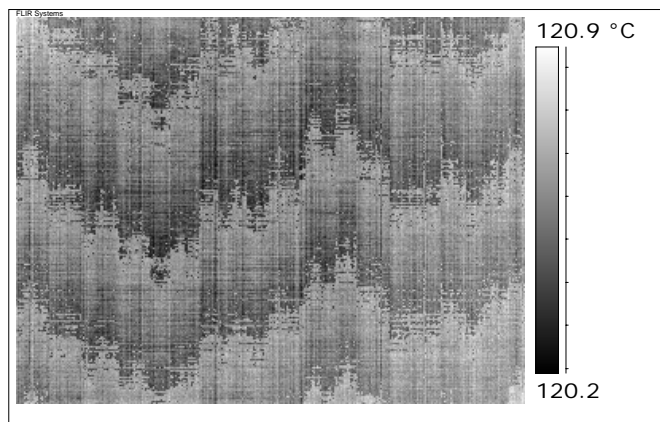


Figure 1.4: Image showing a very clear example of the "shadow" artifact.

1.2.1 Observations

It will be beneficial to make some observations about the characteristics of this problem. Firstly, notice that we are dealing with both global and local artifacts. The rings in Figure 1.3 are an example of a local artifact and the shadows in Figure 1.4 are an example of a global artifact. The global artifacts are best described by the image as a whole (unless of course they have a repeating pattern that, in a fractal-like way, is self-similar over a range of scales), while the local artifacts are insignificant to the image as a whole and need to be characterized by image patches containing only the area of the artifact. The local artifacts will most likely not be evident in any global information extracted from the image.

Secondly it is important to consider that the camera will have to be provoked during the testing procedure in order for the artifacts to show up. Therefore we would rather not make too many

assumptions about the conditions during which the test images were taken. But in some cases it might be necessary to assume the test image is acquired against a uniform radiation.

Thirdly there is of course a certain variance in appearance of artifacts within each class. While they are similar, they may vary greatly on some attributes (or features). For example, the individual instances of one type of artifact might all have almost the same shape while varying greatly in size. In this case, shape would be a good feature to use for classification and size would not be good. A more concrete example is that while some classes of artifacts might be well described by the actual pixel values, other classes of artifacts will display great variance in image space. It will be a challenge to find those features that characterize each class of artifacts well.

Finally, considering the detection of a given artifact, there will be two classes of images: those containing the artifact and those not containing the artifact. If the features are chosen wisely, we will only need a moderate amount of training images to characterize the variability within the class of images containing the artifact. But how do we characterize the extreme variability in the class of images not containing the artifact. This is a problem that has to be solved, preferably without the need for an exhaustive set of negative training images.

1.3 Report Outline

The material in this report will be organized as follows:

Chapter 2 – Computer Vision Background contains an overview of relevant theories from computer vision.

Chapter 3 – Machine Learning Background contains an overview of relevant theories from machine learning.

Chapter 4 – General Architecture contains a short description of a general architecture for an artifact detection system and shows how a detector for a specific artifact can be implemented.

Chapter 5 – Features is a collection of features and measures that can be used for artifact detection. This is the main contribution of this thesis.

Chapter 6 – Results presents the most important results and contains a comparison of the performance of the different features when used in combination with a classifier for artifact detection.

Chapter 7 – Discussion contains an analysis of the results from chapter five along with suggestions for future improvements. The outcome of the results analysis is a better understanding for how the artifacts should be measured and some concrete suggestions are made.

Chapter 8 – Summary and Conclusions contains a summary of the progress that has been made toward artifact detection and a conclusion.

2 Computer Vision Background

This chapter contains a review of relevant techniques from computer vision.

The material presented in this chapter relate to the extraction of discriminative information from images. From now on, such discriminative information will be termed feature. Thus the goal is to extract good features that will enable a machine learning algorithm to pick out images, or image patches, containing artifacts. When searching for a good way to do that it will be useful to keep in mind the observations made in section 1.2.1:

- A good feature will be reasonably invariant to changes in the background, for example global changes in intensity.
- A feature that gives a good description of one class of artifacts might not be a good descriptor of another class of artifacts. A feature is a good descriptor of a class of artifacts if the feature is invariant to variations within that class and at the same time has good discriminative power to separate that class from other classes of artifacts and from images with no artifacts.

The material in this chapter is organized in a general to specific manner. Sections 2.1 to 2.4 discuss general purpose image transformations, filters, methods for interest point detection and image statistics. Sections 2.5 to 2.8 discuss material that is more specifically related to extracting information from images for purposes of classifying, recognizing or describing the image.

2.1 Image Transforms

Image transformation might be important for pre-processing an image to enable extraction of good features. Transforming an image can generally be thought of as representing the image in another space than image space. The new representation may have advantageous properties, for example it might concentrate the energy (or visual content) of the image into just a few elements in the new coordinate vector. This property is possessed by the Fourier transform and the wavelet transform, which is why they are used for image compression (only a small portion of the new coordinate vector need be stored to retain most of the visual content). Each transform is good for analyzing some aspect of the image; the Fourier transform allows frequency analysis and the wavelet transform reveals the scale space properties of the image. Below, these transforms, along with some other transforms commonly used in image and signal processing, will be described briefly.

One transform worth mentioning is the generalized Hough transform, made popular by [1]. The Hough transform can be used to detect any shape (in a binary image), for example lines or circles, that can be described by a set of parameters. These shapes show up as maxima in the so called Hough space, defined by the parameters describing the shape (for example, the Hough space used for circle detection will have three dimensions corresponding to the coordinates of the circle center and the radius of the circle). The transform is defined as a voting procedure. In the case of circle detection in a binary image, each white pixel will vote for all circles that it touches, incrementing the corresponding Hough space element. After each white pixel in the binary image has cast votes on the circles that it might be part of, the maxima in Hough space will represent the circles which have most contributing pixels.

As mentioned in the introduction of this section, the discrete Fourier transform, which is used to characterize an image in the frequency domain, is also important for image analysis. Fourier transforming an image means that the image is projected onto a set of trigonometric basis functions. In the two-dimensional case, the basis functions are called plane waves. An example of a plane wave is the function $f(x, y) = A \cdot \cos(a \cdot x + b \cdot y)$. The image is viewed as the superposition of a number of such plane waves and the output of the transform is basically the

amplitudes of the constituent waves. Large amplitude means that the corresponding plane wave is an influential component of the image. This gives information about which frequencies and wavefront orientations dominate the image. The original image can be reconstructed from its Fourier transform.

Finally we visit the wavelet transform, which characterizes an image in the scale domain. It can also be thought of as the projection of the image onto a set of basis functions. Unlike the Fourier transform, however, the wavelet basis functions have limited spatial extent. The discrete two-dimensional wavelet transform, which is used for image analysis, represents the image as a series of approximations and details. Each approximation represents the image at a coarser scale than the previous approximation and the corresponding detail contains the difference between the two consecutive approximations. A complete tutorial on wavelet decomposition is beyond the scope of this text, but a good description can be found in for example [37]. Wavelet-based methods have been used successfully to detect salient points (the term “feature points” is often used in reference to salient interest points, but is not used here to avoid confusion with the term “feature” introduced earlier to mean information that discriminates a class of artifacts) in images [2][3][4]. This will be discussed further in section 2.4.

2.2 Filters

There are many basic image filters that might be useful in the feature extraction process. Most of these basic filters are described in any elementary book on computer vision, for example [5]. Filtering is often used to reduce the effects of noise. Three examples of noise reducing filters are the mean, median and Gaussian filters. The mean filter simply replaces the intensity value of a pixel with the mean of the intensity values in a local neighborhood around that pixel. The Gaussian filter is similar, but weights the intensity values of the surrounding pixels according to a Gaussian function. The median filter instead uses the median intensity in the local neighborhood, and has the advantage of preserving edges (sharp intensity contrasts), which are smoothed out by the mean and Gaussian filters. These filters are basically low-pass filters and their application to an image results in a dampening of the coefficients corresponding to high spatial frequencies in the Fourier domain.

By taking the difference of Gaussian filter kernels with different widths (the image resulting from applying a wide Gaussian kernel is subtracted from an image resulting from the application of a narrower kernel) one can produce a band-pass filter. This approach is called Difference of Gaussians (DoG) and can be used to achieve a wavelet-like transform [2].

The DoG is really an approximation of another filter created by taking Laplacians of Gaussians, which also can be used for scale-space analysis [6]. The Laplacian of Gaussian (LoG) filter can also be used in a second derivative approach to edge detection [7], since applying this filter corresponds to performing the Laplacian operator on a smoothed version of the original image. Edges are found by looking for zero crossings of the filter response, which would correspond to points of maximal gradient magnitude in the original image.

A more commonly used method for edge detection is the Canny edge detector [8], which is a first derivative approach to edge detection and looks at the gradient magnitude directly. It uses non-maximum suppression to find local maxima of the gradient magnitude. There has also been some more recent development in the area of edge detection. The algorithm described in [9] provides a mechanism for automatically selecting scale levels when detecting edges and ridges, so that diffuse edges will be detected at coarse scales and sharp edges will be detected at fine scales.

2.3 Image Statistics

In this section we shall look at some statistical measures that have been used for describing images and determining the similarity between different images.

A common and simple way of describing images is to use histograms of different attributes. If a spatial dependency is desired, the image can be first divided into regions and then one histogram can be generated for each region. There are several similarity measures for histograms, the simplest being the Euclidian distance. An example of a more sophisticated measure is the Earth Mover's Distance (EMD), which is described in [25]. A very common histogram distance measure is the χ^2 -distance. The χ^2 distance between two histograms H_1 and H_2 is defined as:

$$D_{\chi^2}(H_1, H_2) = \sum_{i=1}^n \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \quad [2.1]$$

Color (or intensity, if the image is gray-scale) is probably the first attribute that comes to mind for generating a histogram. The color space is then divided into discrete bins and the number of pixels in each bin is counted. The number of pixels in each bin is finally divided by the total number of pixels in the image in a normalization step. Other image properties commonly described by histograms include gradient direction and gradient magnitude.

Variance (or standard deviation) and mean of the pixel intensities in an image are also common measures. Usually a large intensity variance in an image region is taken as a sign of complexity or structure in that region; while small variance usually means that the visual content of the region is scarce.

2.4 Interest Point Detection and Saliency

Interest point detection is the first step in many computer vision algorithms. The purpose is to reduce the information in an image to a few interesting or salient points or regions. Once the interest points are found, they can be used to, for example, recognize objects in the image or match points in the image to interest points in another image in order to calculate the transformation (homography) between the two images. For the purposes of matching interest points in different images, it is important to obtain a description of the interest point that is invariant to, for example, changes in perspective or global light level. This desire is usually in conflict with the desire to minimize the probability of false matches. As a result, many different interest point detectors exist to suit different needs. One example of a simple interest point detector is the Harris corner detector [10]; it yields a high response to image points that show two-dimensional structure (in contrast, an edge is a one-dimensional structure). The Harris corner detector belongs to a class of interest point detectors that view the image as a discrete approximation to a function surface. Other interest point detectors work over several different scales and select the scale that maximizes the entropy [3][4] or the result of a differential operator [11]. Whereas the detector described in [3][4] considers points that give high response over a short range of scales to be salient, the popular SIFT detector [2] prefers points that give a high response over a wider range of scales (of course these methods are also separated by the method of response calculation). Since these different interest point detectors use different working definitions of saliency, it is natural that they detect different kinds of image attributes. For example, while the Harris corner detector detects corner points the detector described in [3] and [4] shows a preference for blob-like regions.

In order to get inspiration for the construction of good features to describe and discriminate the different image artifacts it will be valuable to look at what information the successful interest point detectors store about the detected interest points to achieve invariant matching to interest

points in other images. Taking SIFT [2] as an example, the calculation of a descriptor (which is done for each image point that the detector considers salient) follows the steps listed below:

1. The dominating gradient direction of the interest point is calculated by finding the maximum of a 36 bin histogram of gradient directions in a neighborhood of the interest point. The directions are weighted by the gradient magnitude and a Gaussian function centered at the interest point.
2. A 16 by 16 pixel neighborhood around the interest point is divided up into 16 four by four sub-regions.
3. An 8-bin gradient direction histogram is created for each sub region. The gradient directions are rotated relative to the dominating direction from step one and the corresponding gradient magnitudes are weighted by a Gaussian before they are sorted into the histogram. Trilinear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins.
4. A descriptor is formed by concatenating the histograms for all the sub regions to form a 128 element feature vector. The feature vector is then normalized to unit length, the values are thresholded to be below 0.2 and finally the feature vector is renormalized.

A good evaluation of the performance of different interest region/point descriptors may be found in [12], where an extension of the SIFT descriptor named gradient location and orientation histogram (GLOH) is also suggested. Another extension of the SIFT descriptor is suggested in [14], where Principal Components Analysis (PCA) is applied to reduce the dimensionality of the descriptor. The local descriptors described in these papers will be an important source of inspiration in selecting good features to use for detecting and discriminating image artifacts.

2.5 Texture Classification

Research on texture classification is focused on describing texture in a way that groups visually similar textures together and separates dissimilar textures. Often texture description is based on statistical properties. One early approach was to use metrics derived from the so called gray-level co-occurrence matrix (GLCM), C , which is calculated from an image I as follows:

$$C(i, j) = \sum_{p=1}^n \sum_{q=1}^m \begin{cases} 1, & \text{if } I(p, q) = i \text{ and } I(p + \Delta x, q + \Delta y) = j \\ 0, & \text{otherwise} \end{cases} \quad [2.2]$$

The GLCM characterizes the co-occurrence of gray-level intensity values of pixels with a relative offset defined by the parameter $(\Delta x, \Delta y)$. GLCM-based features have been used for image classification [38].

Histograms of so called Local Binary Patterns (LBP) can be used to characterize textures [26]. The LBP histograms can then be used together with some histogram distance measure to get an estimated similarity between images.

Other alternatives for texture-based similarity measures are Gabor filters, Markov random fields (MRF) and filter banks. In the first case the Gabor filter response is used to describe texture. The MRF approach attempts to represent the probability distribution of pixel intensity conditioned on the local neighborhood. The difficulty is how to achieve a compact representation given the huge space of possible neighborhoods. Finally the filter banks use the response of a set of filters for texture description. In [13] the MRF approach is compared to the use of filter banks, with the conclusion that MRF performs better than filter banks (which are described in for example [16] and [17]) when used for texture classification. Yet another descriptor, called “spin image”, is detailed in [15] and compared to Gabor filters for texture classification. Finally, a new interesting algorithm for finding texture regularities in images is proposed in [22].

Some of these methods will be interesting candidates for features for artifact detection, in which case these methods will be given a more in-depth coverage later.

2.6 Content-based Image Retrieval

Content-based image retrieval (CBIR) is a collection of methods for querying databases of images based on the image content (in contrast to context-based image retrieval, which uses associated text for indexing images). A CBIR query can be a keyword, a sample image or a sketch. The methods used in CBIR usually involve some sort of statistical learning or machine learning. This section contains a review of some different approaches used in CBIR (a more comprehensive review can be found in [20]).

Consider the scenario when an image database is queried by a keyword, which usually describes some object or concept. In this setting the general approach is to use a set of labeled example images to build a model (which can either be a statistical model or a machine learning representation) of each object or concept of interest (so the set of possible query words is limited to the objects and concepts that have appeared in the example images). When a user then queries the system using some keyword, the system checks each image in the database to see how closely it agrees with the model corresponding to the entered keyword. All images that are in close enough agreement with the model are returned to the user. The interesting question is of course how to construct the models from a set of example images. One appealing method is presented in [18] and [19], which builds one classifier for each object or concept of interest. The classifier for some object or concept is constructed from a set of labeled example images in the following way, which is divided into a generative and a discriminative part:

1. In the generative part, each image is segmented based on, for example, color or texture. After segmentation each image will contribute a small number of colors or textures (one for each segment) as vectors to color-space or texture-space.
2. The Expectation Maximization (EM) algorithm is used to form a Gaussian mixture model in color-space or texture-space. The mixture model describes the distribution of colors or textures of the object or concept.
3. In the discriminative part, an image is segmented as before to obtain the dominating color or texture vectors. For each component in the Gaussian mixture model, a score is computed based on the agreement between that component and the color or texture vectors from segmentation. These scores make up a feature vector with the same length as the number of components in the Gaussian mixture model.
4. The feature vectors from step three are then fed into some classifier, such as a multi-layered perceptron or a support vector machine, which is trained to decide if the given image contains the object or not, based on the scores in the feature vector.

Another similar example is the ALIP system [21], which uses a two-dimensional multi-resolution hidden Markov model (2D MHMM) to model each object or concept. The models are built from a set of labeled example images and are then collected in a dictionary to map between keywords and models.

The methods presented above build models of low-level image characteristics, such as distributions of color and texture. A more high-level approach is to build models of the spatial distribution of interest points to represent different classes of objects. This approach is taken in for example [23] and [24]. However, these results seem less helpful in the search for a solution to the problem treated in this report.

2.7 Template Matching

Template matching is a very simple method used to find image patches similar to a given reference patch. It can be regarded as a flexible interest point detector. The template is compared to different image patches in the target image and usually a predefined threshold on some distance measure is used to determine if a match was found. The distance measure is usually taken to be the cross-correlation or the normalized cross-correlation between the reference patch and the patch in the target image.

2.8 Face Detection

Face detection is an instance of the more general field of specific object recognition. Many of the techniques used for face detection can also be used for detection of other objects, such as cars or motorcycles, in images. These methods relate to the detection of local artifacts, such as spots.

Generally many face detection algorithms use an explicit model of the intraclass variability of some property or properties of face images. For example, the popular eigenfaces algorithm [39] uses the subspace of image space defined by the principal component directions of a set of example face images to model the intraclass variability of faces. The orthogonal distance in image space from a new image to that subspace is a measure of how closely the new image corresponds to the face class model and can thus be used for face detection.

Other approaches use neural networks [31] or other machine learning models (such as support vector machines) to represent the class of faces. These approaches encounter an interesting difficulty in dealing with the large variability in the class of non-faces. The machine learning method is given a batch of classified example images with and without faces in them. It uses this example batch to build its internal representation of the face and non-face classes. If it is then used to scan an image for faces, it will encounter many image patches from the non-face class and maybe a few image patches from the face class. Most probably it will yield many false positive face detections. The problem is that while the face examples in the example batch might have captured the variability within the face class, the non-face examples probably did not capture the variability in the much larger class of non-faces. The problem of false positives can be solved by letting the machine learning algorithm scan some images with no faces and recycling all positive detections as negative training examples [31].

3 Machine Learning Background

This chapter contains a review of relevant techniques from machine learning. Methods from machine learning can be used to classify images based on their feature space representation.

Each algorithm presented in this chapter has the purpose of classifying unseen data points after undergoing a training phase, where the algorithm is presented with a set of previously classified data points. More detailed descriptions of the machine learning algorithms presented here can be found in basic texts on machine learning or artificial intelligence, such as [27] or [28].

The simplest classifier imaginable is just a threshold. If the feature is single-valued and has the property that the members of one class yield small feature values and the members of the other class yields large feature values, a simple threshold is sufficient to separate the classes in feature space. Classification by thresholding can be easily extended to the case of vector valued features. In this case the threshold is replaced by a plane in multidimensional feature space. In this case we call the plane a decision plane and if two classes are separated by such a plane, they are called linearly separable. In most cases the classes will not be linearly separable and a more complex decision surface is needed. The machine-learning methods below can represent different kinds of decision surfaces and knowledge about the distribution of the two classes in feature space will help choosing the best machine learning algorithm for the classification task.

3.1 Nearest Neighbor

A very simple classification algorithm is the nearest neighbor (NN) algorithm. This algorithm simply saves all the labeled data points from the training set and then assigns any unseen query data-point the same class as the closest data point from the training set. Usually the feature space will be \mathcal{R}^n with the Euclidian distance.

A generalization of the NN algorithm is the k -nearest neighbor (k -NN) algorithm, which instead uses the most frequent class among the k closest data points from the training set as the label for any unseen data-point. A further generalization of the basic algorithm is to weight the k nearest neighbors according to their distance from the query data-point.

The advantages of this algorithm are its simplicity and its ability to create relatively complex decision boundaries in feature space. However, many classification problems require a more elaborate solution.

3.2 Multi-layered Perceptron

The multi-layered perceptron (MLP) is a type of artificial neural network (ANN). In the case of classification of feature vectors in \mathcal{R}^n , a typical network structure is shown in Figure 3.1. Since the feature space is assumed to be \mathcal{R}^n , the input to the MLP is an n -dimensional vector. It is further assumed that there are c classes in total and the output from the MLP will be a c -dimensional vector, indicating the most likely class of the input vector by a large response from the corresponding output neuron. The detailed picture of the first hidden neuron illustrates the processing done by each neuron. First each element of the input vector is multiplied by a weight, and then the weighted input elements are summed and used as input to a so called transfer function. In this case the tanh-function is used, but there are many other possibilities and the choice is usually influenced by the characteristics of the problem. Finally the output of the transfer function is used as input to the neurons in the output layer. Each neuron also has a so called bias term, which is not depicted in the figure.

The MLP is a feed-forward network. This separates it from the class of feed-back networks, which have the common property that their network structure contains loops. The feed-forward networks are usually trained using an algorithm called backpropagation. This implies doing a local minimization of the classification error on a batch of training examples. The error is defined as a continuous function of the weights in the network. The weights represent the learned knowledge of the trained network.

The advantage of the MLP is that it is a versatile and general classifier. On the other hand, training can be slow and there are many parameters to tune, for example the number of hidden neurons and the choice of transfer functions. Furthermore, the MLP only achieves a local minimum of the error function and if there are several solutions that give zero error, there is no mechanism for choosing the “best” one.

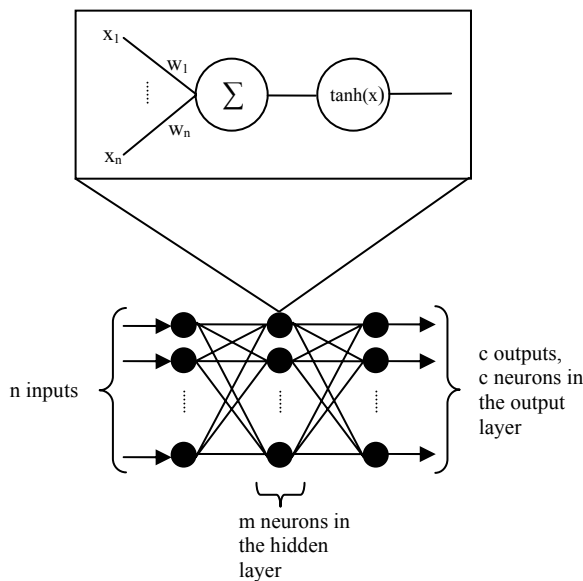


Figure 3.1: An example of a typical MLP used for classification.

3.3 Support Vector Machine

The support vector machine (SVM) addresses many of the shortcomings of the MLP. Given a training set of data points from two classes, the SVM learning algorithm finds the hyperplane or separator that yields maximum margin. The data points that are active in constraining the width of the margin are called support vectors. The situation is illustrated in Figure 3.2. In this case the two classes are linearly separable and the example seems to imply that linear separability of classes is a requirement for the algorithm to work. That would of course be a major limitation and fortunately there is a way to get around this problem. The solution is to spread out non-linearly separable data in high-dimensional space and then use so called kernel functions to calculate scalar products of high-dimensional vectors without explicitly computing these vectors. Thus the task of finding the optimal separator in high-dimensional space can be done without ever representing the data points in this space explicitly.

One advantage of using SVMs for classification is that there are fewer design parameters to consider. The only major design choice involves picking which kernel function to use and setting the parameters of that kernel function. SVMs also address the issue of which separator to use if there are several alternatives by taking the one that maximizes the margin. This makes SVMs a good alternative in situations when the training set is limited. Only the boundary examples are really important, since they are the candidates to be stored as support vectors.

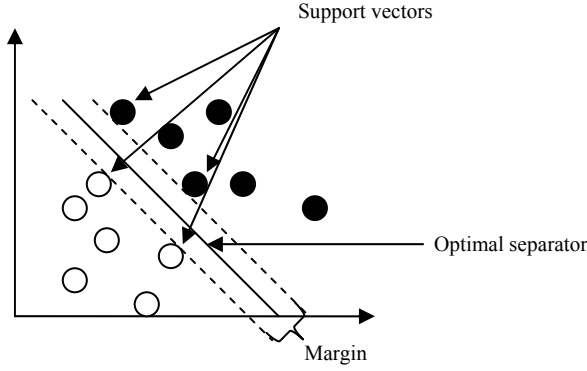


Figure 3.2: The figure shows a training set with data points of two linearly separable classes. The SVM learning algorithm finds the optimal separator.

3.4 Bayesian Classifiers

The common denominator of the Bayesian methods is that they rely on Bayes' theorem (where $P(c|D)$ is the conditional probability that the class is c given observed data point D):

$$P(c | D) = \frac{P(D | c) \cdot P(c)}{P(D)} \quad [3.1]$$

Usually, the probability distribution $P(D|c)$ can be estimated from the training set (this is done by counting when D is a discrete variable or by assuming some probability distribution and estimating the parameters of that distribution if D is continuous). A simple way to classify a query data point is to choose the class c that maximises the conditional probability of observing the query point:

$$c = \arg \max_{c_i \in C} P(D | c) \quad [3.2]$$

This is called the maximum likelihood (ML) hypothesis. If we have knowledge about the prior probabilities, $P(c)$, for each class, we can use Bayes' theorem to get the so called maximum a posteriori (MAP) hypothesis:

$$c = \arg \max_{c_i \in C} P(c | D) = \arg \max_{c_i \in C} \frac{P(D | c) \cdot P(c)}{P(D)} = \arg \max_{c_i \in C} P(D | c) \cdot P(c) \quad [3.3]$$

Other algorithms using the Bayesian approach are the naïve Bayes classifier and the optimal Bayes classifier, which are both given detailed descriptions in [27].

The advantage of using Bayesian classifiers is that we get explicit probabilities from the training phase, which of course facilitates analysis and interpretation of the results. The weakness is that strong assumptions are sometimes needed to estimate some of the needed parameters, for example the prior probabilities for each class.

3.5 Parameter Selection and Dimensionality Reduction

It is often desirable to remove unnecessary or redundant information and reduce the dimensionality of data. A reasonable first step is to try to remove one or more of the dimensions of the original feature space. A good review of methods for selection of relevant parameters is given in for example [30].

A widely used method for dimensionality reduction is principal components analysis (PCA), which uses the covariance matrix of a data set to get the directions in feature space where the points in the data set exhibit most variation. A description of how to combine parameter selection and dimensionality reduction can be found in [29].

4 General Architecture

This chapter contains a description of a proposed architecture for an artifact-detection system.

The system proposed for detecting artifacts has a modular design, which is illustrated in Figure 4.1. This architecture will be given a top-down description, starting with the system core. The system core handles the user interaction and stores information about which detectors should be active at any given time. It is important to be able to turn detectors off, since the detection of some artifact might require special testing conditions (such as placing some object in front of the camera), which could yield false positive detection of some other artifact if the detector for that artifact is turned on. The system core also interfaces with the test image acquisition procedure and might even control it and request images. Thus the system core defines or at least has knowledge of the test sequence. The system core then distributes the acquired test images to the detectors. The detectors store the desired parameter settings for the feature computation modules and know which features to use in which combination. The detectors also store the classifiers (which might be support vector machines or simple thresholds if the feature is single-valued and designed for classification by thresholding). The detector calls the feature computation modules. In the case of local artifacts, such as spots, the feature computation modules will be called many times while the detector scans the image.

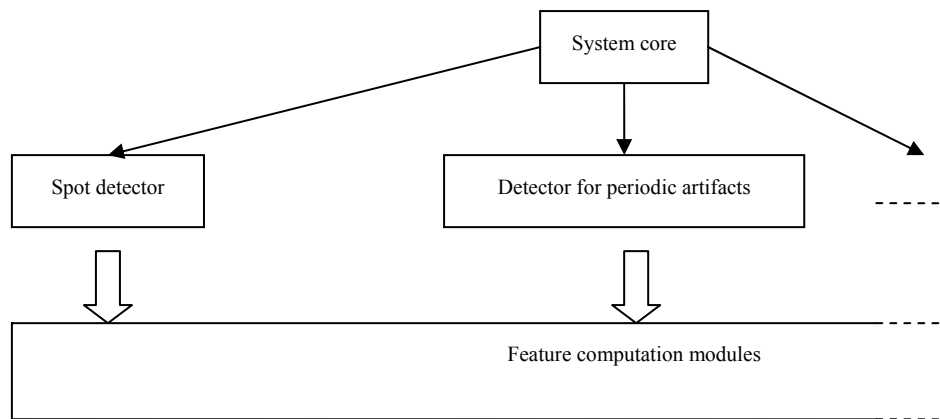


Figure 4.1: Illustration of the proposed artifact detection system architecture.

Alternatively, if only single valued quality measures are used, a simpler architecture might be better. The simpler architecture would consist of only a system core and underlying modules for computing quality measures. The system core would then handle image acquisition, user interaction and store all parameter settings. The system core would call its underlying modules to request computation of quality measures. The system core could then threshold the quality measures to determine if the image meets the quality requirements. Depending on the quality measures, the system core might pass, fail or mark the camera for manual inspection. Figure 4.2 shows this architecture.

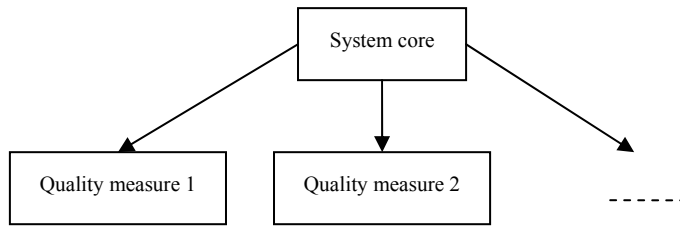


Figure 4.2: Illustration of the alternative architecture.

The rest of this chapter is dedicated to describing the detectors. The main component of a detector is the classifier. The training and subsequent use of the classifier are described in sections 4.1 and 4.2 respectively.

4.1 Training the Classifiers

The support vector machine (SVM) has been chosen as a suitable classifier for vector valued features. The reason for this choice is mainly that the SVM does not need an extensive set of training examples (it does, however, need representative boundary examples that will be good candidates for serving as support vectors). The second reason for choosing the SVM is that it is generally quicker to train than for example the multi-layer perceptron. In the case of single valued features a simple threshold can be used for classification. In this section we describe the situation when the classification is based on several parameters.

The classifier is trained using a batch of example images with known classifications. The process is illustrated in Figure 4.2. Each image in the example batch is passed to each of the feature calculation algorithms (often a single feature will be used). The feature values returned from the feature calculation routines are concatenated to form one big feature vector. Thus each image in the example batch is represented by a point in the final parameter space, which in some cases will be big in the sense that it has many dimensions. It should be stressed that the final parameter space need not have many dimensions, as for example if only one single-valued feature is used; then the final parameter space will be one-dimensional.

Given the parameter space representation of the images in the batch, the classifier tries to find a decision boundary that separates the two classes. If the classes are not separable, the classifier will try to find a decision boundary that separates them as good as possible. There is a risk that the classifier adapts too much to the training batch and the decision boundary gets very convoluted in parameter space. This phenomenon is called overfitting and usually leads to bad performance when the classifier is applied to unseen images. The parameter that controls the tendency for overfitting in support vector machines is the cost of constraint violation (slack). A high cost will increase the tendency for overfitting, while a low cost will decrease that tendency. However, setting the cost too low will prevent the SVM from learning.

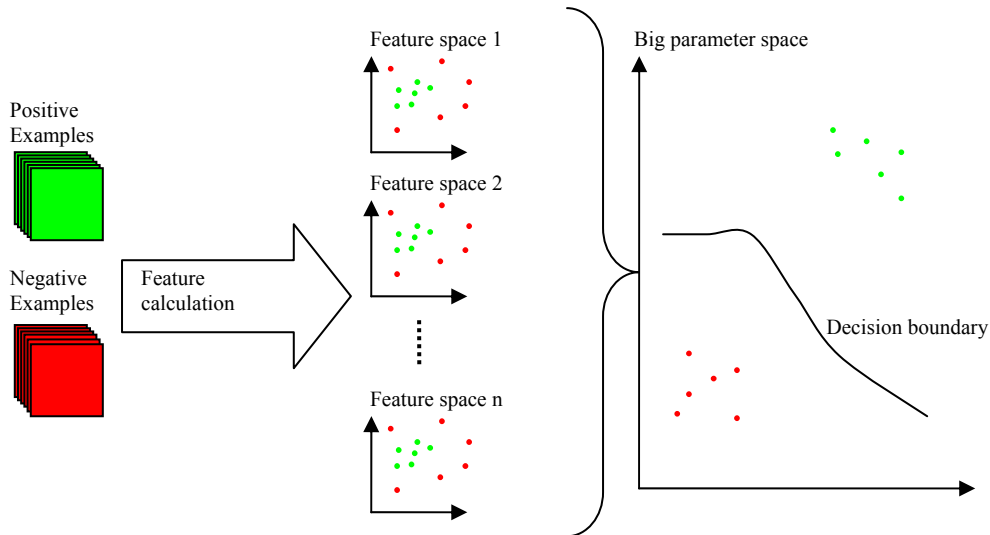


Figure 4.2: Illustration of classifier training.

It is important to also evaluate the effects of training. If there is an abundance of training examples, some can be withheld from the classifier during training and can be used to test the performance of the classifier after training. Unfortunately the supply of example images for the different artifacts targeted in this thesis is scarce and withholding a portion of the set of training images would leave too few examples for the classifier to train on. Luckily there is a procedure called cross-validation that can be used to evaluate the performance of the classifier while withholding a minimum of training examples from the classifier during training. In its extreme form this procedure is termed leave-one-out and works as follows:

1. One training example is removed from the training set.
2. The classifier uses the rest of the training examples to find a decision boundary.
3. The trained classifier is used to classify the withheld training example. If the given classification agrees with the known class of the example, a counter storing the number of correct classifications is increased by one.
4. Another example is withheld from the training set and the process is repeated from step two.
5. After each training example has been withheld once, the process ends and the counter is divided by the total number of training examples to obtain a performance measure.

Leave-one-out cross-validation will be used to determine the performance of the classifiers used for artifact detection (unless the classifier is a simple threshold, in which case the performance will be taken to be the number of correct classifications of images from the training set when the threshold is chosen optimally).

4.2 Detecting the Presence of Artifacts

After training, the classifier is stored and used to detect the presence of artifacts. Figure 4.3 illustrates the usage of the trained classifier. The test image is passed to the feature computation routines and the parameter space representation of the image is returned (of course the same features are used for classification as for training). The classifier uses its decision boundary in parameter space to determine if an artifact is present in the image.

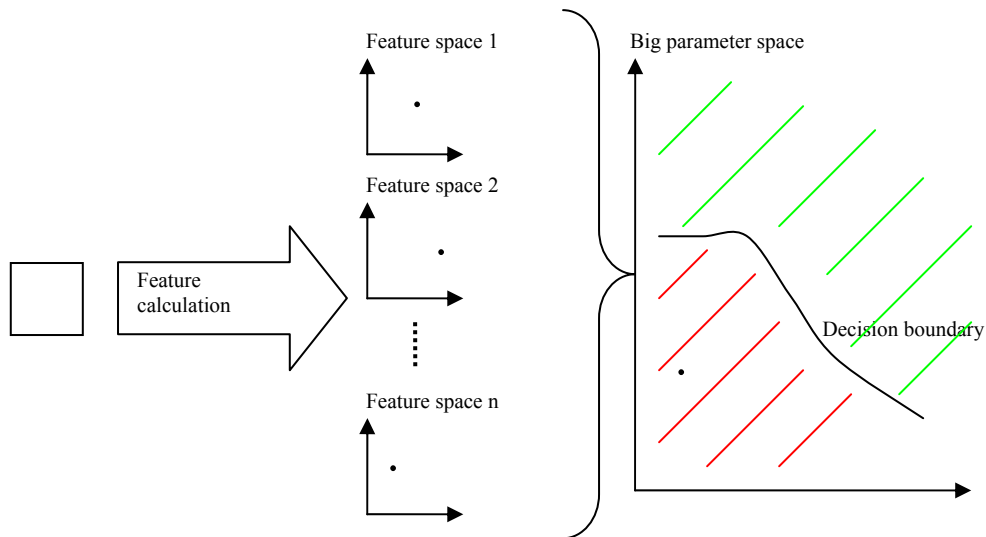


Figure 4.3: Illustration of the usage of a trained classifier.

4.3 Avoiding False Positives

During training the classifier is presented with a number of examples of the artifact it is supposed to detect and a number of example images without that artifact. After training the classifier will encounter a large amount of images without the artifact and only a few images with the artifact. The problem here is that the training examples without artifacts probably did not account for the large variability within this class and therefore many images without artifacts will be misclassified as having artifacts.

The solution to this problem is to tune the decision boundary in parameter space to better capture the actual boundary between the two classes. Figure 4.4 illustrates the decision boundary in parameter space before and after tuning. The tuning is done by finding images that are erroneously classified as having artifacts. Using these images as training examples will force the decision boundary closer to the artifact class. This process can be repeated iteratively until the desired decision boundary is achieved and the frequency of false detections is down to a tolerable level.

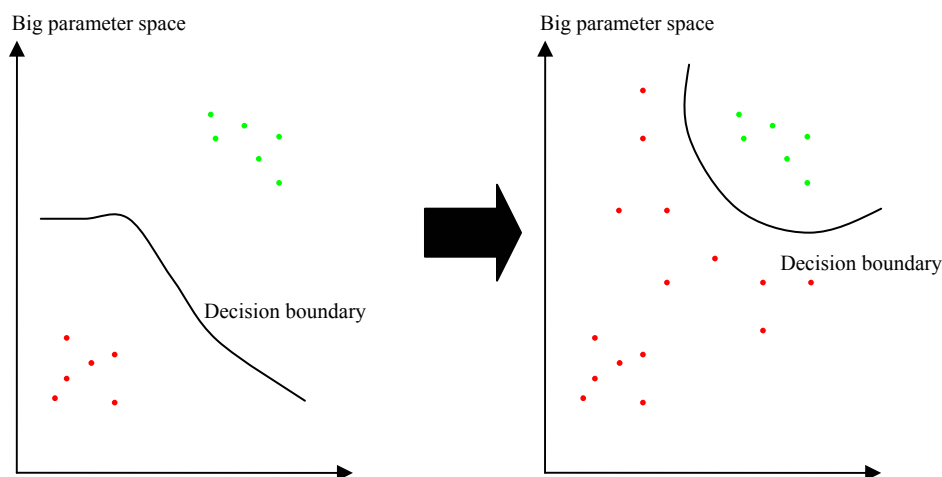


Figure 4.4: Tuning the decision boundary to avoid false detections.

5 Features

This chapter is a collection of features and measures that were proposed for detection of four different types of artifacts.

In this chapter all the features proposed for detection of the different artifacts are described. The description of each feature will begin with the general idea behind the feature and a motivation why the feature might work to distinguish images with the artifact from images without the artifact. Then the feature calculation algorithm will be described, making ample use of figures to illustrate the workings of the algorithm. Finally, there will be a short discussion of any assumptions that the algorithm might rely on (for example regarding the conditions under which the images were acquired). This concluding discussion will also mention limitations of the algorithm, such as situations when the algorithm will not work, and also describe any invariance (for example, invariance to rotation means that if an image is rotated before being passed to the algorithm the output feature vector will still be the same as it would have been if the rotation had not been performed) that the algorithm might have or not have.

The feature descriptions are arranged according to the artifact they are applied to.

5.1 Spot Detection

The features suggested for spot detection are aimed at describing image patches so that patches with spots stand out.

5.1.1 Radon Profile

The radon profile feature is inspired by the radon transform. The motivation is that if an image patch contains a bright spot on a slightly darker background, the sum of pixel intensities along the rows and columns of the image should be large for the central rows and columns and gradually decrease toward the boundary of the image patch.

The first step of feature calculation is to apply a median filter (using a five by five square kernel) to reduce noise. Then the pixel intensities in the image patch are thresholded at the midpoint of the intensity range (not to be confused with the mean intensity) in the image patch, creating a binary patch of black and white pixels. If the central ninth of the image has a large portion of zeros (which will occur if the image patch contains a dark spot on a brighter background), the binary value of each pixel is toggled, changing ones to zeros and vice versa. The number of ones is then counted for each row and column and the counts are distributed into bins according to an exponential distribution over the distance from the row center to the bin center. The number of bins to use and the λ parameter of the exponential distribution are parameters of the algorithm. Figure 5.1 illustrates the final steps of performing the row and column summation of an eight by eight binary patch and distributing the sums into the radon profile array, which has five bins and uses $\lambda=0.1$. Figure 5.2 shows an example of an image patch containing a spot and the corresponding radon profile.

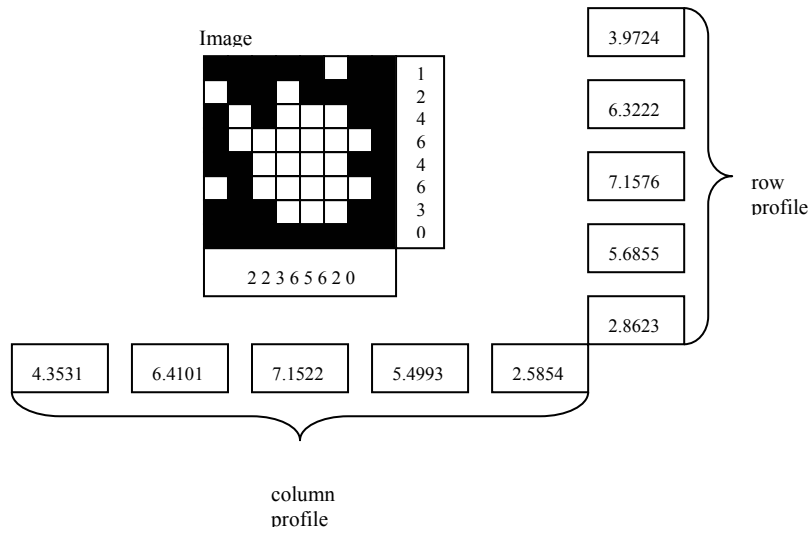


Figure 5.1: Illustrating the final steps of radon profile computation using five bins and $\lambda=0.1$.

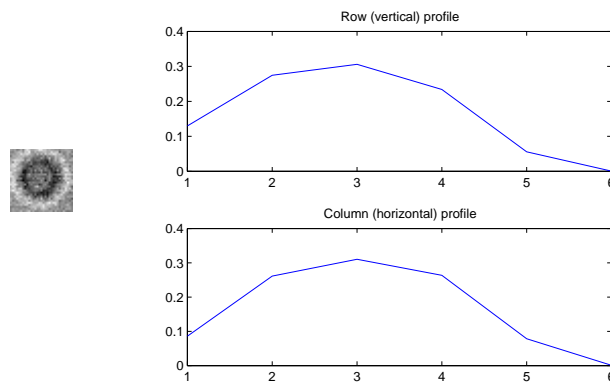


Figure 5.2: An example of an image patch containing a spot and the corresponding radon profile feature.

5.1.2 Edge Pixel Location

Figure 5.3 shows an image patch containing a spot along with a surface plot of the intensity values in the image patch before and after low-pass filtering. If an edge is defined as a local maximum of the gradient magnitude, edge pixels should be found in a circular region close to the boundary of a low-pass filtered image patch containing a spot. That is the motivation for this feature, which forms a feature vector from the percentage of edge pixels at different distances from the center of the image patch. The algorithm for calculating the feature vector is described below and Figure 5.4 shows an example of an image patch containing a spot, the corresponding binary frame of edge pixels and the final feature vector.

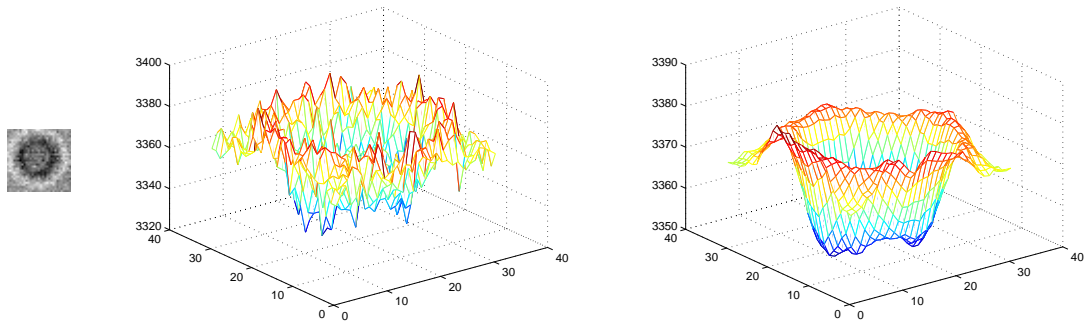


Figure 5.3: An image patch containing a spot and surface plots of the pixel intensities in the image patch before and after low-pass filtering (a Gaussian low-pass filter with $\sigma=4$ and a five by five kernel was used).

The algorithm for computing the edge pixel location feature vector takes two parameters, the number of steps to use in discretizing the distances to the center of the image patch (call this parameter n_{bins}) and the σ parameter of the Gaussian low-pass filter. The computation is done through the following steps:

1. The input image patch is low-pass filtered using a Gaussian kernel with the given value for σ . If the edge detector used in the next step includes low-pass filtering, this step should not be performed.
2. The filtered image patch is passed to an edge detection algorithm. In this case, Canny's edge detector (which includes Gaussian low-pass filtering) was used. The output of the edge detector is a binary frame of edge pixels.
3. An edge pixel counter with n_{bins} places is initialized to zero. For each edge pixel in the binary frame, the distance from the center of the frame is calculated the corresponding place in the edge pixel counter is increased by one.
4. For each place in the edge pixel counter, the count is divided by the total number of pixels that fall within the distance range of that place.
5. The normalized edge pixel counter is returned as the feature vector.

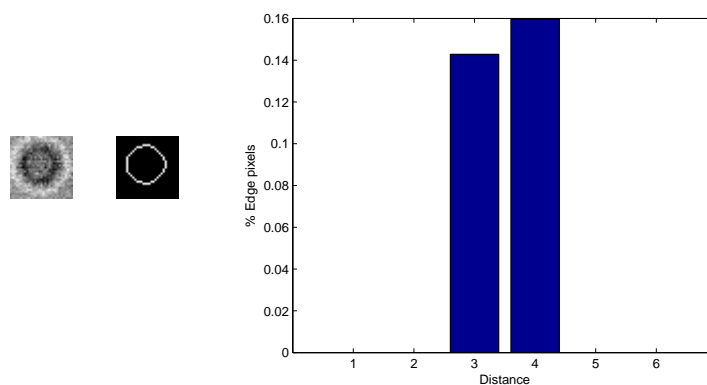


Figure 5.4: An image patch containing a spot, the corresponding binary edge frame and the edge pixel location feature vector, calculated using six discrete distance steps and $\sigma = 4$.

This feature is invariant to rotation of the image patch. It should also be invariant to scaling of the image intensities if a reasonable edge detector is used.

5.1.3 Spin Image

The spin image is a two-dimensional histogram computed from normalized image patches and is described in for example [15]. The two dimensions of the spin image histogram are distance from the center of the image patch and pixel intensity. For the image patches containing spots there is a strong correlation between the intensity of a pixel and its distance from the center of the patch. Thus these image patches should yield distinguishing spin images.

Before computing the spin image, the pixel intensities in the image patch are normalized. The first step in this feature calculation (which is not a part of the original spin image algorithm) is to compare the mean intensity in the central ninth of the image patch to the mean intensity of the whole image patch. If the central mean is lower than the total mean, the pixel intensities are reflected around the midpoint of the intensity range. The reason for this step is to make all spots appear bright on a dark background. The intensity range (which should now be $[0,1]$) is then divided into a number of discrete intervals. The range of distances from the center should also be normalized to the range $[0,1]$ and divided into a number of discrete intervals. The number of intervals to use for the discretization of the intensity and distance are parameters to the algorithm. The spin image histogram is initialized to be a matrix of zeros with as many rows as there are possible intensity values after discretization and as many columns as there are possible distance values after discretization. Then, for each pixel in the image patch, the discretized intensity and distance values determine the row and column of the element of the spin image that is increased by one. Figure 5.5 shows two normalized image patches containing spots and the corresponding spin images, using four discretization intervals for intensity and four discretization intervals for distance. Notice that for short distances there is an emphasis on high intensities and for larger distances there is an emphasis on low intensities, as expected.

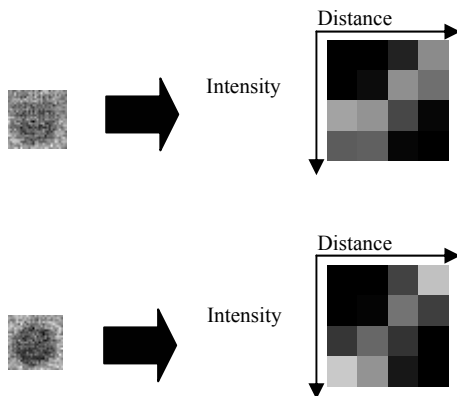


Figure 5.5: Two examples of image patches with spots and the corresponding spin images.

The spin image is invariant to rotation and intensity scaling of the image patch.

5.1.4 Gradient Direction Histogram

When motivating the edge pixel location feature, it was established that the intensity function of the filtered image patch with a spot should have a concentration of large gradient magnitudes in a circular region close to the boundary of the image patch. Continuing this reasoning, the direction of these gradients should point either toward the center of the patch (if the spot is bright on a slightly darker background) or away from the center of the patch (if the spot is dark on a slightly brighter background). That motivates the proposition of a feature that captures the distribution of gradient directions in different regions of the image. Figure 5.6 illustrates how the image patch can be divided into regions and what gradient directions are expected to dominate in each region if the image patch contains a bright spot on a darker background.

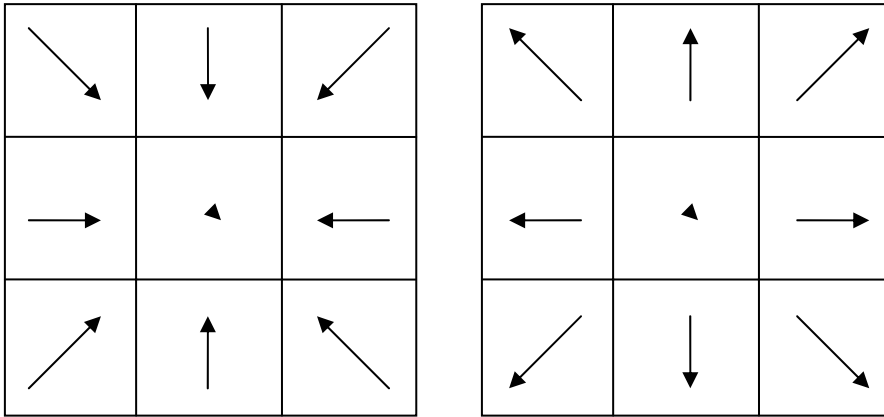


Figure 5.6: The (square) image patch is divided into nine (square) regions in a three by three grid. In each region of the leftmost image the gradient direction that is expected to dominate if the image patch contains a bright spot is shown by an arrow. In the rightmost image the expected situation for dark spots is shown.

The algorithm for computing this feature takes two parameters, the number of bins in the gradient direction histograms and the number of regions for subdivision of the image. However, considering Figure 5.6, it seems reasonable to fix the number of bins to four and sort angles φ and $\varphi + \pi$ into the same bin. Figure 5.7 shows the mapping of angles into bins using the unit circle. The number of bins in the gradient direction histograms will therefore be fixed to four and the only remaining parameter is the number of regions for subdivision of the image. We shall use the square root of the actual number of regions as parameter and call that parameter n_{reg} (for example, the image in Figure 5.6 has $n_{reg}=3$).

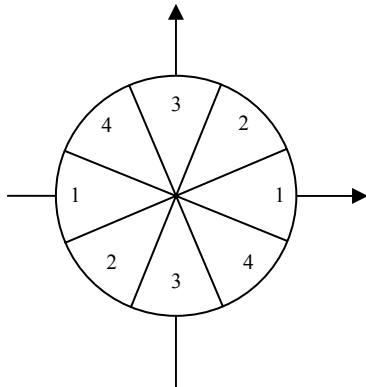


Figure 5.7: An illustration of the mapping of angles into four bins, dividing the unit circle. The bins are numbered from one to four.

The algorithm for computing the gradient direction histogram feature is given below:

1. Apply a Gaussian low-pass filter to the input image patch (use for example $\sigma=0.5$ and a five by five kernel).
2. Compute the gradient direction at each pixel, yielding a gradient direction frame.
3. For each pixel (direction) in the gradient direction frame, determine in which image region the pixel is located and sort the direction into the gradient direction histogram of that region.
4. Normalize the gradient direction histogram for each region by dividing by the total number of pixels in a region.
5. Return the normalized direction histograms as a single column vector.

This feature is not invariant to rotation but should be invariant to scaling of the intensities.

5.1.5 Downsampled Original

The spots actually look similar even in image space. There might be scaling differences, some spots are dark and some are bright and there are slight differences in shape, but otherwise the spots constitute a homogeneous group of image patches. Scaling differences can be reduced by normalization and dark spots can be made bright by reflecting the pixel intensities around the midpoint of the intensity range (making large intensities small and vice versa). Thus, there is reason to expect good classification performance when just using a slightly pre-processed image matrix as feature vector.

The algorithm for computing the downsampled original feature vector takes one parameter specifying the size of the output, for example eight by eight pixels. Call this parameter n_{pixels} . The algorithm is given below:

1. Compute the mean intensity of the whole input image patch and the mean of the central ninth of the patch. If the central mean is lower than the global mean, reflect the intensity value of each pixel around the midpoint of the intensity range of the image patch.
2. Resample the image patch to be n_{pixels} by n_{pixels} . Use bicubic interpolation and apply a Gaussian low-pass filter when downsampling.
3. Normalize the intensities of the resampled image to the range $[0,1]$.
4. Reshape the normalized resampled image to a column vector and return.

This feature is invariant to scaling but not to rotation.

5.1.6 Distance from Center and Intensity Correlation

This is a single-valued feature inspired by the spin image feature, which showed (not surprisingly) that the intensity value of a pixel is strongly correlated to its distance from the center of the image patch (see Figure 5.5). Two common measures of correlation are the covariance and the correlation coefficient, which is basically a dimensionless, normalized version of the covariance. A more detailed description of the covariance, the correlation coefficient and other statistical measures is given in [32]. We shall use the covariance to measure correlation and precede the covariance calculation with a normalization step. The expected result is that bright spots will yield a negative covariance, dark spots will yield a positive covariance and non-spots will yield a covariance close to zero (since the intensity and distance from the center should be uncorrelated in this case). The algorithm for calculating this feature takes no parameters and is described below:

1. Apply a low-pass filter to the input image patch. A median filter with a three by three kernel was used in this case. (Using a Gaussian kernel instead would give a parameter, σ , to tune, which might give the ability to improve performance.)
2. Normalize the pixel intensities to the range $[0,1]$.
3. For each pixel, calculate the distance from the center of the image patch.
4. Normalize the distances to the range $[0,1]$.
5. Calculate and return the covariance between the normalized distances and the normalized intensities.

This feature is invariant to scaling and rotation.

5.2 Detection of Periodic Artifacts

The class of periodical artifacts is characterized by the appearance of almost parallel “lines” in the image. Some examples of images containing artifacts from this class are shown in Figure 5.8.

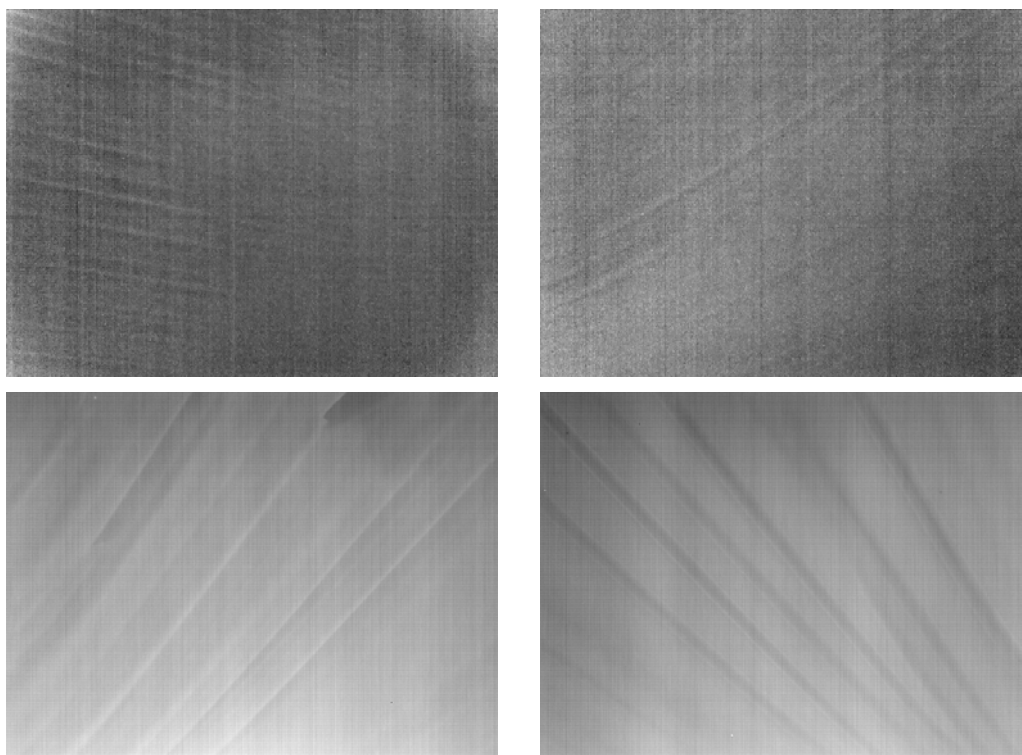


Figure 5.8: Some examples of images with periodic artifacts.

The natural way of distinguishing images with this artifact is to look at the frequency content of the image, utilizing the Fourier transform. This approach is described in section 5.2.1. A reasonable alternative is to use the gradient direction histogram feature, as described in section 5.2.2. A third alternative would be to use edge detection in combination with the Hough transform, but this approach was quickly rejected because the standard edge detection algorithms (for example Canny's algorithm) were unable to detect the soft edges in these images.

5.2.1 Fourier Coefficients

The most straightforward way of analyzing the frequency content of the input image is to look at the amplitude of the frequency components directly. Figure 5.9 gives an example of an image with a periodic artifact and the representation of that image in the Fourier domain (the image shows the amplitudes of the Fourier coefficients shifted so that the low frequency components are in the center). The intensity mean has been subtracted from the image before taking the Fourier transform to zero the otherwise strong DC component, which is uninteresting. If one looks closely at the Fourier transform of the image there are two strong peaks slightly off-center (the center pixel represents the DC level which is zero). As a comparison, Figure 5.9 also shows an image of random white noise with the same mean and variance as the first image. In image space these images look very similar, but in the Fourier domain they are well separated. Figure 5.9 also reveals the intuition behind the name white noise. The white noise shows uniform strength over the entire frequency spectrum, just as white light has a uniform spectrum over the visible frequencies of electromagnetic radiation.

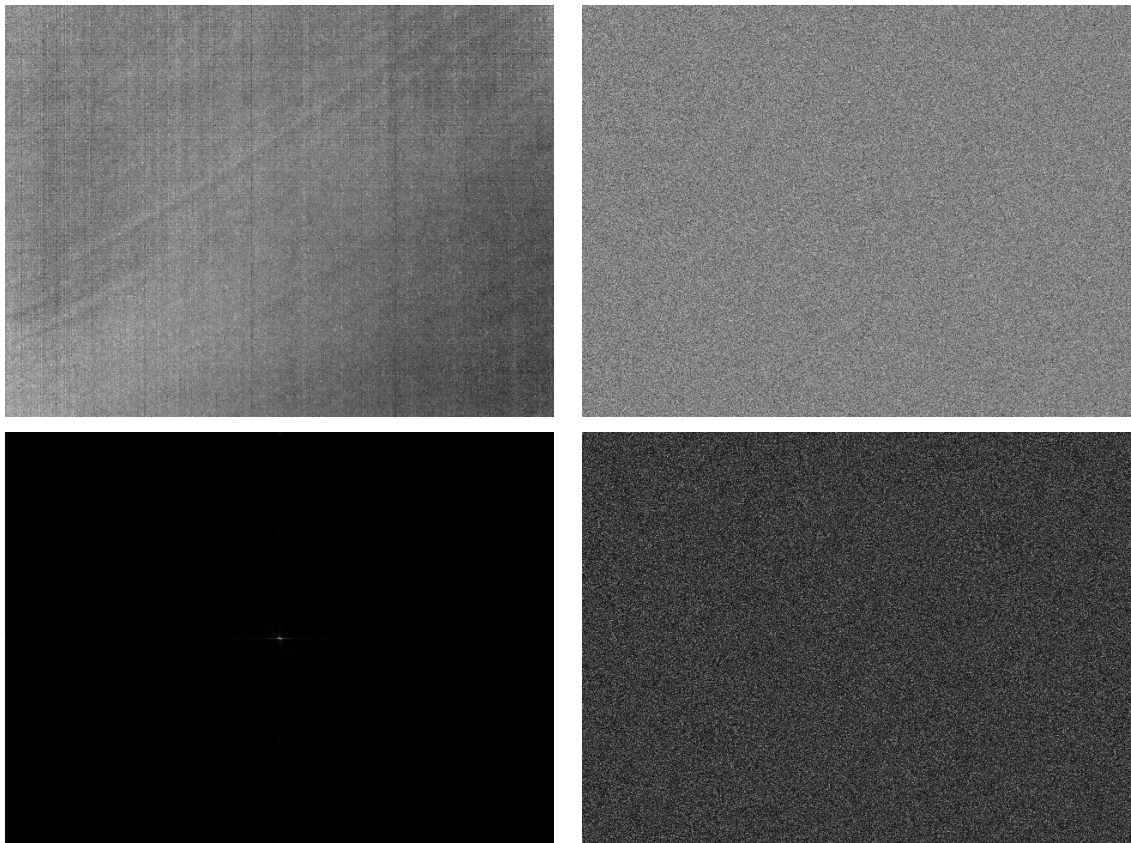


Figure 5.9: The Fourier transform of an image with periodic artifacts and an image with pure white noise with the same mean and variance as the first image.

Based on this observation, it seems like the Fourier transform would be a good feature for detection of periodic artifacts. Then the feature vector would be composed of the amplitude of the frequency components of the image. Figure 5.10 illustrates the procedure for computing the amplitude of the frequency components. First the input image is normalized to avoid scaling effects. Then the two-dimensional discrete Fourier transform is computed and the first fourth of the transform matrix, except the very first element, is used for calculating the amplitudes at different frequencies (the rest of the matrix is redundant and the first element represents the constant background which is of no interest in this application). Finally the amplitudes are collected in a feature column vector. If the input images have different sizes, that has to be taken into account also.

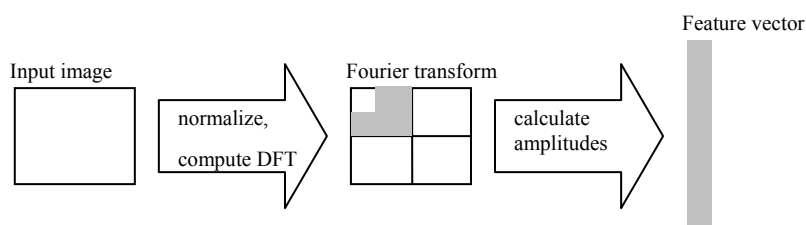


Figure 5.10: How to compute the Fourier transform feature.

One disadvantage of this feature is that it is not invariant to rotation of the input image. Figure 5.11 illustrates the problem. The two input images have periodic artifacts of similar spatial frequency (or wavenumber), but the “lines” in the second image have a different orientation than the “lines” in the first image. Intuitively the artifacts in these two images are similar, but the feature vectors are completely different if they are calculated according to Figure 5.10. If a classifier is exposed to the first image during training and is then asked to classify the second image based on that training, it will probably fail.

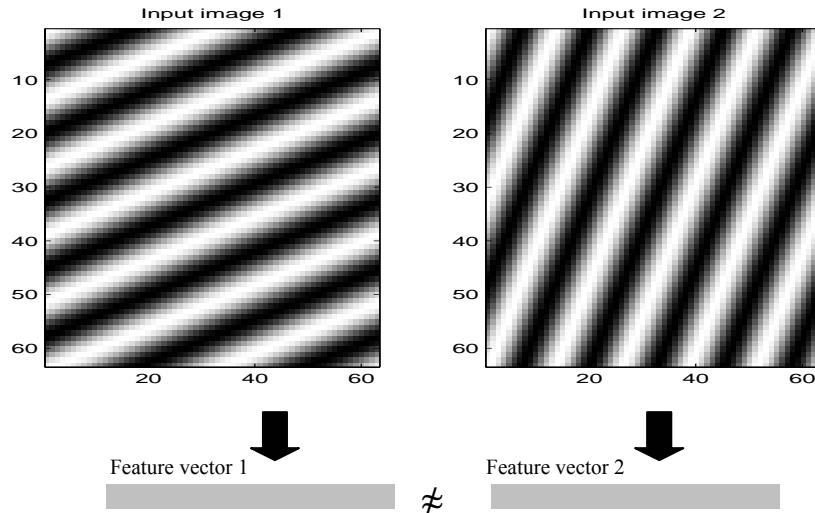


Figure 5.11: Illustration of the problem with not achieving invariance to rotation in the feature for detecting periodic artifacts. Two images with periodic artifacts of similar spatial frequency but different orientation will yield widely different feature vectors.

One way of solving the problem would be to rotate all images in the training batch through several different angles, thus creating an extended batch containing several copies of each image in the original batch. A probably better way is to modify the feature calculation to yield similar feature vectors for images with similar artifacts. Said differently, we would like two images with periodic artifacts with similar spatial frequency to yield similar feature vectors, even if they have different wavefront orientation. In order to construct a feature with this property, we have to take a closer look at the output of the two-dimensional Fourier transform.

The basis functions of the two-dimensional Fourier transform can be thought of as simple cosines, $\cos(u \cdot x + v \cdot y)$, where u and v are integer constants defining the spatial frequency (or wavenumber) and $x, y \in [0, 2\pi]$. Some examples of such cosines, for different values of the constants u and v , are shown in Figure 5.12. The original function or image can be thought of as the superposition of these basis functions. The element f_{ij} in the matrix of Fourier coefficients basically gives information of the strength of the $\cos((j-1) \cdot x + (i-1) \cdot y)$ component in the image. The wavenumber of this component is $\|(j-1, i-1)\|$.

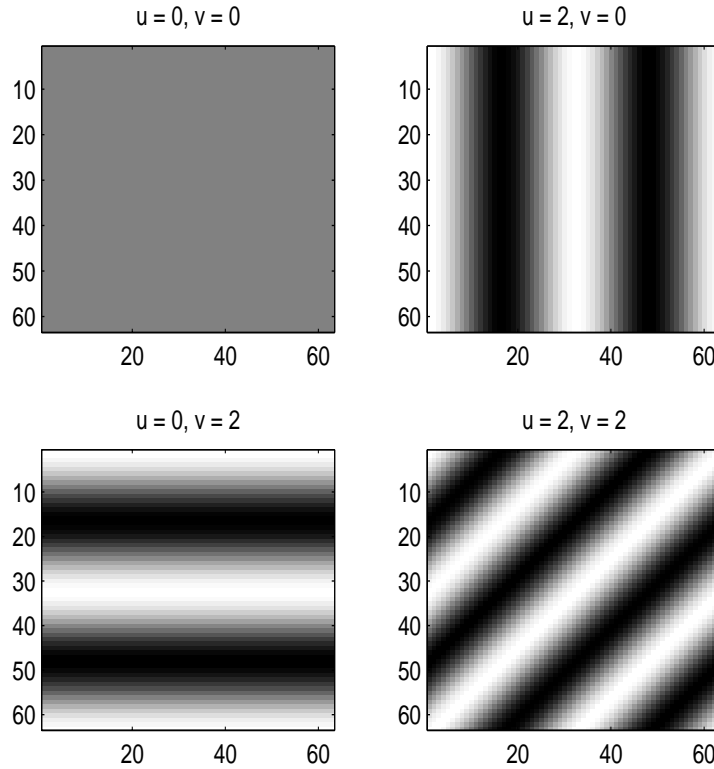


Figure 5.12: Illustrating the basis functions of the two-dimensional Fourier transform.

In order to achieve invariance to image rotation, we would like to aggregate the magnitude information for all coefficients that correspond to a single wavenumber. According to the above discussion, the Fourier coefficients corresponding to a single wavenumber are to be found at a constant distance from the “origin” of the matrix of Fourier coefficients. Thus the rotationally invariant Fourier feature amounts to summing the magnitudes of the Fourier components along circular arcs in the matrix of Fourier coefficients.

One question is if the sum of the coefficient magnitudes should be normalized by the number of terms in the sum. Basically the normalization factor will be the inverse of the circumference of a circle in the Fourier plane with radius corresponding to the wavenumber of the terms in the sum. Thus the normalization factor for a given element in the feature vector will be inversely proportional to the wavenumber of that element. The normalization factor will be the same for corresponding elements in different feature vectors. Thus this normalization simply corresponds to a scaling of each channel. Therefore, this normalization will not be applied. Instead each channel will be scaled to yield input values to the SVM in the range $[-1, 1]$. The algorithm for calculating the rotationally invariant Fourier feature can be summarized in the following steps:

1. Normalize the input image.
2. Compute the two-dimensional discrete Fourier transform of the normalized image.
3. Create a magnitude accumulator, which is an array with $nDist$ positions. $nDist$ is a parameter to the algorithm.
4. For each Fourier coefficient f_{ij} in the region of the Fourier transform matrix marked with gray in Figure X, compute the wavenumber of the corresponding component as $\|(j-1, i-1)\|$. If the computed wavenumber is less than or equal to $nDist$, add the magnitude of f_{ij} to the magnitude accumulator at position i , where i is the smallest number in $\{1, \dots, nDist\}$ that is bigger than or equal to the computed wavenumber.

5. Return the magnitude accumulator.

The final feature will be invariant to rotation and scaling of the image. It should also be robust to background structure. That is, it should not require test images to be acquired against a uniform radiation. Of course, if the background contains any parallel lines, that will lead to false detections.

5.2.2 Gradient Direction Histogram

The gradient direction histogram is simply a histogram of gradient angle calculated over the whole image. The motivation for this feature is that the gradient direction distribution of an image with no periodic artifacts should be reasonably uniform, while the direction distribution of an image with periodic artifacts should have a peak corresponding to the direction perpendicular to the wavefront of the periodic artifact.

Figure 5.13 illustrates how a gradient direction histogram with six bins divides the unit circle. The output of the algorithm is the number of gradients in each bin. The number of bins is a parameter to the algorithm. Another option is if the algorithm should include the gradient direction in every pixel or only the gradient directions corresponding to large gradient magnitudes, for example count only the gradients with 20 % largest magnitudes or count only the gradients with magnitudes above a certain threshold. The final output of the algorithm is the χ^2 distance from to the uniform distribution. The feature is thus single-valued and should produce small values for images with no periodic artifacts and large values for images with periodic artifacts.

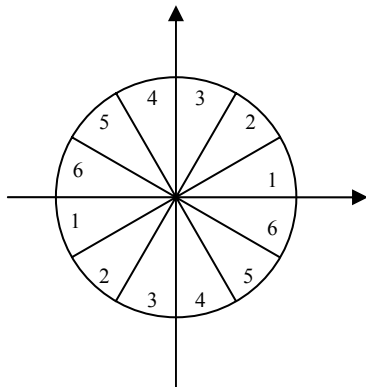


Figure 5.13: Illustration of a gradient direction histogram with six bins. The gradients can take directions in the interval $[-\pi, \pi]$. If the angle θ is in the interval $[-\pi, 0]$ it will be sorted into the same bin as $\theta + \pi$.

The feature should be invariant to rotation and scaling of the image. An exception is if a hard threshold is used to include only large gradients, then scaling of the image will change the feature value output.

5.3 Detection of “Shadows”

Figure 5.14 gives some examples of images with the shadow artifact. Images with this artifact seem to be divided into irregular regions with slightly different pixel intensity means. They bear visual resemblance to dark shadows on a slightly brighter background.

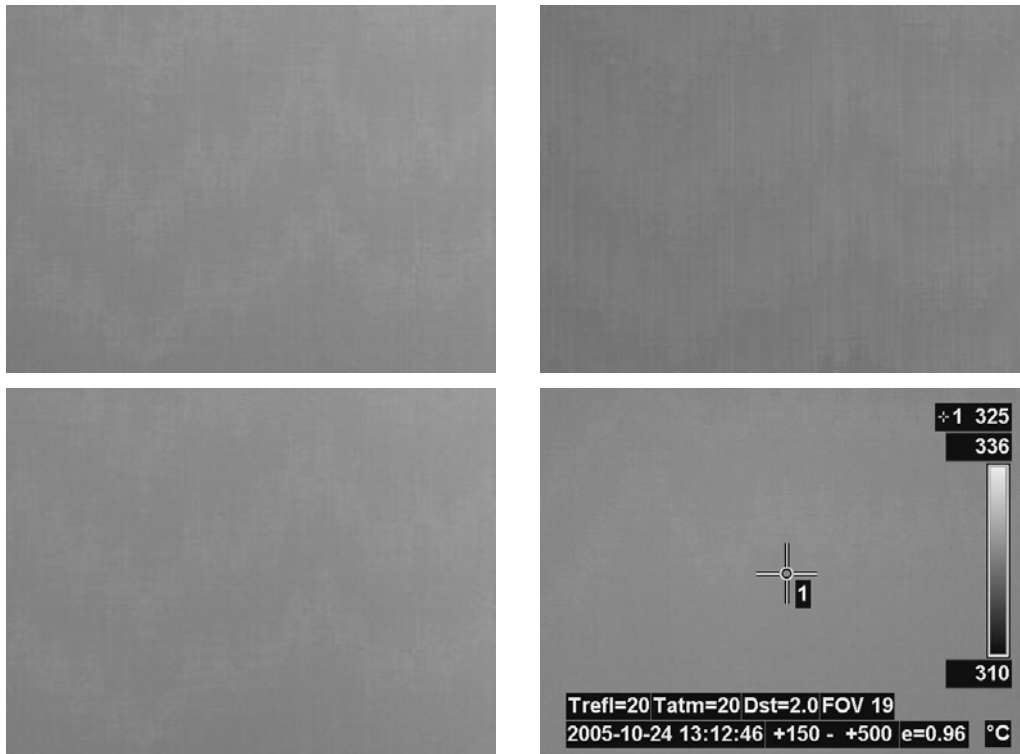


Figure 5.14: Some examples of images with shadow artifacts.

5.3.1 Gray-Level Co-Occurrence Matrix (GLCM)

The GLCM is a way of describing gray-scale texture. It seems reasonable that the shadow artifact can be described and distinguished as a texture using statistical measures.

Figure 5.15 illustrates the calculation of a GLCM. The gray values of an image are rounded to a small number (typically eight) of discrete values. Then, for each pixel, the algorithm looks at the next pixel to determine which of the elements in the GLCM to increase by one. When the calculation is done, the GLCM contains a description of the gray-level co-occurrence of horizontal pixel pairs.

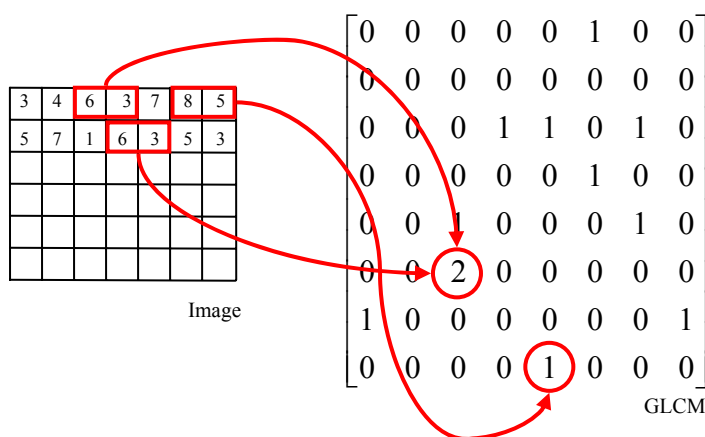


Figure 5.15: An illustration of the calculation of the GLCM. The gray level values of each pixel are first rounded to eight discrete values. The GLCM is then initialized to be an eight by eight matrix of zeros. For each horizontal pixel pair, the rounded gray level values determine which element in the GLCM to increase.

The GLCM matrix does not have to be calculated on horizontal pixel neighbors; any offset can be used to define the relative location of the pixels in a pair. We can thus define several different offsets and compute a GLCM for each offset to get a better description of the texture in the image. On each GLCM we can then calculate statistical measures describing the gray-level co-occurrence for the offset used to compute the GLCM. The statistical measures include measures of contrast, correlation, energy and homogeneity [38]. For the purposes of evaluation, we shall use these measures and fix the direction of offset according to Figure 5.16. The only degree of freedom left as a parameter to the feature calculation algorithm will be the number distances to include in the set of offsets. Figure 5.16 also shows the set of offsets when using five distances. Of course the definition of this feature can be changed. For example, there might be a better choice of offsets or statistical measures than that suggested above. So if this feature turns out to yield good performance, it will be valuable to do a more thorough evaluation.

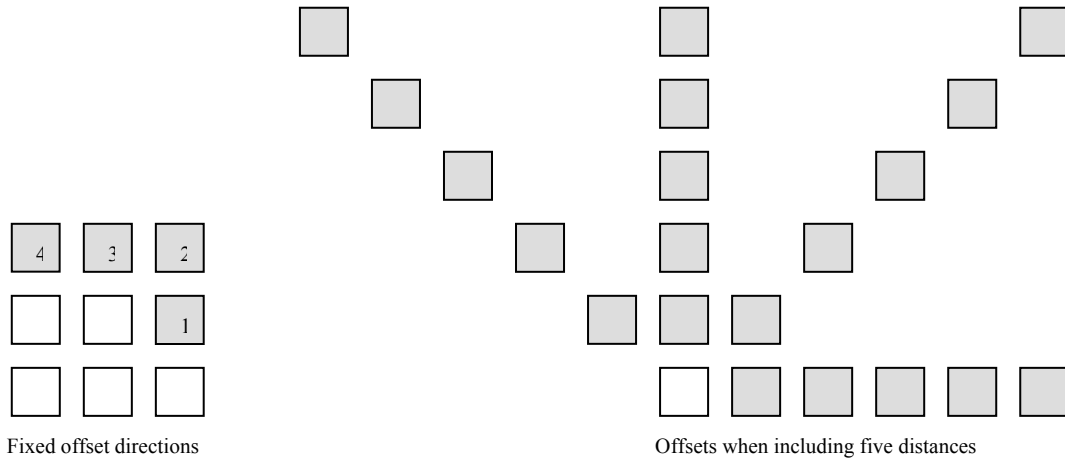


Figure 5.16: The fixed offset directions and the set of offsets when including five distances.

5.3.2 Point-Point Distance Distribution

The motivation for this feature is that there are subtle edges between the bright and dark regions in an image with shadows. These edges are not detected by a standard edge detector (such as Canny's algorithm), but a range filter should give a high response on those edges. Thus, the subtle edges in an image with shadows should give rise to clusters of local maxima in the response of a range filter. When a range filter is applied to an image without shadows, however, the local maxima in the range filter response should be more evenly distributed. This feature is designed to detect the presence of subtle lines by investigating the spatial distribution of range filter maxima.

The algorithm is summarized by the following steps:

1. The input image is band-pass filtered using a Difference of Gaussians (DoG) filter, thus removing the really high and low spatial frequency content. High spatial frequencies correspond to noise and low spatial frequencies correspond to the background.
2. A range filter is then applied to the frame resulting from step one. The response of a range filter for a given pixel is the local range (or the difference between the maximum and minimum) of pixel intensities in a neighborhood around that pixel. The range filter will thus accentuate weak edges in the image.
3. The result of step two is then normalized, so that all pixel values lie between zero and one.
4. The normalized image is then thresholded at 0.5, to create a black and white (binary) image. The white pixels then signify large local range and the black pixels signify small local range.

5. The idea is to then calculate the distribution of distances between white pixels in the binary image. For each pair of white pixels in the binary image, the distance is calculated using image coordinates. The calculated distance is then normalized by the maximum distance in the image (that is, the image diagonal), so that each distance is in the interval between zero and one. Finally, the normalized distance is sorted into a distance histogram. The number of bins to use for the distance histogram is a parameter of the algorithm. For the purposes of classification, this parameter is preferably set to a low value (good results are achieved using only two bins).

The main steps of the procedure are illustrated using a very clear shadow example in Figure 5.17. The distance histogram for images containing shadows will be dominated by short distances, while the distance histogram for images that do not contain shadows will be more uniform.

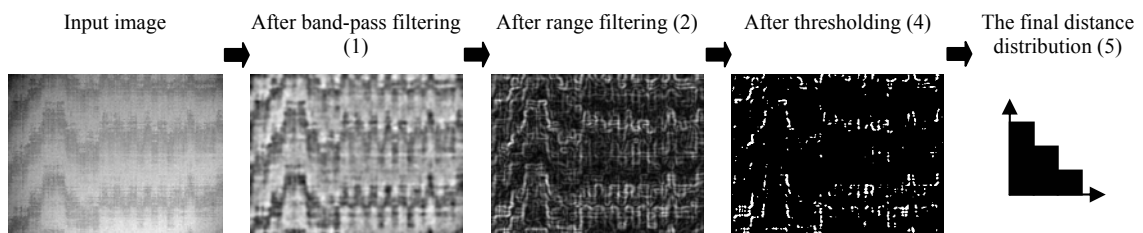


Figure 5.17: The main steps of the feature extraction method for shadow detection.

Since this feature detects the presence of subtle edges, it requires that the test images are acquired against a completely uniform radiation. The feature should be invariant to scaling and rotation.

5.3.3 Graininess

As previously stated, shadows make the image seem divided into irregular regions with slightly different intensity means. The previous feature was aimed at detecting the edges between these regions. This feature is aimed at detecting the regions themselves. If an image containing a shadow artifact is band-pass filtered, to remove high-frequency noise and background trends, those characteristic regions should stand out more clearly. If the image then is thresholded in a way as to minimize the variance of black and white pixels within regions in the output binary image, the regions should be even more apparent. The final binary image should thus be divided into big black and white regions. If, on the other hand, the same procedure is applied to an image with no shadows (i.e. an image containing only white noise), the binary image should be significantly grainier (divided into many small black and white regions). So if the graininess of the binary image can be measured, that measure could be used to separate images with shadows from images without shadows.

The question then is how to measure graininess. One idea is to use the total edge length in the binary image as a measure of its graininess and normalize that measure so that a perfect checkerboard would get a value of one and a constant image would get a value of zero. If the binary image I_{bw} has n_r rows and n_c columns and i_{ij} is the element at row i and column j , the graininess G can be calculated in the following way (which is effectively performed with three MATLAB commands):

$$\begin{aligned}
E_x &= \sum_{\substack{i \in \{1 \dots n_r\} \\ j \in \{2 \dots n_c\}}} |i_{i,j} - i_{i,j-1}| \\
E_y &= \sum_{\substack{i \in \{2 \dots n_r\} \\ j \in \{1 \dots n_c\}}} |i_{i,j} - i_{i-1,j}| \\
G &= \frac{E_x + E_y}{(n_c - 1) \cdot n_r + (n_r - 1) \cdot n_c}
\end{aligned} \tag{5.1}$$

Figure 5.18 illustrates the procedure of first band-pass filtering the image, then thresholding it and finally calculating the graininess measure. The example shows one input image with shadows and one without. As expected, the image with shadows yields a lower graininess measure from its binary image. The algorithm is given below. It takes two parameters, σ_h and σ_l , the sigma values for the Gaussian kernels of the high-pass filter and the low-pass filter respectively.

1. Apply a Gaussian filter with σ_h to the input image I . Call the response I_h .
2. Apply a Gaussian filter with σ_l to the input image I . Call the response I_l .
3. Subtract I_h from I_l to form $I_{bp} = I_h - I_l$.
4. Calculate a threshold t according to Otsu [35] and produce a binary image, I_{bw} , using that threshold.
5. Calculate the graininess G from I_{bw} according to equation 5.1. Return G .

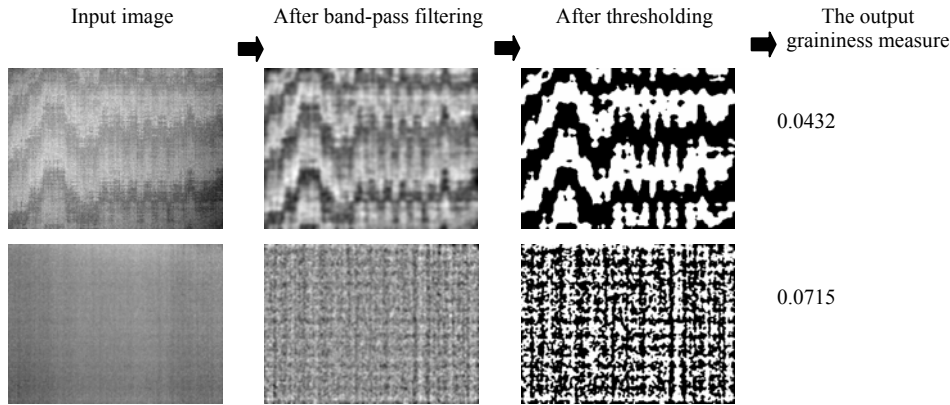


Figure 5.18: Two example images (one with and one without shadows) are followed through the process of calculating the final graininess measure.

This feature requires the images to be acquired against a uniform radiation. It should be invariant to scaling and rotation in steps of $\pi/2$.

5.4 Detection of Row Noise

Row noise is a temporal artifact, meaning that it is not readily noticeable in single images but are visible in image sequences. Then row noise is manifested by a row-oriented flickering. The presence of row noise is most easily detected against a background of uniform radiation. In this situation the image will show a constant temperature with superimposed (usually normally distributed) noise, so the pixel intensities will be normally distributed around a mean corresponding to the actual temperature. The flickering in sequences with row noise seems to be caused by single rows with deviant intensity distribution. Compared to the intensity distribution of the whole image, the distribution of a noise-causing row seems to have a deviant mean intensity and a smaller variance. The noise-causing rows seem to appear randomly in time and space, meaning for example that it will not be useful to do Fourier analysis on the mean

intensity of a single row over time and look for periodicities. Neither will it be useful to do Fourier analysis on the mean intensities of the rows in a single image. Furthermore, the number of noise-causing rows is small compared to the total number of rows, so it seems reasonable that a successful approach for row noise detection will look for extremes or outliers rather than look at means or medians. The sections below will propose several different features to use for row noise detection. The features are based on the above reasoning.

All features assume that the input image sequence is acquired against a constant and uniform radiation. The requirement that the radiation be uniform could probably in most cases be removed if the mean image (attained by taking the mean of each pixel over time) is subtracted from each frame in the sequence before feature calculation.

5.4.1 Variance of Row Intensity Means

The motivation for this feature is that for an image sequence without row noise the variance over time of the mean intensity for each row in the image matrix should be small.

Mathematically, if X_1, X_2, \dots, X_n are independent random variables with expected value m and variance σ^2 , then we have the following formula for the expected value and variance of the arithmetic mean:

$$E(\bar{X}) = m; V(\bar{X}) = \sigma^2 / n \quad [5.2]$$

In this case, n will be the number of columns in the image, m corresponds to the expected intensity in each pixel (which corresponds to the scene temperature) and σ^2 is the variance of the intensity in each pixel. Thus the mean intensity for each row should be centered on the expected value m with a small variance (given that there are reasonably many columns in the image). On the other hand, a noise-causing row will yield a deviant intensity mean, increasing the variance of intensity means over time for that row. So we would expect row noise to be manifested by large values for this feature.

The process of calculating this feature is illustrated in Figure 5.19. First the intensity mean is calculated for each row in each frame. Then the variance over time of the mean intensity for each row is computed and collected in a variance vector. The mean of these variances is computed. The important final step is to normalize the variance mean by the expected value given in equation 5.2, where σ^2 can be estimated as the intensity variance over the whole image sequence. There are no parameters involved in the calculation and the output is a single value, which is expected to be small for image sequences without row noise.

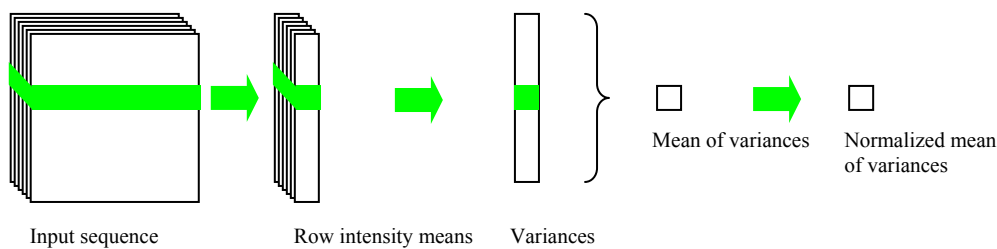


Figure 5.19: An illustration of the procedure for calculating the variance of row intensity means feature for detection of row noise.

As an optional step, the same calculation can be performed for the columns of the image sequence and the hypothesis would be that the column value should equal the row value for sequences without row noise. The difference between the two values could be used as a feature value.

5.4.2 Row Intensity Variance Minimum

The motivation behind the row intensity variance minimum feature is that the variance of pixel intensities within a noise-causing row seems to be less than the variance of the pixel intensities in the image sequence as a whole. Thus looking at the minimum over time of the intensity variance for each row should yield lower values for the image sequences without row noise than for the sequences with row noise.

Except from the measures used, the process of calculating this feature is similar to the previous feature. The algorithm is illustrated in Figure 5.20. The pixel intensities of the input sequence are first normalized to the range $[0,1]$. Then the variance of the intensities in each row is calculated and the minimum variance for each row over time is computed and stored in a vector. The mean of these variance minima is calculated and normalized by the pixel intensity variance estimated from the whole image sequence. This normalized mean of variance minima forms the final feature value. There are no parameters involved in the computation and the image sequences without row noise are expected to yield higher feature values than the sequences with row noise.

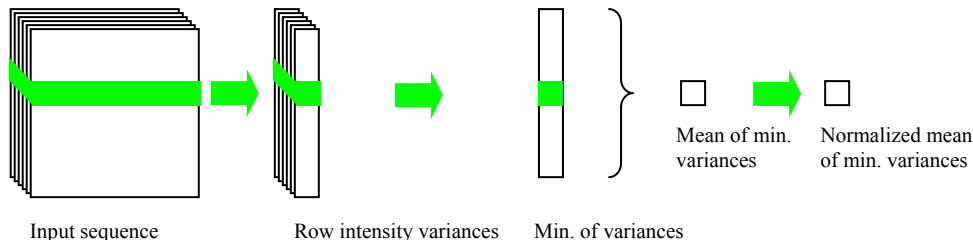


Figure 5.20: An illustration of the procedure for calculating the row intensity variance minimum feature for detection of row noise.

As with the previous feature, we have the option to perform the same calculation for the columns of the image sequence and use the difference between row value and column value as the final feature value instead of using the row value directly.

5.4.3 Number of Variance and Mean Outliers

We have established that the noise-causing rows have deviant mean intensities and intensity variances. Based on that observation it seems reasonable to propose a feature that counts the number of rows with deviant mean and variance in the image sequence. Figure 5.21 shows the normalized means and variances (calculated according to step two below) for each row in an image sequence with row noise. Surprisingly, the mean intensities of all rows are very close to the mean intensity over the whole image sequence. The normalized mean is thus always very close to one. So the distance d in the algorithm below will closely equal the absolute value of the normalized variance minus one.

The feature calculation algorithm will take one threshold p to separate outliers from inliers. The feature value will be calculated in the following way:

1. Calculate the mean and variance of the intensity values in the whole input image sequence.
2. Calculate the mean and variance for each row and normalize by the mean and variance calculated in step one. Call the normalized mean and variance m_n and v_n respectively.
3. Calculate the distance d from the normalized mean and variance for each row to the expected values as $d = \sqrt{(m_n - 1)^2 + (v_n - 1)^2}$.
4. Count the number of rows with distance d greater than the parameter p .
5. Return the count divided by the total number of rows.

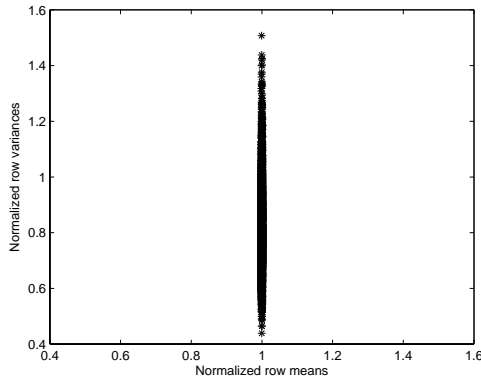


Figure 5.21: The normalized mean and variance for each row in an image sequence with row noise.

5.4.4 Model Conformity

The model conformity feature is an attempt to further exploit that we have a model for the intensity value in each pixel. The intensity value x of a pixel can be regarded as an observation of a random variable X , drawn from a normal distribution with expected value m and standard deviation σ . We assume that the expected value and the standard deviation is the same for each pixel in the image and use all the intensity values in the whole input sequence to estimate the mean intensity m and standard deviation σ of the random variable X representing a model pixel. Our basic hypothesis is then that the intensity value of each pixel in a row of the image matrix is accurately described as a random variable from a normal distribution with the calculated parameters m and σ . For each row in each frame of the sequence we can then use the observed intensities in that row to try to refute our basic hypothesis. If the observed values deviate enough from the hypothesized model, we will reject it in favor of the hypothesis that the row is a noise-causing row. If many rows cause the basic hypothesis to be refuted, we can suspect that there is a large amount of row noise present in the image sequence.

In statistics the basic hypothesis is called a null hypothesis and is assumed to be true before an experiment is conducted. If the data generated by the experiment does not agree with the null hypothesis, the null hypothesis is rejected in favor of an alternative hypothesis. The rejection of the null hypothesis is done based on some test quantity, which is calculated from the experimental data. The rejection is done so that there is a low probability of rejecting a true null hypothesis. The use of null hypotheses has been much debated, but seems to be applicable here. A more complete description can be found in [32].

The strategy for refuting the null hypothesis will be to make new estimates of mean and standard deviation based only on the intensity values in the current row. If these values deviate too much from the m and σ estimated from the whole image sequence, the null hypothesis that the pixels in the row are normally distributed random variables with expected value m and standard deviation σ ($X \in N(m, \sigma)$) is rejected in favor of the alternate hypothesis that the pixels in the row have another distribution and the row is a noise-causing row. The next step will be to determine numerical thresholds for refuting the null hypothesis.

Firstly we determine a confidence interval for the expected value m based in the intensities $x = (x_1, x_2, \dots, x_n)$ in the current row:

$$I_m = (\bar{x} - t_{\alpha/2}(f) \cdot d, \bar{x} + t_{\alpha/2}(f) \cdot d), d = s / \sqrt{n}, f = n - 1 \quad [5.3]$$

In the above equation $t(f)$ is the t-distribution (or Student distribution) with f degrees of freedom and $t_{\alpha/2}(f)$ is the $\alpha/2$ quantile of the same distribution. The variable n is the number of observations, which is the number of pixels in the current row in our case, and s is the sample

standard deviation. The confidence level α is a parameter giving the risk that the true expected value lies outside the confidence interval, so naturally the interval increases if α is decreased.

In a similar way we determine a confidence interval for the standard deviation σ based on the intensities in the current row:

$$I_\sigma = (k_1 s, k_2 s) \quad [5.4]$$

$$k_1 = \sqrt{f / \chi_{\alpha/2}^2(f)}; k_2 = \sqrt{f / \chi_{1-\alpha/2}^2(f)}, f = n - 1$$

In the above equation $\chi^2(f)$ is the χ^2 -distribution (chi-square distribution) with f degrees of freedom and $\chi_{\alpha/2}^2(f)$ is the $\alpha/2$ quantile of the same distribution. The variable n is again the number of pixels in the current row and s is the sample standard deviation. The confidence level is α .

Assuming that m and σ are the true expected value and standard deviation of the pixels in the current row, we then have:

$$P(m \in I_m) = 1 - \alpha \quad [5.5]$$

$$P(\sigma \in I_\sigma) = 1 - \alpha$$

If we dare assume that the probabilities of the two events are independent of each other, the probability of both events occurring is:

$$P(m \in I_m \cap \sigma \in I_\sigma) = (1 - \alpha)(1 - \alpha) \quad [5.6]$$

It seems reasonable to state that if either $m \notin I_m$ or $\sigma \notin I_\sigma$ (which is the complementary event to both being in their respective confidence intervals) the null hypothesis shall be rejected. The probability (which we shall call β) of rejecting a true null hypothesis will then be:

$$\beta = 1 - (1 - \alpha)(1 - \alpha) \quad [5.7]$$

Solving equation 5.7 for α yields:

$$\alpha = 1 - \sqrt{1 - \beta} \quad [5.8]$$

The probability β of rejecting a true null hypothesis will be a parameter of the feature calculation algorithm. The steps of final algorithm are given below:

1. Given an image sequence S , estimate the true expected value m and standard deviation σ for a model pixel.
2. Given the β parameter, calculate α according to equation 5.8.
3. Initiate a row noise counter c to zero.
4. For each row in the image sequence S , calculate the confidence intervals I_m and I_σ using the pixel intensities in the current row according to equations 5.3 and 5.4. If $m \notin I_m$ or $\sigma \notin I_\sigma$ increase the row noise counter by one.
5. Normalize the row noise counter by the total number of rows in the sequence S (divide c by the number of frames in S times the number of rows in each image).
6. Return the normalized row noise counter.

5.4.5 Maximum χ^2 Histogram Distance

Another way of measuring model conformity is to use the χ^2 distance of the normalized intensity histogram for each row to the normalized intensity distribution of the whole image sequence. The χ^2 distance between two histograms H_1 and H_2 is defined as:

$$D_{\chi^2}(H_1, H_2) = \sum_{i=1}^n \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \quad [5.9]$$

The χ^2 distance of the normalized intensity histogram of a row from the normalized intensity histogram of the whole image sequence will give an indication of how similar the row is to the model histogram. It can be expected that the noise-causing rows will yield a relatively large χ^2 distance.

The process of calculating this feature is simple. The normalized intensity histogram for the image sequence as a whole and the histograms for each row separately are calculated. Then, for each row, the χ^2 distance is computed. Finally the maximum observed χ^2 distance observed is used as the feature value. It is not obvious that the maximum is the best choice of aggregate function. It was chosen because it looked promising in preliminary testing and it is consistent with the observations stated in the beginning of section 5.4.

A variation of this feature would be to try other distance measures than the χ^2 distance. The earth mover's distance [25][33] is an example of another distance measure that could be investigated for this application.

5.4.6 Number of χ^2 Distance Outliers

Instead of just looking at the maximum χ^2 distance the idea is to count the number of rows with significantly larger χ^2 distance than average.

The algorithm for calculating this feature will take one parameter, $p \in [0, 2]$, to be used as a threshold for separating outliers and inliers. The process of calculating the feature value is given below:

1. Compute the normalized intensity histogram for the whole image sequence and also for each row in the sequence.
2. Compute the χ^2 distance for each row.
3. Count the number of distances that have a value greater than the parameter p . Return the count divided by the total number of rows.

Like the previous feature, this feature can of course also be varied by using another distance measure. Another variation is to use the intensity histogram of each frame separately as a norm, to compensate for global intensity variations. In this case the feature calculation algorithm would get the following appearance:

1. Initiate an outlier counter c to zero.
2. For each frame in the image sequence, compute the intensity histogram for the whole frame and also for each row in the frame. Compute the χ^2 distance for each row in the frame. Count the number of distances greater than the parameter p . Add that count to c .
3. Return c divided by the total number of rows in the sequence.

Yet another variation is to look at the distribution of χ^2 distances, instead of just looking at the outliers (the tail of the distribution). In that case, the feature will become vector valued and a support vector machine (or some other classifier) is needed for classification.

5.4.7 Number of Small Entropies

Since we expect the pixel intensities to be normally distributed with some common expected value m , we can expect that the binary value attained when thresholding the intensity at m should have entropy one. However, if m is estimated as the mean intensity over the whole sequence, thresholding the intensities in a noise-causing row at m should yield a binary dataset with entropy closer to zero. The feature value will be the portion of rows with entropy smaller

than some threshold $p \in [0,1]$, which is a parameter to the algorithm. It is expected that the image sequences with row noise will yield bigger values for this feature than the image sequences without row noise.

6 Results

This chapter presents the main results in a concise format.

The subsections below give an overview of the results for each artifact type. Results relevant for the comparison of different features will be presented, but the more detailed results have been left out of this chapter and can be found in Appendix A. For example, to see how the classification performance of a feature varies as a function of parameter values the reader is referred to Appendix A. In this section only the peak performance is given, along with the corresponding parameter values. Looking at the classification performance as a function of the parameter values will give an insight on how sensitive the feature is to changes in the parameter settings.

As a general remark to the results, it can be said that all evaluations have been done on relatively small batches of test images and should not be interpreted as statistical truths, but is a good indication of what can be expected from automatic image quality control.

The experiments were conducted in MATLAB and for tests involving support vector machines, the free toolbox OSU SVM was used [40].

6.1 Spot Detection

This section contains the results for spot detection. The objective of this section is to rank the features proposed for spot detection according to their classification performance and determine what can be gained by combining several features. The most promising features have been given some extra attention. The experiments were conducted as follows:

1. A batch of 47 manually classified examples of spots and 47 examples of non-spots was compiled. Figure 6.1 shows thumbnail images of the 47 basic positive spot examples and Figure 6.2 shows the negative examples.
2. The information value of each feature was evaluated by performing classification based on each feature separately. The parameters involved in the computation of each feature were varied in search of optimal parameter values. For the purposes of these experiments, reasonable (but not optimal) settings were used for the classifier (SVM). The classification performance was determined through leave-one-out cross-validation.
3. The classification performance for different feature combinations was evaluated. Each feature was calculated using the optimal parameter values as determined in the previous step.
4. Using the feature combination that was found to give the best performance in the previous step, a search for optimal values for the SVM parameters was conducted.
5. Finally the spot detector, using the features and parameters found in the previous steps, was applied to a real image containing spots to get an understanding of the actual performance that can be expected from this spot detector.

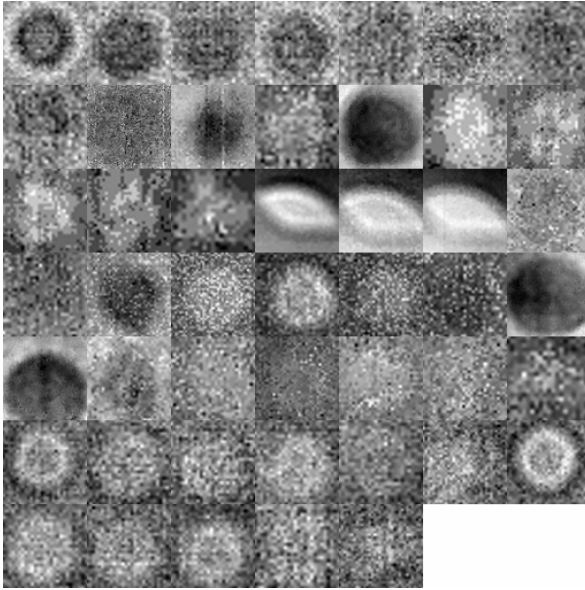


Figure 6.1: This image shows the set of spot images, after being normalized, contrast enhanced and downsampled.

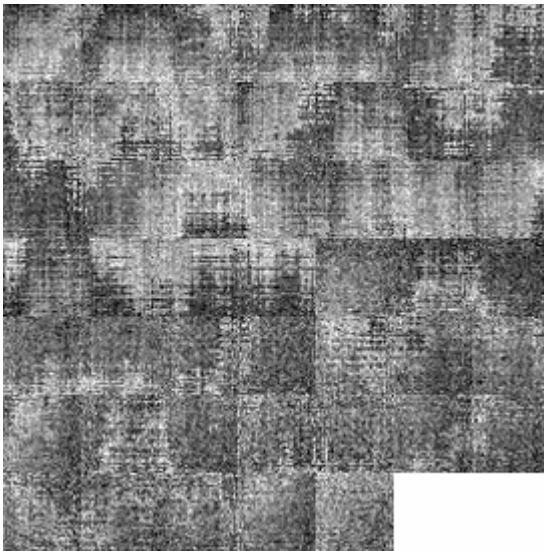


Figure 6.2: This image shows the set of non-spot images, after being normalized, contrast enhanced and downsampled.

For the vector-valued features, the classification performance was measured using leave-one-out cross-validation. The first step of this procedure is to calculate the feature vectors for each example image in the batch. Then all but one feature vectors are used to train the support vector machine. The withheld feature vector is used to test the classification performance; it is presented to the trained support vector machine, which outputs a classification. If the output class (which will be “spot” or “non-spot”) matches the known correct class, one is added to a performance counter. The procedure is repeated for each example in the batch and the final performance counter value is divided by the total number of examples to get the estimated classification performance.

For the single valued features (which is just the center distance and intensity correlation feature for spot detection), simple thresholding was used for classification. The classification performance was determined as the percent of correct classifications when using an optimal threshold. An optimal threshold can be found by sorting the examples according to their feature

value and then exhaustively evaluating a set of candidate thresholds midway between adjacent examples from different classes [36].

The computations of most features depend on some parameters. For each feature, the classification performance was optimized with respect to the value of these parameters. Table 1 presents the results for the spot detection features. For each feature, the best performance achieved is given along with the optimal parameter values.

Table 1: The best classification performance and corresponding parameter settings for each feature.

Feature	Best performance	Parameter settings
Radon profiles (1)	0.9043	$c_d = 0.001$ $n_{bins} = 16$
Edge pixel histogram (2)	0.8298	$\sigma = 4$ $n_{bins} = 10$
Spin image (3)	0.9681	$n_{dist} = 4$ $n_{int} = 3$
Gradient direction histogram (4)	0.9255	$n_{bins} = 6$
Resampled original (5)	0.9894	$n_{pixels} = 8$
Center distance and intensity correlation (6)	0.9572	

Can one feature complement another, so that combining them improves performance over using each feature separately? To answer this question the classification performance was evaluated for pairs of features instead of just one feature at a time. The results are presented in Table 2. When combining features like this it is usually necessary to introduce some weighting or normalizing procedure to match the ranges of the different features. But since each feature in this case is naturally normalized to the range $[0,1]$, further normalizing or weighting was considered unnecessary.

Table 2: The classification performance when using pairs of features as input data to the SVM. The feature calculation was done using the parameter settings from Table 1.

		Second feature				
		1	2	3	4	5
First feature	2	0.8617				
	3	0.9574	0.9681			
	4	0.9362	0.9362	0.9468		
	5	0.9894	0.9894	0.9681	0.9787	
	6	0.9149	0.8298	0.9574	0.9255	0.9894

The results in Table 2 indicate that given a set of features little is gained by using more than one feature compared to just using the best feature in the set.

Two features stand out in the comparison: the downsampled original because it achieved the best performance and the center distance and intensity correlation because it achieved almost as good performance while using only a single value. These features deserve further investigation. The downsampled original was treated first.

The next step is to search for optimal settings for the SVM parameters. The parameters that have to be considered are the cost of constraint violation, the SVM kernel type and the kernel

parameters. Table 3 shows the performance for the three different kernel types: linear, polynomial and radial basis function (RBF).

Table 3: The classification performance for different kernel types

Kernel type	Performance
Linear	0.9681
Polynomial	0.9894
RBF	0.9574

According to Table 3, the polynomial kernel was the best choice. The polynomial kernel is defined by the following expression:

$$K(u, v) = (\gamma \cdot \langle u, v \rangle + c)^d \quad [6.1]$$

The parameters to determine, in addition to the cost, C , of constraint violation, is therefore γ , c and d from equation 6.1. In doing this a new performance measure was also introduced. Previously classification performance, as measured by cross-validation over the test set, was the only performance measure but at this stage it seemed reasonable to also look at the number of support vectors, which gives an indication of the complexity of the decision boundary found by the SVM. Few support vectors means simple decision boundary, which can be argued to imply better capability of generalization. Thus, the desire is to maximize classification performance while minimizing the number of support vectors. Default values for each parameter were set according to Table 4 and the parameters were then varied separately. The results are shown in Figure 6.3.

Table 4: Default parameter values

Parameter	Default value
C	1
γ	1
c	0
d	3

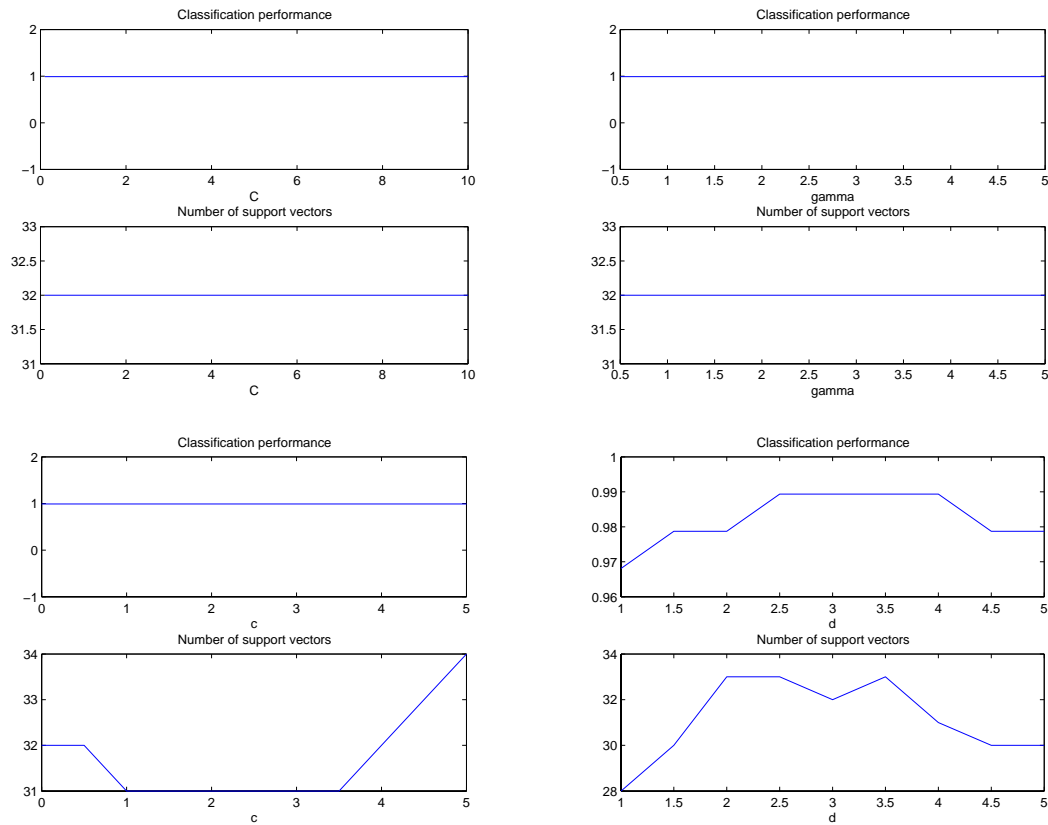


Figure 6.3: Classification performance and number of support vectors plotted against parameter value for each of the parameters C , γ , c and d .

Figure 6.3 shows that it is impossible to improve the classification performance beyond 0.9894 by varying the SVM parameters. However, by changing the coefficient c from zero to one the number of support vectors decreases from 32 to 31. The SVM parameters were henceforth fixed to the following values: $C=1$, $\gamma=1$, $c=1$ and $d=3$.

To evaluate the practical utility of the spot detector, it was used to scan an image containing spots. Since the downsampled original is not invariant to rotation and since the spot images are not rotationally symmetric, the set of training images was extended by rotating the images in the original set. The images were rotated through the angles 90, 180 and 270 degrees to create a new training set four times as large as the original set. The SVM was then trained on this extended set and used to scan a test image. The test image is shown in Figure 6.4 and the result after scanning that image is shown in Figure 6.5. The test image is 240 by 320 pixels and the scales inspected were 25 by 25 to 30 by 30 pixel patches. The total number of inspected patches was thus approximately $240 \times 320 \times 5 = 384000$. 12130 of those image patches were classified as spots.

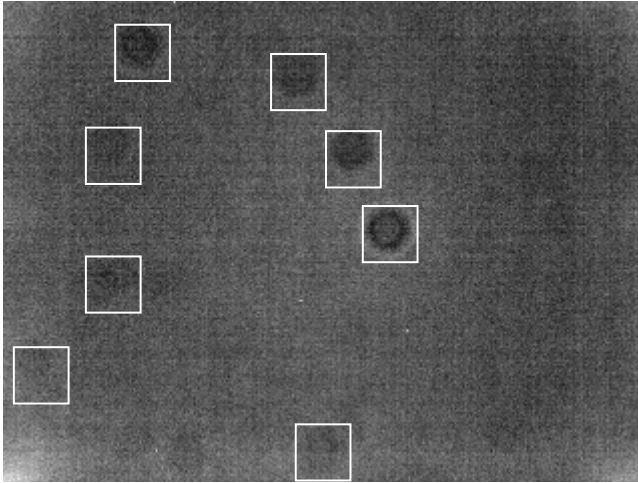


Figure 6.4: Test image that was scanned for spots to evaluate the practical utility of the spot detector. The spots in this image have been manually marked with white squares to illustrate the “ground truth”.

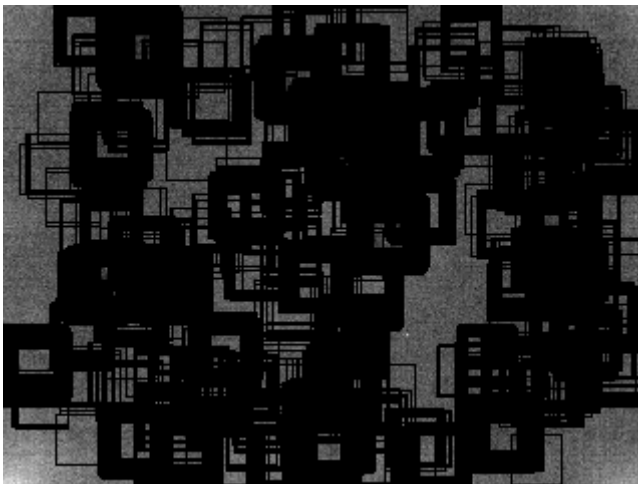


Figure 6.5: The result of scanning the test image in Figure 6.4 over space and scale. Each image patch that was classified as a spot by the SVM has been marked by a black square in the image.

Figure 6.5 shows much false positive detection. In an attempt to reduce the number of false positives, the detection was run on some images that did not contain any spots and all image patches that were detected as spots were recycled as negative training examples. This is a way of tuning the decision surface that has been shown essential to avoid false positives in face detection [31]. The detection results after this boosting step is shown in Figure 6.6 and indeed the number of false positives has been dramatically reduced. The strong spots have been detected a large number of times and form big clusters in the three-dimensional search space (the dimensions are the image coordinates and scale). There are still some false positives; further tuning of the training set (additional rounds of boosting with carefully chosen images) would be one way to refine the results even more. Since it is the training set that defines for the SVM what is to be considered a spot, it is very important to use a representative training set. The expressiveness of the feature used for spot detection further increases the necessity of having a good training set.

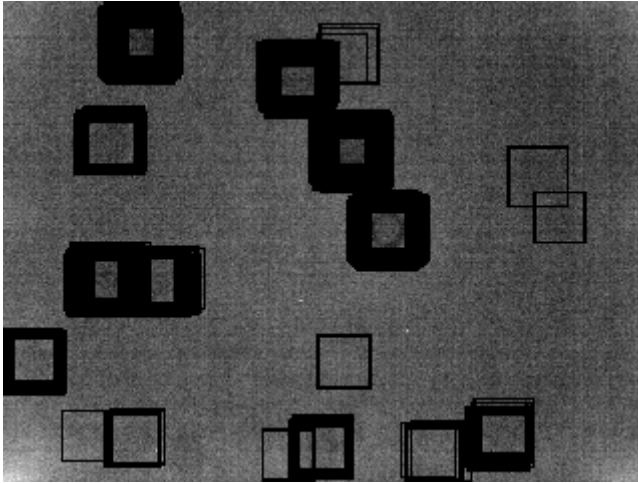


Figure 6.6: The result of scanning the test image after extending the set of negative training examples with false positive detections from some images with no spots.

As mentioned good performance was also achieved from the simple, single-valued center distance and intensity correlation feature. Figure 6.7 shows the feature values for each example in the batch. As expected, the example images without spots, marked by (green) stars, generally yield correlation values close to zero, while the images with spots, marked by (red) rings, generally yield correlation values of greater absolute value (dark spots on bright backgrounds yield positive correlation and bright spots on dark background yield negative correlation).

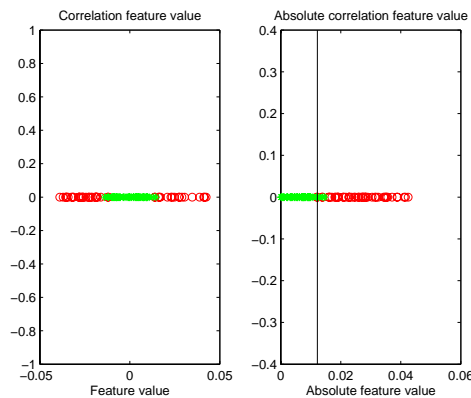
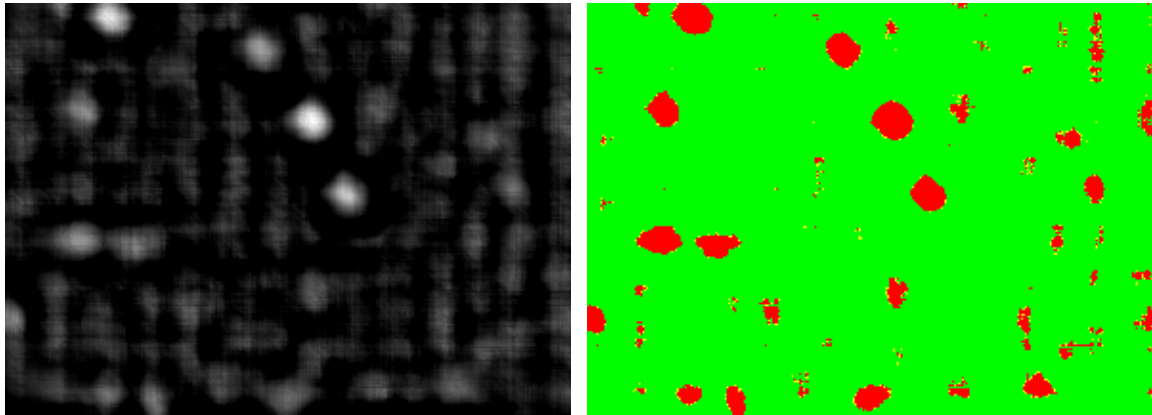


Figure 6.7: Covariance between the normalized distance from the center of the image patch and the normalized intensity. The values for the positive examples with spots are marked as (red) rings and the values for the negative examples without spots are marked as (green) stars. The second plot shows the absolute feature values and the black line marks the optimal threshold, which is located at 0.0122, for classification. When using this threshold, the classification performance is 0.9574.

The simple correlation feature can be used as a filter (which will be called correlation filter even though that term is commonly used with different meaning). For each pixel, the filter outputs the absolute correlation feature value. The size of the filter kernel can be specified as a range, in which case the filter outputs the maximum correlation over that range for each pixel. The response of the correlation filter when applied to the image in Figure 6.4 is shown in Figure 6.8. Figure 6.8 also shows the outcome of thresholding the correlation filter response. If the response was above a high threshold, a certain spot detection was marked in red (dark). If the response was below a low threshold, a certain non-spot detection was marked in green (bright). If the response fell between the thresholds, a possible spot detection was marked in yellow.



a) The correlation filter response.

b) The thresholded filter response.

Figure 6.8: a) The response of the correlation filter using a filter kernel size range of 25 to 30 pixels squared. b) The filter response after thresholding. The low threshold (for marking the response as green) was 0.0121 and the high threshold (for marking the response as red) was 0.0125. Responses between the low and high thresholds were marked as yellow.

Comparing the thresholded filter response in Figure 6.8 with the SVM spot detection in Figure 6.6 shows that the two methods generally agree on spot detections, with the difference that the correlation filter response has captured the small scale blotchiness of the original image.

6.2 Detection of Periodic Artifacts

The features described in section 5.2 for detection of periodic artifacts were evaluated and optimal parameters for feature calculation and SVM classification were sought through the following steps:

1. A batch of 55 manually classified examples of images with periodic artifacts and 40 examples of images without artifacts was compiled.
2. The classification performance of the two features was evaluated by doing classification based on each feature separately. The parameters involved in the computation of each feature were varied in search of optimal parameter values.
3. The classification performance for a combination of features was evaluated.
4. Using the feature setting that was found to give the best performance in the previous step, a search for optimal values for the SVM parameters was conducted.

For each feature, optimal parameter choices for feature computation were determined and the best classification performance achieved for each feature is given in Table 5, along with the optimal parameter values. The first feature, termed “full Fourier feature”, was illustrated in Figure 5.10 and the result for this feature was included as a reference for comparison to the rotationally invariant Fourier feature.

The components of the feature vectors of the Fourier features differ widely in range, so it might be wise to scale the components of the feature vector to prevent components with greater numeric ranges to dominate components with smaller numeric ranges. Another advantage of scaling in this case is that if the components of the feature vectors are too large, there might be numerical problems when calculating the inner products in the kernel. To determine the value of scaling, the feature vectors were scaled so that each component had the range $[-1,1]$ before being passed to the leave-one-out procedure. This improved the classification performance slightly for the Fourier feature.

The gradient direction histogram feature is single valued and the classification was in this case done by simple thresholding (if the feature value is above the threshold, the image will be considered to contain periodic artifacts). The optimal threshold was found to be 0.0087, using the settings from Table 5.

Table 5: Summary of the results for detection of periodic artifacts

Feature	Best performance	Parameter settings
Full Fourier feature	0.9579	
Full Fourier feature w. scaling	0.9579	
Fourier feature (1)	0.9579	$n_{comp} = 150$
Fourier feature w. scaling (1)	0.9895	$n_{comp} = 150$
Gradient direction histogram (2)	0.9474	$n_{bins} = 9$

The performance when combining the Fourier feature with the gradient direction histogram feature was also evaluated. The Fourier transform feature was calculated using 150 frequency components and the gradient histogram feature value was calculated using 9 bins. The data was scaled before being passed to the leave-one-out cross-validation procedure. Table 6 gives the performance when using both features for classification. As for spot detection, combining features did not improve the classification performance beyond what was achieved with the best feature alone.

Table 6: Classification performance when using both features

0.9895

Finally, the search for optimal SVM parameters was conducted. Table 7 gives the classification performance for different choices of kernel when using only the Fourier feature and when using it in combination with the gradient direction histogram feature. The linear kernel was the best choice in both cases and using both features seemed to give more stable performance over different kernel choices. The linear kernel has no parameters.

Table 7: The classification performance for different kernel types

Kernel type	Performance (using only the Fourier feature)	Performance (using both features)
Linear	0.9895	0.9895
Polynomial	0.9684	0.9789
RBF	0.8632	0.8632
Sigmoid	0.2316	0.5158

6.3 Detection of “Shadows”

In this section, the evaluation of the features for shadow detection is described. The calculation of these features was detailed in section 5.3. The first feature is the GLCM statistics, which is a vector-valued feature. The second and third features are single-valued and designed for classification by simple thresholding. The features were evaluated through the following steps:

1. A batch of 25 manually classified examples of images with shadow artifacts and 48 examples of images without artifacts was compiled.
2. The classification performance of the two features was evaluated by doing classification based on each feature separately. The parameters involved in the computation of each feature were varied in search of optimal parameter values.

3. The classification performance when combining the GLCM statistics and the point to point distance distribution feature was evaluated.

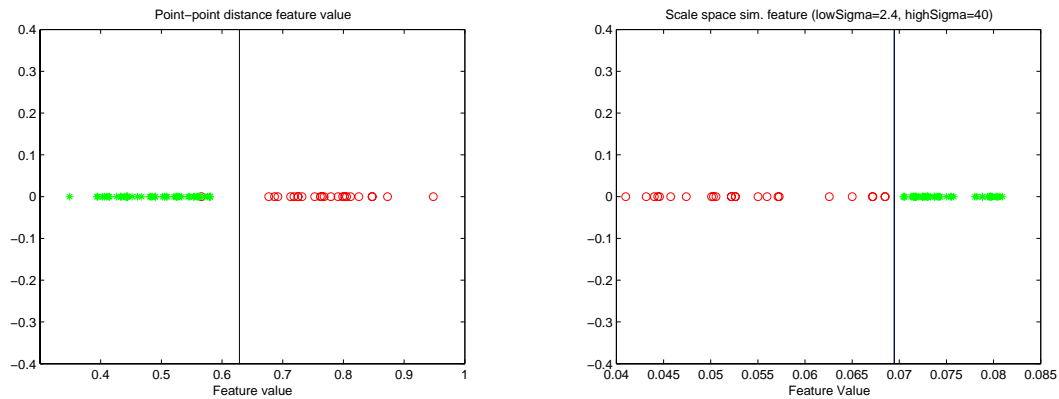
The classification results are summarized in Table 8. The calculation of the GLCM feature is dependent on a single parameter corresponding to the number of distances to include in the GLCM calculation. The elements of the GLCM feature vectors were scaled to the range $[-1,1]$ before being passed to the SVM leave-one-out cross-validation. Classification results are shown with and without the application of a band-pass filter (using Gaussian kernels with $\sigma_{lp}=2$ for the low-pass filter and $\sigma_{hp}=10$ for the high-pass filter) before computing the feature vector.

The point to point distance distribution feature and the graininess feature are both single-valued. Classification performance in these cases refers to what would be achieved with an optimal threshold. Both features achieved optimal classification performance for several parameter settings. Therefore a second performance measure was introduced. The separation between classes (which can be negative in the case of overlap), was used in a secondary optimization step.

Table 8: Summary of the results for detection of shadow artifacts

Feature	Best performance	Parameter settings
GLCM	0.9863	$n_{dist} = 12-16$
GLCM (with band-pass filtering)	0.9726	$n_{dist} = 5-6$
Point to point distance distribution	0.9863	$\sigma_{lp} = 0.5$ $\sigma_{hp} = 9$
Graininess	1	$\sigma_{lp} = 2.4$ $\sigma_{hp} = 40$

The single-valued features proved to yield surprisingly good performance. Figure 6.9 shows the point to point distance distribution and graininess feature values for each of the images in the example batch. The point to point distance distribution has one misclassification, but otherwise gives better separation between the two classes. The reader is referred to Appendix A for more detailed results, including a cautioning example showing the graininess feature applied to a validation batch of images without shadows.



a) Feature values for the point to point distance distribution feature.

b) Feature values for the graininess feature.

Figure 6.9: a) Plot showing the point to point distance feature values for images with shadows, marked with (red) rings, and images without shadows, marked with (green) stars. The threshold used for classification was 0.6286 and is marked with a black line in the plot. There is one false negative classification (that is one image with shadows was classified as having no shadows). b) Graininess feature values for images with shadows, marked with (red) rings, and images without shadows, marked with (green) stars. The threshold used for classification was 0.0695 and is marked with a black line in the plot.

Finally, the classification performance when combining the GLCM feature with the point to point distance distribution feature was evaluated. Feature calculation was done using the best parameter settings according to the previous results. Table 9 shows the classification performance in this case. Each feature was scaled to the range $[-1,1]$ before being passed to the leave-one-out performance evaluation. The classification performance did not improve when combining features.

Table 9: Classification performance when using both GLCM and point to point distribution features for shadow detection

0.9863

6.4 Detection of Row noise

This section contains the evaluation of the features for detection of row noise. The algorithms for calculating the feature values were described in section 5.4. Most features are single valued and designed to give large values if row noise is present and small values if not, or vice versa. Thus if a single feature is used for classification, a simple threshold should be enough to separate the image sequences with row noise from the image sequences without row noise. If several features are combined, a support vector machine (or some other classifier) is needed. The steps used for the acquisition of row noise detection results are given below:

1. A batch of 30 manually classified image sequences with row noise and 16 examples of image sequences without row noise was compiled.
2. The classification performance of each feature separately was evaluated and for those features with parameters or optional variations, the optimal settings were sought.

Table 10 summarizes the classification performance achieved for each proposed feature. The variance of row intensity means feature and the row intensity variance minimum feature have optional steps described at the end of sections 5.4.1 and 5.4.2. The results for these variants are

also given in the table. The χ^2 distance distribution refers to the variant of the χ^2 distance outliers feature. Instead of thresholding on the number of χ^2 distance outliers, the complete histogram of χ^2 distances is passed to a support vector machine for classification.

The best performance was achieved with the χ^2 distance outliers feature with separate normalization. The classification performance and optimal threshold are plotted versus the parameter p in Figure 6.10, which also shows the feature values for each image sequence in the batch. The intensity histograms used for the χ^2 distance calculations had six bins. The first plot shows the classification performance and optimal threshold plotted against the value of the parameter p . The upper curve and lower dashed curve represent the result after separate normalization of each frame. Then the best performance, 0.9348, was achieved for $p=0.25$. The optimal threshold was then 0.0949. The best performance without separate normalization of each frame was 0.8044, achieved for $p=1.15$ and with optimal threshold 4.1667e-005. The second plot shows the feature values (calculated for $p=0.25$ and $p=1.15$ respectively) for the image sequences with row noise, marked as (red) rings, and for the image sequences without row noise, marked as (green) stars. The optimal thresholds have been marked by black vertical lines.

Table 10: Summary of the results for row noise detection

Feature	Best performance	Parameter settings
Variance of row intensity means	0.4783	
Variance of row intensity means variant	0.5870	
Row intensity variance minimum	0.7174	
Row intensity variance minimum variant	0.7826	
Variance and mean outliers	0.7391	$p = 0.5$
Variance outliers	0.8043	$p = 0.9$
Model conformity	0.7609	$\beta = 0.01$
Maximum χ^2 distance	0.8261	$n_{bins} = 6$
χ^2 distance outliers	0.8044	$p = 1.15$
χ^2 distance outliers with separate normalization	0.9348	$p = 0.25$
χ^2 distance distribution	0.6522	$n_{bins} = 9$
χ^2 distance distribution with separate normalization	0.8261	$n_{bins} = 9$
Minimum entropy	0.5870	

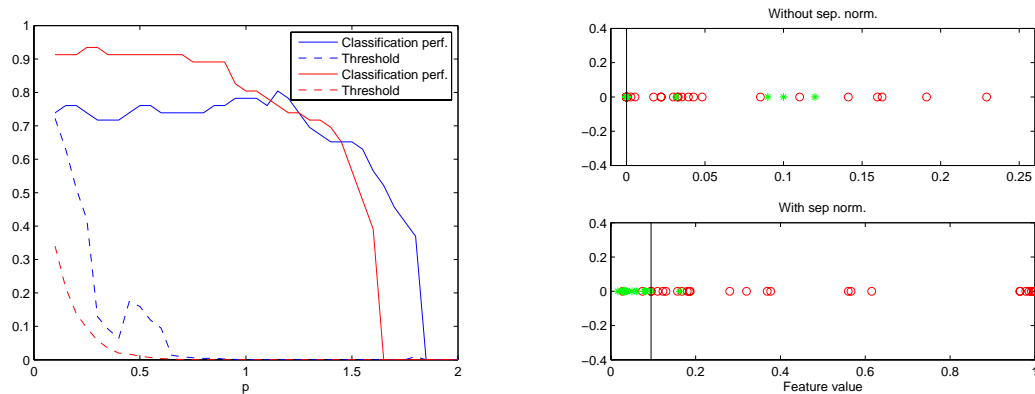


Figure 6.10: The classification performance of the number of χ^2 distance outliers feature.

7 Discussion

This chapter contains a discussion of the results from the previous chapter. How do we interpret, use and build upon these findings?

Section 7.1 is an attempt to deduce the characteristics of each artifact from the classification performance of the different features. In section 7.2, the implementation of the live system is discussed. One of the topics in this section is which features should be chosen. There are several aspects to consider beyond the classification performance on the test batch. In section 7.3 some areas suitable for future improvements are identified. Finally, in section 7.4, an alternative architecture for artifact detection is described. The alternative architecture was rejected at an early stage because it showed inferior performance.

7.1 Results Analysis

The goal in this section is to gain a better understanding for each artifact. Each feature was suggested with some model of the artifact in mind. The classification performance achieved for each feature is thus a measure of how well that model agreed with the real characteristics of the feature. Getting a better model of the artifacts will aid future development of image quality measures.

The visual characteristic of spots is that a small, near circular region of the image is either slightly darker or slightly brighter than its surroundings. The question was how to describe an image patch so that patches with spots would stand out. Descriptions using the gradient field, the intensity profiles, the downsampled original and statistical measures were proposed. The spin image feature did (not surprisingly) reveal a strong correlation between intensity and distance from the center for image patches with spots. Using this correlation explicitly as a single-valued feature turned out to distinguish spots from other image patches well. The correlation measure can thus be concluded to correspond well to the visual appearance and is therefore a good model of spots. While most other features for spot detection also yielded fairly good performance, the correspondence between feature value and visual appearance is not as close.

The periodic artifacts visually look like claw marks on the image. One idea was to describe them as parallel lines. While these lines were too weak to be detected by an edge detector (in which case for example the Hough transform could have been used to see if there were any straight lines in the binary output from the edge detector), they did show up in the gradient direction histogram as peaks the directions perpendicular to the “wave front” of the periodic artifact. So the images with periodic artifacts can be described as images with one dominant direction in the gradient direction histogram. This model has the advantage of being single valued and correspond well to visual appearance. Using the Fourier transform to extract the frequency content was also a successful approach. In this case the images with periodic artifacts were characterized by relatively large amplitudes in a certain range of spatial frequencies. The advantage of this model was a slightly better classification performance. Maybe the Fourier feature could be made single-valued, for example by specifying a fixed frequency range to measure and then aggregating the amplitudes within that range.

The shadow artifacts are visually characterized by large image regions that are slightly darker than the rest of the image. The first suggestion for describing these artifacts was to use the gray-level co-occurrence matrix. This description yielded quite good classification performance. For an image with constant intensity, one of the elements on the diagonal of the GLCM (corresponding to the single gray-level in the image co-occurring with itself) will receive a large count and the rest of the elements will be zero. This corresponds to the situation in an image without shadows; the weight of the GLCM will be centered on one diagonal element. If an

image consists of two gray-levels, two elements on the GLCM diagonal (corresponding to the two gray-levels in the image co-occurring with themselves) will receive large counts and two off-diagonal elements (corresponding to the two gray-levels co-occurring with each other at the boundary between the regions) will receive small counts. This corresponds to the situation in images with shadows; the weight of the GLCM will be centered around two diagonal elements with two small local maxima forming the third and fourth corners in a rectangle together with the global maxima on the diagonal. Maybe this insight can be turned into a single valued feature with close correspondence to the visual appearance of shadows.

The single valued features tried for shadow detection also showed very good performance. However, they both had one large disadvantage. The point to point distribution feature, aimed at detecting the weak edges between regions, was computationally demanding. The graininess feature, aimed at detecting the regions themselves, turned out to be sensitive to the choice of parameter values for the band-pass filter. The values that yielded complete separation of the test batch did not perform as well on a validation batch of images without shadows (see Appendix A).

The last artifact that was studied was named row noise because it is manifested as a row-oriented flickering in image sequences. The features suggested for detection of row noise were based on the idea that the cause of row noise was the appearance of rows with deviant statistical properties. These noise causing rows seemed to appear randomly in both time and space, so frequency analysis was not tried. Most features instead aimed at finding and often counting the statistically deviant rows. Most of these features were unsuccessful at achieving good classification performance (apart from χ^2 distance outliers feature, which yielded decent separation). It is important to note that row noise is by far the most difficult artifact to classify manually by visual inspection, so inconsistent classifications in the test batch might be one reason for the discouraging results. But maybe the number of statistically deviant rows does not correspond well to the visual manifestation of row noise. Since human classifiers need to watch image sequences to detect row noise, maybe describing the timed behavior of image sequences with and without row noise is essential for row noise detection. For example, a strong correlation would be expected between the intensity values in a pixel at time t and $t+1$. Flickering might be better characterized as unnaturally rapid change (or discontinuity) in pixel intensities. This would not capture the row-oriented nature of the flickering, but is still an interesting avenue of future investigation (note that the variance of row means over time was tried and did not perform well).

Another interesting aspect to note about the row noise detection results is that treating each frame separately improved the detection performance significantly. So it is better to measure the statistical deviance of a row relative to a single frame compared to measuring the deviance relative to the whole sequence. This suggests that there are significant differences between the frames in a sequence (even though they show the exact same scene) and that it is important to consider the reference frame used for deviance measures.

7.2 Implementation

This section contains a discussion of various issues relating to the implementation of a live system for artifact detection.

7.2.1 Choice of Features

The most important design choice is which features to use. A simple approach to choosing features is of course to just pick the features that yielded the best classification performance in the test runs. There are, however, several other aspects to consider.

Firstly, having features that correspond closely to the visual appearance of the artifacts is desirable. So it is good to have a feature space that not only separates the artifacts from the rest

of the images but also makes sense. A good example of a feature that makes sense is the correlation feature for spot detection. If a spot is misclassified, the reason is that it is very weak and almost invisible. A feature that does not make as much sense is the Fourier feature for periodic artifacts. In this case it is more difficult to explain any misclassifications, since the representation is more obscure. In this case, images with seemingly strong periodic artifacts might be misclassified and having no artifacts. Is it worth the price of a few extra misclassifications to ensure that the misclassifications that do happen are at least visually weak artifacts?

Secondly, complexity also needs to be considered. Simple features seem better and more reliable than complex features involving much pre-processing. It is of course essential that the features are robust to changing conditions and it is also good to have a clear understanding of when (under what conditions) a feature will work.

Finally, number of operations needed to compute the feature is also important. This is especially true if the artifact detection is to be implemented on the camera (which will be discussed in section 7.3.1).

Given these three criteria, which features should be used for artifact detection? For spots it seems like a good solution would be to use the correlation filter as a pre-processing step and then concentrate on the image patch corresponding to the maximum filter response. If that patch has a correlation that is much smaller than what would be expected for a spot, the image will pass the spot test. On the other hand, if the correlation is really strong, the image will be considered to contain at least one spot. If there is uncertainty, the patch can be used to calculate the downsampled original feature and passed to a SVM or the image could be marked for manual inspection.

To detect periodic artifacts, one recommendation would be to use the Fourier feature, but not pass an image that yields a too high value for the gradient direction histogram feature. This makes use of the good classification performance achieved by the Fourier feature at the same time as it decreases the probability of misclassifying an image with visually strong artifacts.

Using a similar argument, it seems reasonable to suggest that the GLCM feature be used for shadow detection with the point to point distance distribution as a backup. If the classifier, using the GLCM feature as input, suggests that the image should be classified as not having shadows and the point to point distance distribution outputs a value lower than the threshold, the image should still be passed on for manual inspection.

The only feature that gave reasonable performance for row noise detection was the χ^2 -distance outliers feature, so in lack of something better this feature has to be used alone.

In general, it seems like a good rule of thumb to use a simple feature with good classification performance on the test batch as primary feature and then give a feature with good visual correspondence the right of veto. This will ensure good overall classification performance while reducing the risk of approving an image with visually strong artifacts. The downside is of course an increased rate of false detections, which increases the need for (possibly unnecessary) manual inspection.

7.2.2 Test Image Acquisition

There are two main issues to consider when defining the test setup. Firstly, the test images should be acquired under those conditions that are most likely to produce as clear artifacts as possible. It is also important to consider the invariance of the features that will be used in the detection step. For example, some features require the test image to be taken against a background of uniform radiation. If there are objects in the background, that might cause false detections.

7.2.3 Selection of Training Examples

It is the set of training examples that defines what is considered an artifact and what is not. The training set is a visual and objective way of defining that distinction and has to be chosen carefully.

If the training set is used to train a SVM, it is important to include boundary cases in the training batch, since the boundary cases are the most likely candidates to be used by the SVM as support vectors. For example, if a SVM is used for spot detection with the downsampled original feature, the boundary cases would be the really weak spots that are just strong enough to cause the camera to fail the quality control. It is also important to include the negative boundary cases, which in this case are image patches that are almost but not quite spots.

The training examples should also be chosen with regard to the test setup and the features that will be used by the detector. If, for example, the test setup is designed to detect shadows against a uniform radiation and the GLCM feature (together with a SVM classifier) is used for detection, the training data also has to consist of GLCM feature vectors computed from images acquired against a uniform radiation. The training data has to be representative for what will be encountered by the detector during operation.

It will probably be necessary to replace the test set and retrain the system when switching from one camera model to another. There will be different image quality requirements and maybe even different artifacts, which will require the system to be retrained and retuned. Thus there will be a running-in period after production-start for each new model.

7.2.4 Obtaining a Confidence Measure

When a detector classifies an image as having or not having a certain artifact it would be interesting to also get a measure of how “sure” the detector is that the output classification is correct. If a simple threshold is used to classify images based on a meaningful single valued feature, the actual feature value can be directly used as a confidence measure. For example, if the correlation feature is used to determine if an image patch contains a spot, a feature value that is significantly bigger than the threshold would be a certain spot detection, while a feature value close to the threshold would be an uncertain classification. If the feature value is close to the threshold (either on the spot side or on the non-spot side), the image can be passed to a manual classifier.

Is it possible to obtain a confidence measure for the classifications done by a support vector machine on feature vectors? The SVM actually outputs a so called decision value along with each classification. The decision value is a measure of the distance from the feature vector to the decision boundary in feature space, so in that sense it is analogous to the single valued case. However, the images that get mapped to points in feature space close to the boundary might not be the same image that a manual operator would pick out as visual boundary cases. Figure 7.1 shows a few examples of images with periodic artifacts, ordered according to the decision value. The decision values were obtained by first training the SVM on all images in the example batch except from the test image. The SVM was then used to classify the test image and the corresponding decision value was recorded. The first image was misclassified as not having periodic artifacts. It can be seen that the correspondence between visual appearance and decision value is not perfect. If the decision value is used as a confidence measure, that should be done with caution.

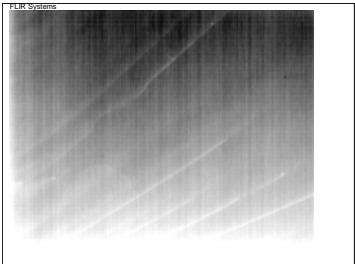
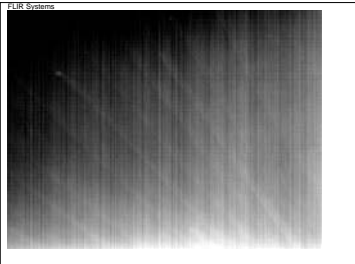
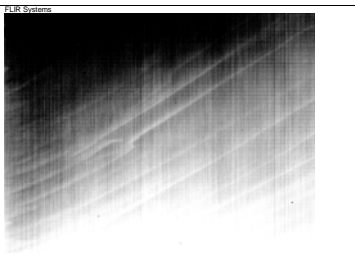
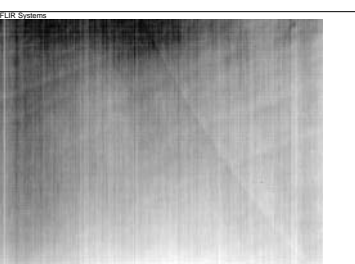
Image	Decision Value
	-1.43
	0.95
	14.51
	15.98

Figure 7.1: Images for the example batch for periodic artifacts with their corresponding decision values.

7.3 Future Improvements

In this section some areas suitable for (or even requiring) future development will be identified.

7.3.1 Artifact Detection on the Camera

It would be desirable if the detection of artifacts could be done by the camera itself. Then the camera could notify the user or operator that it suffers from low image quality. This would provide a logistically simple solution for image quality control. The bounding factors are memory and processing power, which are limited in the camera. In this section we shall discuss the prerequisites for performing feature extraction and classification on the camera.

In order to assess the storage requirements we ask what needs to be stored. Firstly, there are of course the algorithms for feature computation and the algorithm that administers the image

handling and the test sequence (i.e. the system core according to Figure 4.1). These algorithms are compact and do not influence the storage requirements significantly. Then there is the classifier, which might be either a simple threshold or a support vector machine. How much memory does the support vector machine require? The implementation of the SVM is compact and should not be a problem to store, so how about the support vectors? The support vectors, which define the decision surface of the trained SVM and are used by the SVM implementation to classify new images, are just the feature vectors of the boundary examples from the training set. Usually there are only a few support vectors (many support vectors would be a sign of overfitting, which should be avoided) and since the feature extraction is generally an attempt to extract salient information, the support vectors will require less space for storage than the original images. Since the camera definitely can store data equivalent in size to a few images, we conclude that the storage requirements are definitely satisfied. Memory might, however, still be an issue since some of the feature computation algorithms store partial results in RAM.

The required processing power also needs more investigation. Some representative computations have been timed and the results are given in Table 11. These computations were performed in MATLAB on an Intel Pentium 4 processor with one gigabyte of RAM. MATLABs `tic` and `toc` functions (which work like a stopwatch) were used for recording the time.

Not surprisingly, searching an image for a local artifact (spots in this case) takes a long time. The compute time is of course dependent on the size of the search space, which is defined by the user. The search space is three-dimensional (x , y and $scale$). The sizes of the x and y dimensions are given by the number of columns and rows in the image, but the size of the $scale$ dimension is defined by the user. The image in the example computation had 320 columns and 240 rows and six different scales were searched. Searching fewer scales would decrease computation time linearly. Another measure to decrease the computation time for spot detection is to use the correlation filter, which has also shown good classification performance and is computed much quicker. A third option would be to use the correlation filter as a pre-processing step. Only the regions marked as possible spots need be inspected by the SVM with the downsampled original feature.

It should be noted that the computation time for the correlation filter response, which was noted to be 769 seconds, can easily be decreased dramatically. The first possibility is to implement the algorithm in a more efficient language, like C or Fortran. The second possibility is to decrease the number of correlation computations by only computing the response for every second pixel in both x and y directions and scale. This would decrease the computation time by a factor of 1/8. Since it was noted that the spots were detected over a wide range of positions and scales, this should not affect performance and it might even be possible to make the response calculations sparser yet.

For detection of periodic artifacts we note that the Fourier feature is computed quickly, owing to the fast algorithms for computing the finite Fourier transform. Furthermore, since the periodic artifacts are of global nature this feature will only be computed once for each image and no scanning is necessary. The image used as input to the algorithm was 480 by 640 pixels.

Shadow detection using the point to point distance feature is generally slower. The reason is that computing the distance distribution requires looking at the distance between each pair of white pixels in the binary frame. There is a large variation in computing time depending on which image is input to the algorithm. The computing time grows rapidly with the number of white pixels in the binary frame. The images used as input to the algorithm were 480 by 640 pixels.

The χ^2 -distance outliers feature for row noise detection is reasonably fast to compute. The computation time for the other row noise features should not be significantly different from this value, since all features basically look at each pixel in each frame to compute the statistical measures used to find deviant rows. The image sequence used for this test had 100 frames with 120 by 120 pixels.

Finally, we see that making one SVM classification is very fast. It basically corresponds to computing the inner product between the input feature vector and each of the support vectors. In this case there were relatively many support vectors (28 support vectors on the positive side and 36 on the negative side), so the classification might be even faster in the case of fewer support vectors.

Table 11: Computation time for some representative operations

Operation	Time (s)
Computing the downsampled original feature for spot detection	0.92
Computing the correlation feature for spot detection	0.15
Computing the response of the correlation filter for spot detection	769
Scanning an image for spots using the downsampled original feature and an SVM classifier	3585 (1 hour)
Computing the Fourier feature for detection of periodic artifacts	0.31
Computing the point to point distance feature for shadow detection	1-100
Computing the χ^2 -distance outliers feature for row noise detection	21
Making one SVM classification using a feature vector with 64 elements and X support vectors	0.00021

It can be concluded that global artifacts can be efficiently detected on the camera, but faster algorithms are needed before detection of local artifacts can be done well on the camera. There are several possibilities for speeding up detection of local artifacts. The most trivial way is to decrease the size of the search space (for example by searching fewer scales or using a bigger step in the x and y directions). Calculating approximate feature values instead of exact ones is another possibility. Trying to develop global features that detect local artifacts might also be possible, and that will be discussed next.

7.3.2 Global Detection of Local Artifacts

In the previous section it was noted that detection of local artifacts is very slow because of the need to scan the image. Is it possible to detect local artifacts, like spots, without scanning the image? One approach using wavelet decomposition was tried but without success. Another possibility would be to use a feature similar to the point to point distance feature for shadow detection. This feature was designed to detect weak connected edges and might work to detect the edges surrounding spots.

If implementation in the camera is intended, global detection of local artifacts seems worthwhile to investigate.

7.3.3 Online Learning

The basic system that has been proposed is trained on a carefully chosen training batch before being used as a classifier. After that initial round of training, it does not update its basic data for decision-making. This is called offline learning. The alternative, called online learning, is to let the classifier learn continuously. In this approach, the classifier would refine its decision boundary incrementally during operation. Whenever an image is passed on for manual classification (i.e. because the classifier is uncertain of its classification), the result of that classification is fed back to the classifier as a new training example. This might also be a good mode of operation during the transition from the production of one camera model to another. Initially, a human operator would supervise the output of the classifier and correct any misclassifications made by the classifier. After some experience, there will be more and more agreement between the operator and the classifier. Then the classifier might be allowed to operate with increasing independence.

7.3.4 Letting the System Request Images

The problem statement in this thesis is image classification for artifact detection. This gives the detection system a passive role in the quality control procedure. The system is simply fed a series of images and is then expected to output a (hopefully empty) list of detected artifacts. It might be beneficial to give the system a more active role. Instead of being a passive classifier, the system could be given the role of an active investigator.

The final goal is to determine if a camera is defective or not. Formulated this way, the ultimate solution to the problem seems to be a decision support system to aid the operator. A decision support system could be implemented on the system core level according to Figure 4.1. The quality measurements made on the image and the output of the detectors would act as the sensors of this system. The decision support system could then request which test images should be acquired, what measurements should be made on those images and which detectors should be applied to the images in order to decide if the camera is defective or not. The best decision could be presented to the operator along with a confidence measure and the reason for any doubt (if the decision is to pass the camera, reason for doubt might be images that show weak signs of artifacts). The operator decides if the camera should pass the quality control and the decision is fed back to the decision support system as training data. In this system, the operator would have the passive role and the decision support system would be the active investigator.

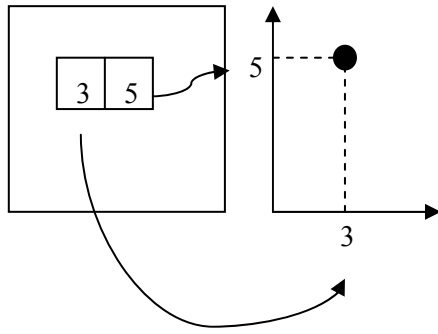
The advantage of having this system structure is that it will still be a human operator that makes the final decision. However, the operator will not have to conduct the investigative part of the quality control procedure, but will be presented with a suggested decision along with a few key images (or other measures). This would dramatically decrease the time the operator needs to spend on each camera, it would improve the work environment of the operator (who would no longer have to sit in a room full of radiators, but could pass judgments on cameras from any location) and it would increase the throughput of each operator.

7.3.5 Detecting Other Artifacts

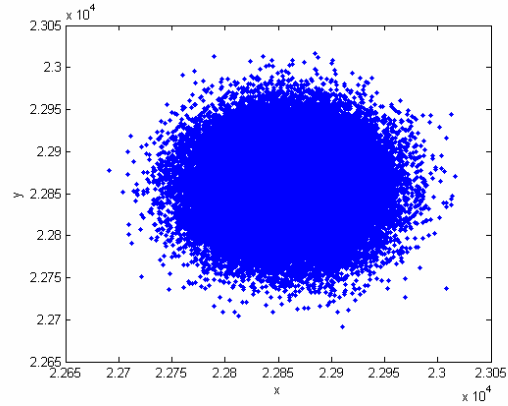
Only four different artifacts have been targeted in this thesis. However, these artifacts represent the main artifact categories: local, global and temporal artifacts. While it might be necessary to develop specialized features for each new artifact, the hope is that the discoveries resulting from this thesis can be generalized to help the development of new features. Hopefully some features will be sufficiently general and robust to be used to detect several different artifacts, but development of new features will be a continuous process. Some general image quality measures are suggested below.

7.3.5.1 A Measure of Image Uniformity

As a result of the analysis of the GLCM feature for shadow detection in section 7.1, a new feature for measuring image uniformity is suggested. An intuitive motivation will be given first. Assume that we have an image I that contains only white noise (that is each pixel is an observation of an independent random variable with some common mean and variance). Assume further that we take each pair of horizontally neighboring pixels and use the pair as coordinates to plot a point in the xy -plane (letting the first pixel represent the x coordinate and the second pixel represent the y coordinate). We would then expect to get a circular distribution of points with equal spread in all directions. Figure 7.2 illustrates how the points are generated and shows the point distribution for a random field of observations of independent random variables.



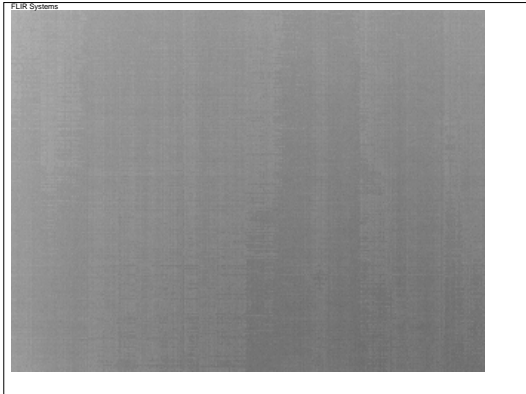
a) The horizontal pixel pair gives the coordinates of the point.



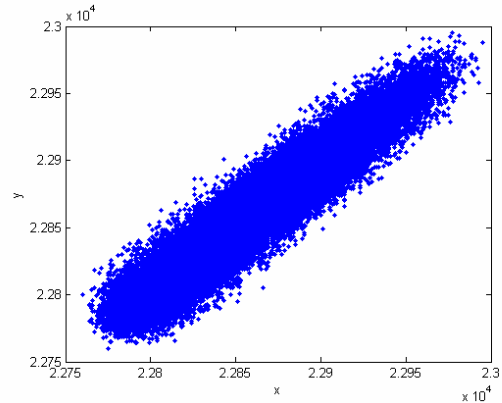
b) The neighboring pixels in a random field are independent as expected.

Figure 7.2: a) An illustration of how the point distribution is generated. b) Point distribution for a random field.

From the discussion above it can be realized that if the image was divided into two regions with slightly different mean intensities (call them m_1 and m_2), we would get two point clusters, one centered at the point (m_1, m_1) and one centered at the point (m_2, m_2) . Taking this one step further; if the intensity mean fluctuates over the image that will increase the spread in the direction $\bar{v} = (1,1)/\sqrt{2}$, but not in the perpendicular direction $\bar{u} = (-1,1)/\sqrt{2}$. An example of an image with shadows along with its point distribution is given in Figure 7.3.



a) Shadow image.



b) Point distribution for shadow image.

Figure 7.3: a) An image with shadows. b) The corresponding point distribution, which shows clear signs of non-uniformity.

It might be desirable to construct a uniformity measure that gives results in degrees Celsius (if the image gives the temperature in degrees Celsius). In this case using a covariance measure is a good suggestion (it is also not a bad idea to use the coefficient of correlation, but that would yield a normalized measure). The units of the covariance of the right and left values in horizontal pixel pairs would be degrees Celsius squared in an image giving temperature measurements in degrees Celsius in each pixel. So if the desired unit is degrees Celsius, we have to take the square root of the covariance measure. Figure 7.4 shows a plot of the uniformity measure suggested above computed on images from approved cameras and images with shadows. It can be seen that shadows generally yield large values while images from approved cameras yield values close to what would be expected for pure white noise.

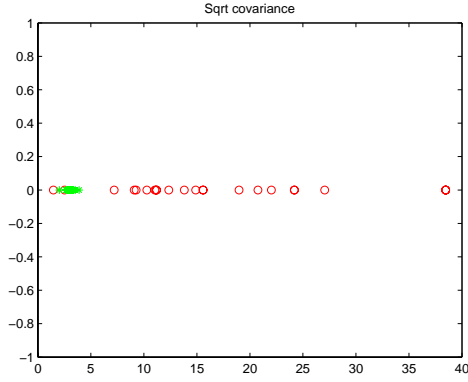


Figure 7.4: The square root of the covariance computed on images of a uniform radiation taken by approved cameras are marked by (green) stars and the uniformity measure computed on images with shadows are marked by (red) rings.

Finally, it should be noted that it is not necessary to use a horizontal pixel pair as a local neighborhood. One could just as well use a vertical pixel pair or a two by two pixel region. If the local neighborhood consists of n pixels, non-uniformity would still cause increased variance in the direction $\bar{v}_n = (1, \dots, 1)/\sqrt{n}$.

7.3.5.2 A Single Valued Fourier Feature for Periodic Artifacts

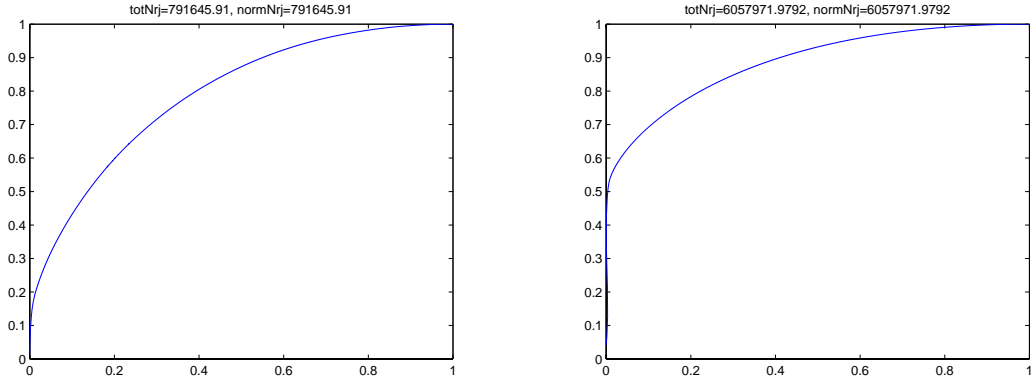
Based on the gained insights in the Fourier feature, a single valued measure is suggested in this section. It was seen that the distinguishing property of random white noise in the Fourier domain is a uniform frequency spectrum. On the contrary, images with periodic artifacts have very peaked spectra. Therefore, the energy contained in an image of white noise should be evenly distributed over the frequency spectrum of that image, while the energy in an image with periodic artifacts (or other structure for that matter) should be concentrated into a few Fourier coefficients. The energy concentrating property of the Fourier transform is often used for compressing images while retaining the structure in that image. This suggests using the energy concentration as a measure of deviance from random white noise.

An algorithm for computing a measure of the energy-concentration in the Fourier domain is described below:

1. Compute the mean intensity m in the input image I and subtract that from I to get $I' = I - m$.
2. Compute the discrete Fourier transform F of I' .
3. Compute the power spectrum P of F as $P_{i,j} = |F_{i,j}|^2$.
4. Sort P in descending order.
5. Compute the cumulative sum S of the sorted power spectrum and normalize by dividing each element in S by the last element in S (which is the total energy in the image).
6. Integrate the normalized sum as a function over the interval $[0, 1]$.
7. Return the value of the integral.

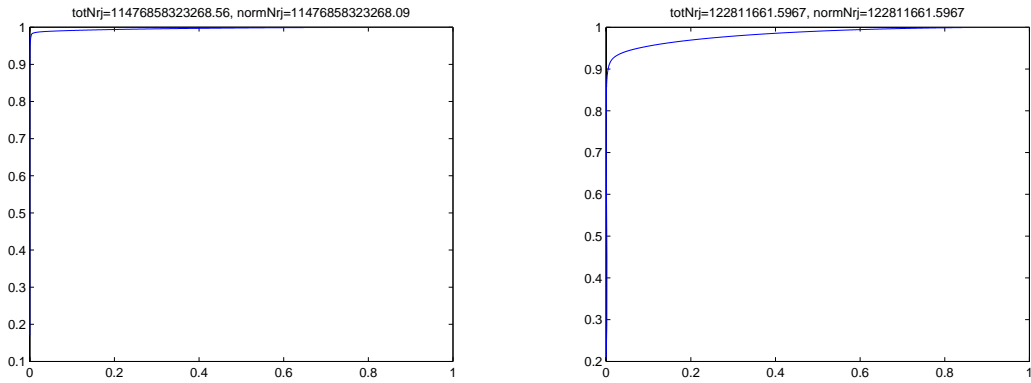
The value returned from the algorithm described above will always be in the range $[\frac{1}{2}, 1]$. If the energy is uniformly distributed over the power spectrum of the image, the returned value will be $\frac{1}{2}$. On the other hand, if all the energy in the image is contained in one single Fourier coefficient, the returned value will be one. Figure 7.5 shows the normalized sums (as calculated in step five above) for representative images with (a) no artifacts, (b) spots, (c) periodic artifacts and (d) shadows. These curves are integrated to get the final measure of energy concentration. The energy concentration measure was calculated for all images with artifacts and for a batch of images without artifacts and the result is shown in Figure 7.6. It can be seen that the images

with periodic artifacts are well separated from the images with no artifacts. The images with shadows are also distinguished by a high energy concentration. Even the spots have a tendency toward high energy concentration, even though the distinction is not as clear in this case.



a) No artifacts.

b) Spots.



c) Periodic artifact.

d) Shadows.

Figure 7.5: Plots of the normalized accumulated energy in the Fourier transform of images with (a) no artifacts, (b) spots, (c) periodic artifact and (d) shadows.



Figure 7.6: Energy concentration in images from the different classes.

7.3.6 Validation and Tuning

Before a live system can be taken into use, practical validation and tuning is of course necessary. Even though the results have shown good classification performance, this evaluation was done on a small test batch and does not guarantee the same performance in practice. It will

probably also be necessary to do additional parameter tuning. Test images for a live system will be generated under different conditions than the images in the test batch. Therefore the parameter values that were found optimal on the test batch will most likely need retuning and the classifiers will need retraining.

7.4 Alternative Set of Features

This section presents an attempt to explicitly model the distribution of images with a given type of artifact in the respective feature spaces based on the idea that the variability within an artifact class could be modeled as a (maybe multidimensional) normal distribution in feature space.

Figure 7.7 illustrates how the positive examples of an artifact are used to estimate the parameters of the normal distributions in feature space. The probability distribution function (PDF) of the normal distribution will then serve as a measure of how well an image conforms to the model of the artifact class in feature space. An image will thus yield one such conformity measure for each feature. The vector of conformity measures will then be used as the final feature vector.

During training the final feature vector described above will be computed for each (positive and negative) training example and a classifier will be trained to classify images based on this collection of conformity measures. The training procedure is illustrated in Figure 7.8.

Figure 7.9 shows how the alternative architecture is used for classification. First the features are computed and the PDFs in each feature space are evaluated to get the final vector of conformity measures. That vector is passed to the classifier that determines if the image belongs to the artifact class.

Treating the features this way was tried but it was quickly determined that this extra step of modeling the artifact class in feature space and computing the conformity measures had a negative impact on performance, compared to just feeding the features directly to the classifier.

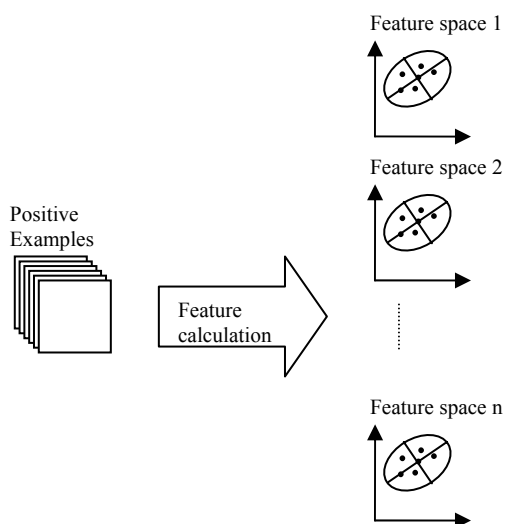


Figure 7.7: Modeling the feature space distribution of images some common artifact.

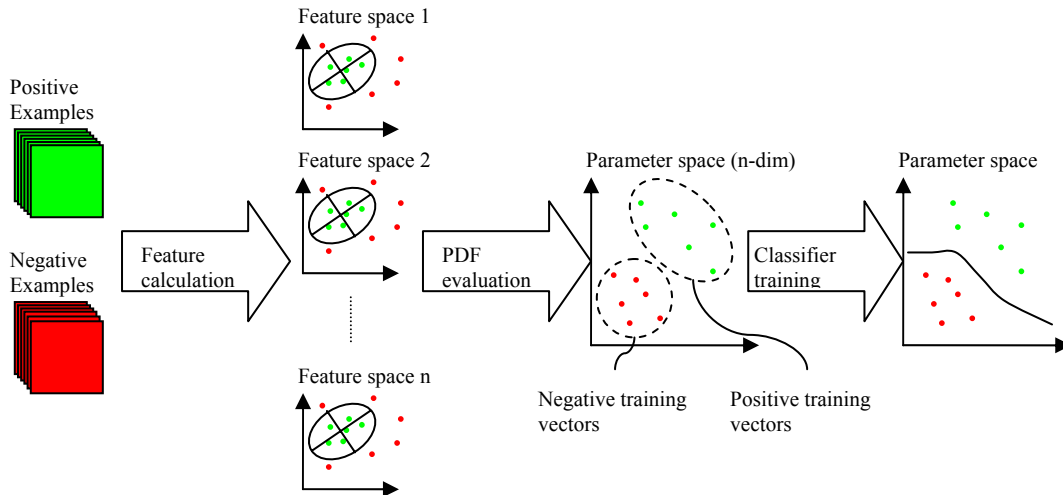


Figure 7.8: Training the classifier on the alternative set of features.

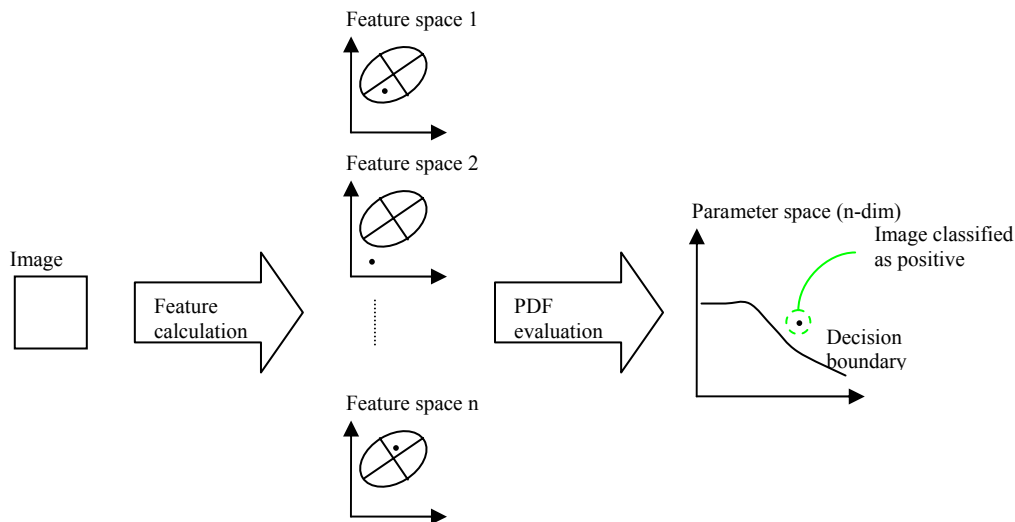


Figure 7.9: Using the classifier with the alternative set of features.

7.5 Demo Software

A demo-software was written to illustrate the artifact detection performance. The requirements, installation and use of this software are described below.

7.5.1 Requirements

The software is written in MATLAB. It requires a MATLAB installation with the Image Processing Toolbox and capability to read IR images.

7.5.2 Installation

The software will be distributed in a .zip file. Perform the following steps to make the demo-software available in the MATLAB environment:

1. Unpack the .zip file to any directory.
2. Add that directory with subfolders to the MATLAB path.

7.5.3 Use

Perform the following steps to use the demo-software to check an image:

1. Type `faultDetectionDemo` in the MATLAB prompt.
2. Select which features to use in the center panel.
3. Double-click an image in the directory in the left panel.
4. Wait for the result to appear in the right panel.

Figure 7.10 shows a screen dump of the user interface. The main elements of the interface have been numbered and are described below:

1. The left panel of the interface allows the user to select an input image from the file system. The input image must contain IR intensity information.
2. The current directory is shown at the top of the left panel.
3. The content of the current directory is listed. Input images are selected by double-clicking on the file name in the list.
4. The center panel allows the user to select which features to use for detecting artifacts in the image. Each feature has its own detector. The detector simply contains a threshold for single valued features or a support vector machine for vector valued features. The threshold or support vector machine is used to classify the image based on the feature value.
5. The right panel displays the output from each detector.
6. The final result is given last. The image is approved if all of the detectors used for the image classify the image as approved.
7. The bottom panel shows the status.

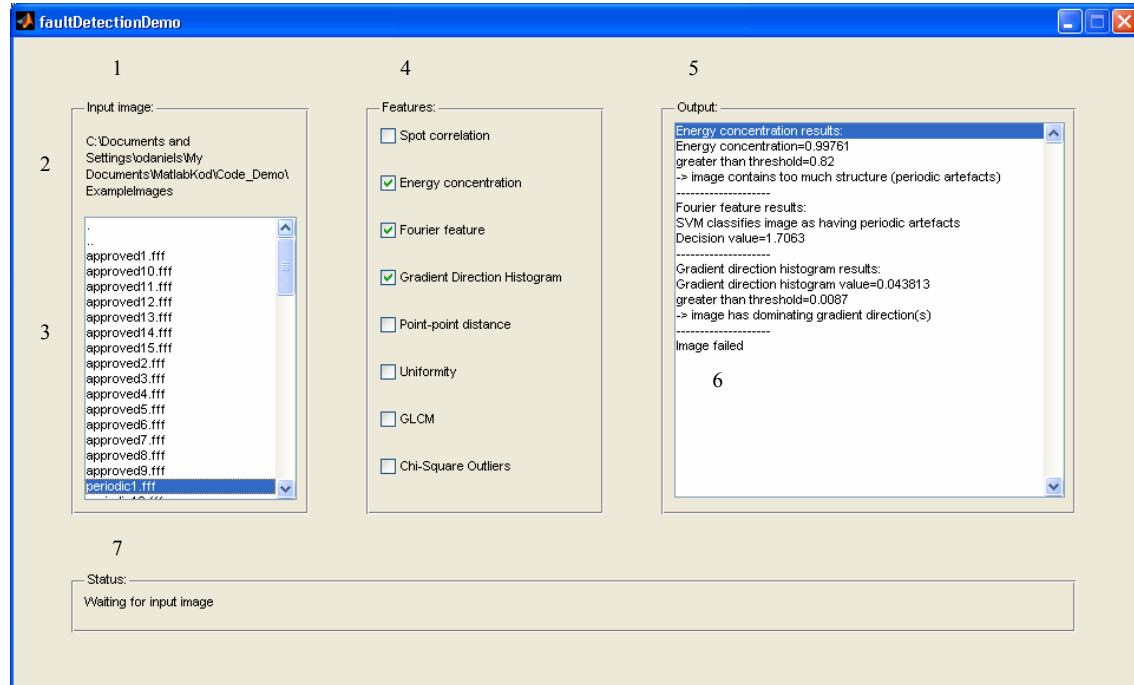


Figure 7.10: The user interface of the demo-software.

This section will be concluded with a comment on the computation time. If all features are selected, the software requires approximately six seconds complete all computations on a small image (120 by 120 pixels) using a Pentium 4 computer with one gigabyte of RAM. A large image (640 by 480 pixels) requires approximately 50 seconds. This might seem fast considering

that the computation time for the spot correlation filter alone was measured to 769 seconds in section 7.3.1. These fast computation times are possible because the spot correlation feature uses a step size of 10 in both spatial directions. While dramatically decreasing computation time (by a factor of 1/100), this does not seem to significantly decrease classification performance (note that it is important to set the scale range for the spot correlation filter sufficiently wide; as a rule of thumb, the smallest scale should equal the diameter in pixels of a small spot and the biggest scale should be 130 % of the diameter in pixels of a big spot).

8 Summary and Conclusions

The most important findings of this thesis are summarized and conclusions are drawn in this chapter.

This chapter begins with a review of some of the highlights among the findings of this thesis and then continues with some conclusions.

8.1 Summary

The purpose of this thesis was to investigate the possibility of automatically detecting subtle image artifacts that signify camera defects. Artifact detection is a two-step process. The first step is to extract measures and features from the image. The second step is to classify the image based in the measures and features extracted from the image. The classification step simply uses a standard classifier (a support vector machine was used in this thesis because it has suitable characteristics for this problem) or just a threshold, if the image is classified based on a single measure. The difficult part is to construct measures and features that separate images with artifacts from images with no artifacts, without making too strong assumptions about the artifact to be detected.

The main effort of this thesis has been concentrated on proposing and evaluating different features and measures to characterize the different artifacts. Many novel measures have been proposed and the results have often showed that simple, single valued measures that can be thresholded for classification yield comparable classification performance to more complex feature vectors that need a support vector machine to provide the classification. The results have also given insight into how the artifacts should best be described, which has inspired the suggestion of additional measures for artifact detection. Some of the more interesting measures will be mentioned next.

The first artifact studied was named spots because the visual appearance as slightly darker or brighter spots against a uniform background. The most notable feature for spot detection is the correlation between distance from the center of the image patch and intensity. This is a single-valued measure that corresponds well to the visual appearance of spots, which suggests actually using the correlation measure to define what a spot is. This measure can be implemented as a filter and the maxima in the response of this filter give the locations of the most probable spot occurrences in the image.

The alternative method for spot detection was to scan an image and feed downsampled versions of local image patches to a support vector machine that determines if the image patch is a spot or not. In this context an interesting method for avoiding false positive detections was demonstrated. One problem with both spot detection methods was the long computation time.

For detection of periodic artifacts, a feature based on the Fourier transform was suggested and showed good performance when used together with a support vector machine with linear kernel. Alternatively, the non-uniformity of the gradient direction histogram was a single-valued measure that also turned out to give good performance. Both features had short computation times.

The measures and feature suggested for detection of shadows were all successful in separating the images with and without shadows. The point to point distance distribution measure, which is single valued, gave good separation but misclassified one example. The graininess feature correctly classified all images in the example batch but failed in validation (see Appendix A). A good, simple and visually correspondent measure for detecting both shadows and general non-uniformity was suggested in section 7.3.5.1. This feature would be the firsthand choice for shadow detection, if it succeeds in evaluation. The point to point distance distribution or the

GLCM feature (which has many unexplored variations) would be the second best alternative. The graininess feature should be avoided.

Finally, a temporal artifact called row noise was investigated. This artifact was characterized by a row-oriented flickering seen in image sequences. Several measures were tried and only one, the χ^2 -distance measure (which was single-valued), gave good performance. One surprise was that only very poor performance was achieved by measuring the variance over time of the row mean intensities. Intuitively, this would be a distinguishing characteristic of image sequences with row noise. Obviously more work has to be done, maybe focusing harder on the time dimension.

In general all methods mentioned above serve to detect structure in what should be random white noise. What separates the methods is what kind of structure they detect (local correlation, periodic patterns, weak connected edges or non-uniformity for example). So artifacts are detected based on how they deviate from random white noise.

8.2 Conclusions

The main conclusion of this thesis is that there are good prospects of successfully detecting image artifacts automatically. Some artifacts can even be detected with high accuracy using only simple and intuitive single-valued measures instead of using complex features that need support vector machines for classification. The single valued measures that correspond well to the visual appearance of their respective artifacts could potentially be used to define that artifact. For example, a spot could be defined as an image patch where the distance from the center of the patch and the intensity are correlated above some threshold. It is generally desirable to have measures that are simple, intuitive and descriptive.

In some cases, the single valued measures are not sufficient to achieve the desired classification performance. In these cases complex features and support vector machines have to be used. However, single valued measures might be valuable as separate complements in those cases.

In general, there is no gain in performance by using more than one feature as input to a support vector machine compared to using only the best feature in the set.

Bibliography

- [1] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," in *Pattern Recognition 13*, 1981, pp. 111–122.
- [2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [3] T. Kadir and M. Brady, "Saliency, Scale and Image Description," *International Journal of Computer Vision* 45(2), 83–105, 2001.
- [4] T. Kadir and M. Brady, "Scale saliency: A novel approach to salient feature and scale selection," in *International Conference Visual Information Engineering*, pp. 25–28, 2003.
- [5] Shapiro and Stockman, "Computer Vision," Prentice-Hall, 2001.
- [6] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics*, vol. 21(2), pp. 224–270, 1994.
- [7] D. Marr and E. Hildreth, "A theory of edge detection," *Proceedings of the Royal Society of London B*, 207, 187 – 217, 1980.
- [8] J. Canny, "A computational approach to edge detection," in *IEEE Transaction Pattern Analysis Machine Intelligence* 8(6), pp. 679–698, 1986.
- [9] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection", *International Journal of Computer Vision*, 30, 2, pp 117 – 154, 1998.
- [10] C. Harris and M. Stephens, "Combined corner and edge detector," in *Proc. Fourth Alvey Vision Conference*, pp. 147–151, 1988.
- [11] T. Lindeberg, "On scale selection for differential operators," in *Proc. 8th Scandinavian Conf. on Image Analysis*, pp. 857 – 866, 1993.
- [12] K. Mikolajczyk and C. Schmid "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10, 27, pp 1615--1630, 2005.
- [13] M. Varma and A. Zisserman, "Texture Classification: Are Filter Banks Necessary?," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 477-484, 2003.
- [14] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 511-517, 2004.
- [15] S. Lazebnik, C. Schmid, and J. Ponce, "Sparse Texture Representation Using Affine-Invariant Neighborhoods," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 319-324, 2003.
- [16] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *IJCV*, 43(1):29–44, 2001.
- [17] O. G. Cula and K. J. Dana, "Compact representation of bidirectional texture functions," in *Proc. CVPR*, pp. 1041–1047, 2001.
- [18] Y. Li and L. G. Shapiro, "Object class recognition using images of abstract regions," in *Proceedings of the International Conference on Pattern Recognition*, pp. 40-43, 2004.
- [19] Y. Li, L. G. Shapiro and J. A. Bilmes, "A generative/discriminative learning algorithm for image classification," in *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, vol. 2, pp. 1605–1612, 2005.
- [20] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, "Content-Based Image Retrieval at the End of the Early Years", *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 22, No. 12, 2000.

- [21] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *PAMI*, 5(9):1075–1088, 2003.
- [22] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu, "Discovering texture regularity as a higher-order correspondence problem," in *Proceedings of ICCV 2006*, 2006.
- [23] R. Fergus, P. Perona, and A. Zisserman, "Object-class recognition by unsupervised scale-invariant learning," in *CVPR*, pp. 2:264–271, 2003.
- [24] M. Weber, M. Welling and P. Perona, "Unsupervised Learning of Models for Recognition," in *ECCV (I)*, pp. 18-32, 2000.
- [25] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, 40(2):99-121, November 2000.
- [26] T. Maenpaa, T. Ojala, M. Pietikainen, M. Soriano, "Robust Texture Classification by Subsets of Local Binary Patterns," *15th International Conference on Pattern Recognition (ICPR'00)*, Vol. 3, pp. 3947-3950, 2000.
- [27] T. M. Mitchell, "Machine Learning," McGraw-Hill, 1997, ISBN: 0-07-115467-1.
- [28] S. Russell and p. Norvig, "Artificial Intelligence – A Modern Approach," Prentice Hall, 2003, ISBN 0-13-790395-2.
- [29] L. Wolf and S. Bileschi, "Combining Variable Selection with Dimensionality Reduction," in *CVPR*, Vol. 2, pp. 802-806, 2005.
- [30] A. L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," *Artificial Intelligence*, 97:245-271, 1997.
- [31] Rowley, H., Baluja, S., and Kanade, T., "Neural Network-Based Face Detection", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 203-207, 1996.
- [32] Blom, G., "Sannolikhetsteori och Statistikteori med Tillämpningar", Studentlitteratur, 1989, ISBN: 91-44-03594-2.
- [33] Y. Rubner, C. Tomasi and L. J. Guibas, "A metric for distributions with applications to image databases," *ICCV*, 1998, pp.59-66.
- [34] Majer, P. (1999). Self-similarity of noise in scale-space. In Nielsen, M., Johansen, P., Olsen, O., and Weickert, J., editors, *Scale-Space Theories in Computer Vision, Scale-Space99*, volume 1682 of *Lecture Notes in Computer Science*.
- [35] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.
- [36] U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Machine Learning*, 1992, 8, pp. 87-102.
- [37] B. E. Usevich, "A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG 2000," *IEEE Signal Processing Magazine*, 2001, pp. 22-35.
- [38] R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, 1973, pp. 610-621.
- [39] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, 1991, pp. 71-86.
- [40] OSU SVM - download site, 2006-12-05. <http://sourceforge.net/projects/svm/>
- [41] "Termografi Nivå 1 Kursmaterial," Infrared Training Center, FLIR Systems AB, Rinkebyvägen 19, S-18211 DANDERYD, Sweden, 2006.

Appendix A – Detailed Results

Appendix A contains a more complete set of results than the results section.

The purpose of this appendix is to give a detailed evaluation of the classification performance of each feature as a function of the parameters involved in feature computation. This is interesting because it gives insight into the sensitivity of the features to their parameters. Insensitivity is of course desired.

Spot Detection

This section contains results describing how the spot-detection performance depends on the various parameters used when calculating features and training the classifier.

Table 12 lists the classification performance when using only the Radon profile feature to describe the images. The performance is given for several different values of the parameters affecting the Radon profile calculation. Tables 13-16 give the results of analogous experiments for each of the other features suggested for spot detection. Figure A.1 shows the distance from center and intensity correlation feature values for each image in the batch. Table 17 summarizes the best classification performance and the corresponding parameter settings for each feature.

Table 12: Classification performance for different parameter settings using only radon profiles as feature.

		n_{bins}							
		3	4	5	6	7	8	12	16
c_d	0.001	0.8511	0.8936	0.8830	0.8617	0.8617	0.8617	0.8617	0.9043
	0.005	0.8617	0.8617	0.8830	0.8830	0.8617	0.8617	0.8830	0.8617
	0.01	0.8617	0.8617	0.8830	0.8617	0.8617	0.8617	0.9149	0.8830
	0.1	0.5638	0.5851	0.5532	0.3936	0.2021	0.0319	0	0

Table 13: Classification performance for different parameter settings using only edge pixel histogram as feature.

		n_{bins}			
		6	8	10	12
σ	0.5	0.5532	0.5851	0.6383	0.6489
	1	0.7340	0.7553	0.7553	0.7766
	1.5	0.7660	0.7660	0.7766	0.7766
	2	0.7872	0.7979	0.8191	0.8085
	3	0.8085	0.8191	0.8298	0.8191
	4	0.8298	0.8191	0.8298	0.8298
	5	0.7553	0.7766	0.7979	0.7979

Table 14: Classification performance for different parameter settings using only spin image histogram as feature.

		$n_{\text{intensity bins}}$								
		2	3	4	5	6	7	8	10	12
$n_{\text{distance bins}}$	2	0.8723	0.9362	0.9149	0.9149	0.9255	0.9043	0.9149	0.8936	0.8936
	3	0.9255	0.9681	0.9574	0.9468	0.9468	0.9362	0.9362	0.9362	0.9574
	4	0.8936	0.9681	0.9681	0.9362	0.9255	0.9255	0.9362	0.9255	0.9362
	5	0.8830	0.9574	0.9574	0.9468	0.9362	0.9362	0.9468	0.9574	0.9362
	6	0.8830	0.9574	0.9468	0.9362	0.9362	0.9468	0.9468	0.9574	0.9362
	7	0.9043	0.9574	0.9255	0.9468	0.9362	0.9468	0.9255	0.9362	0.9255
	8	0.8723	0.9468	0.9255	0.9362	0.9362	0.9468	0.9255	0.9468	0.9362

Table 15: Classification performance for different parameter settings using only gradient direction histogram as feature.

n_{reg}				
2	3	4	5	6
0.6809	0.7872	0.8723	0.8617	0.9255

Table 16: Classification performance for different parameter settings using only the resampled (usually downsampled) version of the original image as feature.

n_{pixels}										
4	6	8	12	16	20	24	28	32	36	40
0.9681	0.9787	0.9894	0.9681	0.9468	0.9787	0.9787	0.9681	0.9681	0.9681	0.9787

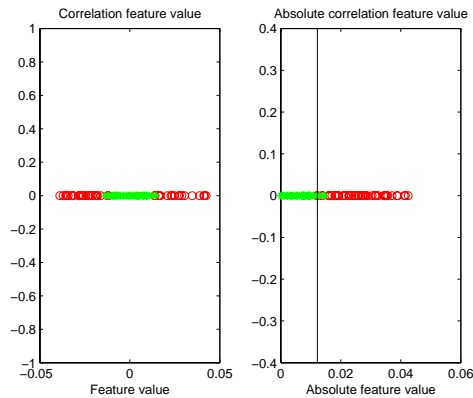


Figure A.1: Covariance between the normalized distance from the center of the image patch and the normalized intensity. The values for the positive examples with spots are marked as red rings and the values for the negative examples without spots are marked as green stars. The second plot shows the absolute feature values and the black line marks the optimal threshold, which is located at 0.0122, for classification. When using this threshold, the classification performance is 0.9574.

Table 17: The best classification performance and corresponding parameter settings for each feature.

Feature	Best performance	Parameter settings
Radon profiles (1)	0.9043	$c_d = 0.001$ $n_{bins} = 16$
Edge pixel histogram (2)	0.8298	$\sigma = 4$ $n_{bins} = 10$
Spin image (3)	0.9681	$n_{dist} = 4$ $n_{int} = 3$
Gradient direction histogram (4)	0.9255	$n_{bins} = 6$
Resampled original (5)	0.9894	$n_{pixels} = 8$
Center distance and intensity correlation (6)	0.9572	

Detection of Periodic Artifacts

Figure A.2 shows the classification performance when using only the Fourier transform feature. The classification performance is plotted against the number of frequency components included in the feature vector.

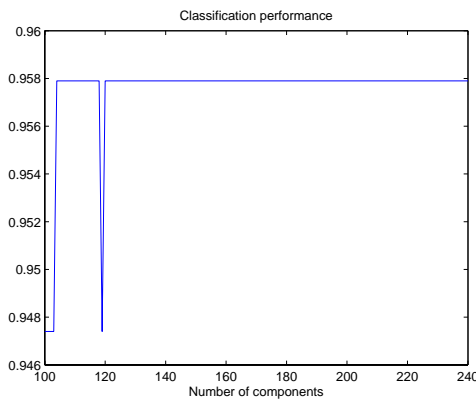


Figure A.2: Classification performance of the Fourier transform feature plotted against the number of frequency components included in the feature vector. The best performance achieved was 0.9579.

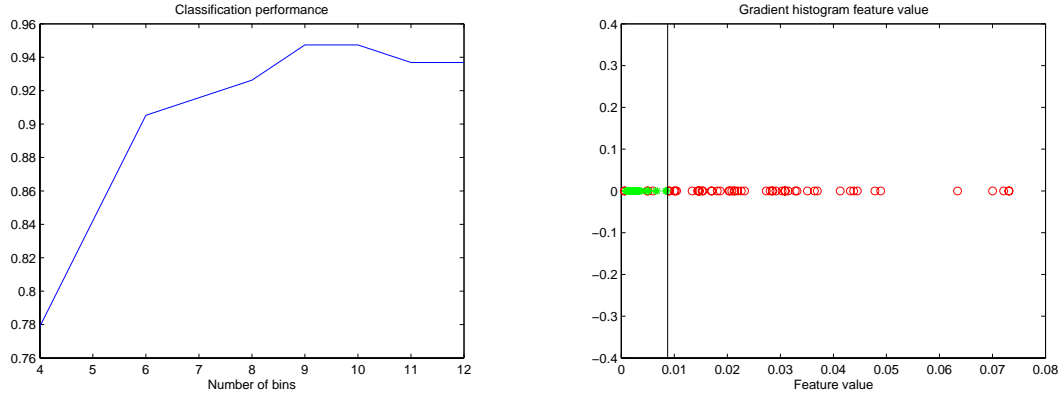
The components of the frequency vector differ widely in range, so it might be wise to scale the components of the feature vector to prevent components with greater numeric ranges to dominate components with smaller numeric ranges. Another advantage of scaling in this case is that if the components of the feature vectors are too large, there might be numerical problems when calculating the inner products in the kernel. To determine the value of scaling, the feature vectors (computed with 150 components) were scaled so that each component had the range $[-1, 1]$ before being passed to the leave-one-out procedure. This improved the classification performance to 0.9895.

To put these results in perspective, the classification performance when using the unprocessed Fourier transform matrix as a feature was evaluated. Table 18 gives the classification result. Scaling did not improve the results in this case.

Table 18: The classification performance for the unprocessed Fourier feature. Scaling did not improve the results in this case.

0.9579

Next, the performance of the gradient histogram feature was evaluated. The results are shown in Figure A.3, where the classification result is plotted against the number of bins in the gradient histogram. Since the feature was designed to give low values for images with no periodic artifacts and high values for images with periodic artifacts, a simple threshold was used for classification (if the feature value is above the threshold, the image will be considered to contain periodic artifacts).



a) The classification performance plotted against the number of bins for the gradient histogram feature.

b) The gradient histogram feature value for each image in the batch was plotted on the real number line.

Figure A.3: In a) the best performance was 0.9474, when using 9 bins. The optimal threshold in this case was 0.0087. In b) the example images with periodic artifacts have been marked with (red) rings and the images without periodic artifacts have been marked with (green stars). The feature values have been calculated using 9 bins and the threshold, 0.0087, is marked by a black vertical line.

The performance when combining the two features was also evaluated. The Fourier transform feature was calculated using 150 frequency components and the gradient histogram feature value was calculated using 9 bins. The data was scaled before being passed to the leave-one-out cross-validation procedure. Table 19 gives the performance when using both features for classification.

Table 19: Classification performance when using both features

0.9895

Detection of “Shadows”

Figure A.4 shows the classification performance of the GLCM feature plotted against the number of distances included. The dashed line shows the performance when the images were band-pass filtered (using Gaussian kernels with $\sigma_{lp}=2$ for the low-pass filter and $\sigma_{hp}=10$ for the high-pass filter) before GLCM calculation. The elements of the GLCM feature vectors were scaled to the range $[-1,1]$ before being passed to the SVM leave-one-out cross-validation. The best classification performance was 0.9863, which was achieved when 12 to 16 distances were used and no band-pass filtering was applied.

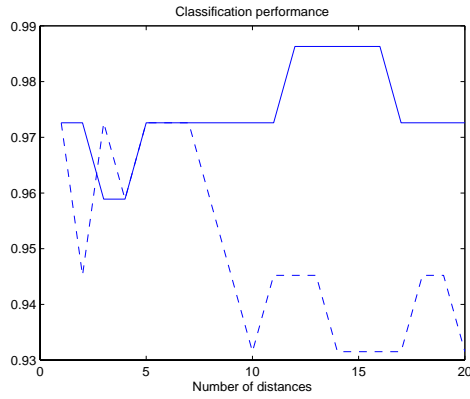


Figure A.4: The classification performance (evaluated using leave-one-out cross-validation) for shadow detection when using only the GLCM feature plotted against the number of distances included when generating the GLCM matrices. The dashed line shows the performance when the images were band-pass filtered (using Gaussian kernels with $\sigma_{lp}=2$ for the low-pass filter and $\sigma_{hp}=10$ for the high-pass filter) before GLCM calculation.

The computation of the point to point distance feature depends on two parameters, σ_{lp} and σ_{hp} , representing the low-pass and high-pass filters, respectively. Table 20 shows the classification performance for different values of the filter parameters. The images were classified using an optimal threshold on the calculated feature values.

Table 20: The classification performance when thresholding on the point-point distance feature calculated using several different settings for the band-pass filter parameters σ_{lp} and σ_{hp} .

		σ_{hp}						
		5	7	9	10	11	13	15
σ_{lp}	0.1	0.9863	0.9863	0.9863	0.9863	0.9863	0.9863	0.9863
	0.5	0.9863	0.9863	0.9863	0.9863	0.9863	0.9863	0.9863
	1	0.9863	0.9863	0.9863	0.9863	0.9726	0.9589	0.9589
	1.5	0.9589	0.9589	0.9178	0.9041	0.8904	0.8904	0.8904
	2	0.9452	0.9041	0.8904	0.8493	0.8356	0.8356	0.8356
	2.5	0.9315	0.8767	0.8356	0.8219	0.8082	0.7808	0.7671
	3	0.8904	0.8219	0.7808	0.7671	0.7534	0.7534	0.7671

The best classification performance was 0.9863, which was achieved for several different parameter settings. In order to determine which setting to prefer, the separation between the two classes (measured as the smallest feature value from the images with shadows minus the largest feature value from the images without shadows) was calculated for the same values of the σ_{lp} and σ_{hp} parameters. The results are shown in Table 21. Since none of the settings yielded a classification performance of one, naturally the separation is always negative. The biggest separation was achieved for $\sigma_{lp}=0.5$ and $\sigma_{hp}=9$. The smallest feature value from the images with shadows was then 0.56597 and the biggest value from the images without shadows was 0.58022, thus giving the separation $0.56597-0.58022 = -0.01425$. The feature values for all images using $\sigma_{lp}=0.5$ and $\sigma_{hp}=9$ are plotted in Figure A.5. Images with shadows have been marked with (red) rings and images without shadows have been marked with (green) stars. The threshold used for determining classification performance was 0.6286 and is marked with a black line in the plot.

Table 21: The separation between the smallest point-point distance feature value for the images with shadow artifacts and the biggest feature value for images without shadow artifact.

		σ_{hp}						
		5	7	9	10	11	13	15
σ_{lp}	0.1	-0.0320	-0.0239	-0.0223	-0.0222	-0.0217	-0.0212	-0.0210
	0.5	-0.0246	-0.0182	-0.0142	-0.0147	-0.0150	-0.0163	-0.0156
	1	-0.0624	-0.1229	-0.1149	-0.1155	-0.1431	-0.1443	-0.1436
	1.5	-0.2201	-0.2628	-0.2830	-0.3007	-0.3039	-0.3021	-0.3021
	2	-0.2795	-0.2965	-0.2871	-0.2821	-0.2873	-0.2862	-0.2873
	2.5	-0.2697	-0.2447	-0.2729	-0.2726	-0.2773	-0.2823	-0.2921
	3	-0.2067	-0.2752	-0.2826	-0.2946	-0.3015	-0.3133	-0.3184

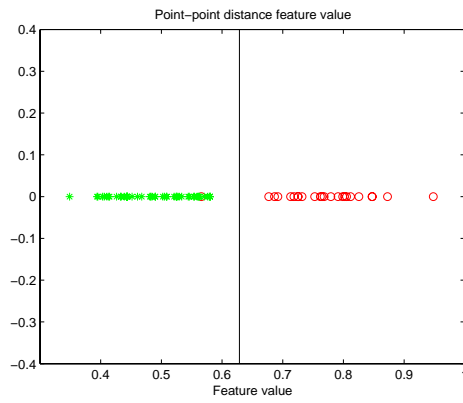


Figure A.5: Plot showing the point-point distance feature values for images with shadows (marked with red rings) and images without shadows (marked with green stars). The threshold used for classification was 0.6286 and is marked with a black line in the plot. There is one false negative classification (that is one image with shadows was classified as having no shadows).

The classification performance when using both of the above features was evaluated. Feature calculation was done using the best parameter settings according to the previous results. Table 22 shows the classification performance in this case. Each feature was scaled to the range $[-1, 1]$ before being passed to the leave-one-out performance evaluation. The classification did not improve when using both features.

Table 22: Classification performance when using both features for shadow detection

0.9863

The last feature to evaluate is the graininess feature, which takes the σ -values for the low- and high-pass filters as parameters. The classification performance was determined for a range of parameter values and the results are shown as a set of contour plots in Figure A.6. The classification performance, separation and optimal threshold are plotted as functions of σ_{lp} and σ_{hp} .

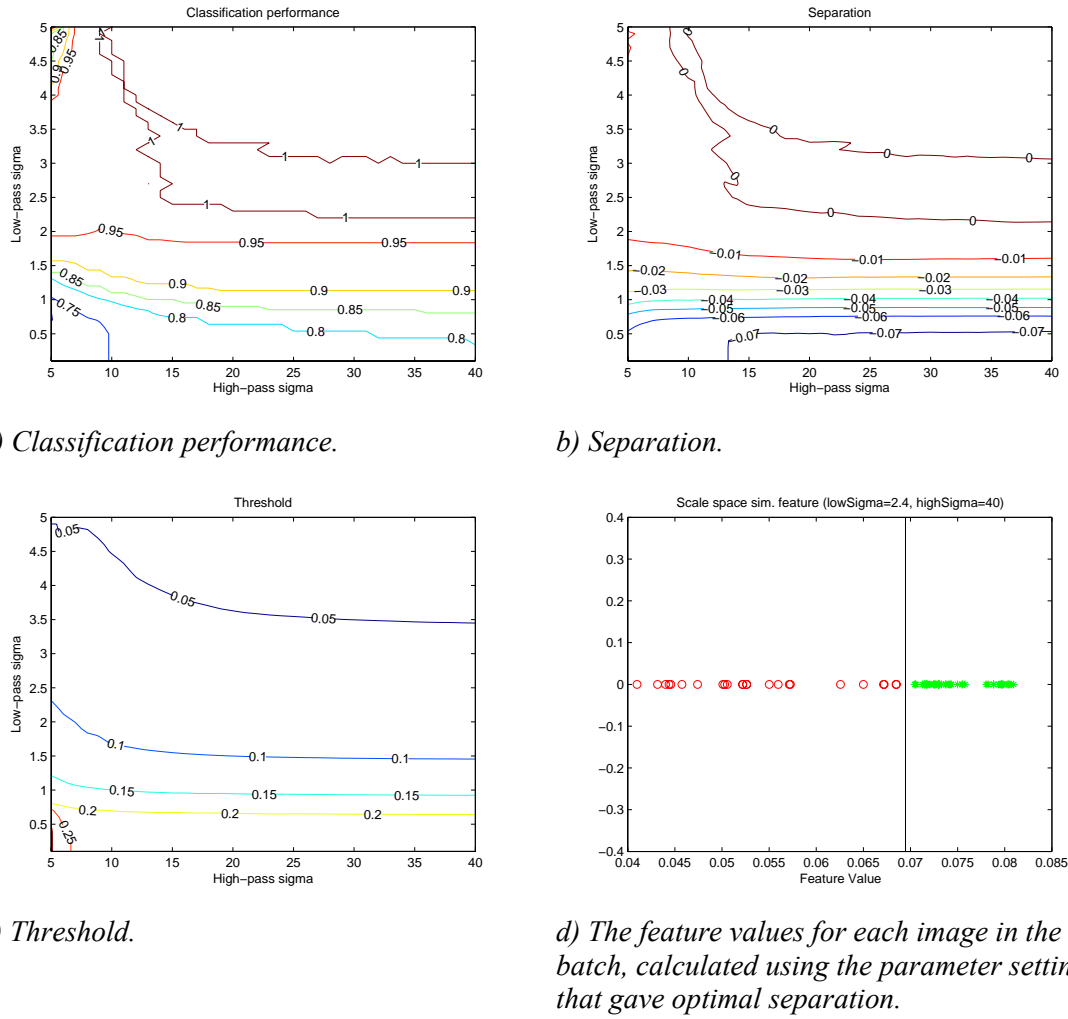


Figure A.6: Performance evaluation for the graininess feature.

How good is the graininess feature with these parameter values at classifying new examples using the threshold marked in Figure A.6? Due to lack of example images with shadows the validation set compiled to answer that question contains only images without shadows. The feature values for these images are plotted in Figure A.7, which shows that several images will be wrongly classified as having shadows. This example illustrates the importance of validation.

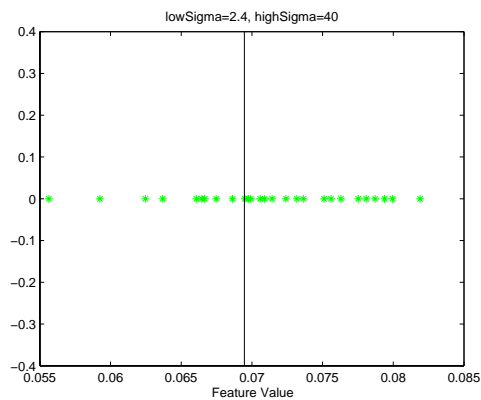


Figure A.7: The feature values of a set of validation images without shadows. Several images will be wrongly classified as having shadows.

Detection of Row noise

This section contains the evaluation of the features for detection of row noise. Figure A.8 and A.9 give the results for the variance of row intensity means feature. The feature values for the image sequences without row noise have been marked with (green) stars and the feature values for the image sequences with row noise have been marked with (red) rings in Figure A.8. In the first plot of Figure A.9, the variance of column intensity means has been calculated also and each image sequence is represented by a point in the plane with the variance of row intensity means on the x axis and the variance of column intensity means on the y axis. In general all image sequences are close to the line $y=x$ in this plane and the separation is not good. The second plot shows the absolute value of the variance of row intensity means minus the variance of column intensity means.

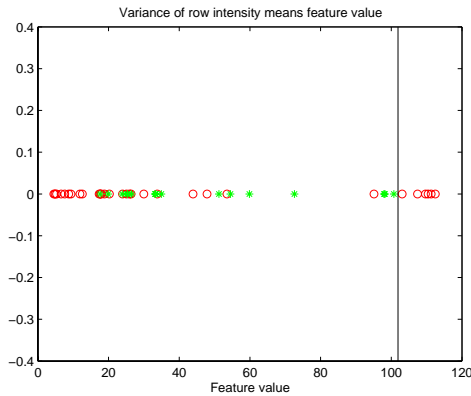


Figure A.8: The feature values for image sequences with and without row noise. The classification performance when thresholding the feature value was 0.4783. The optimal threshold was 101.9058.

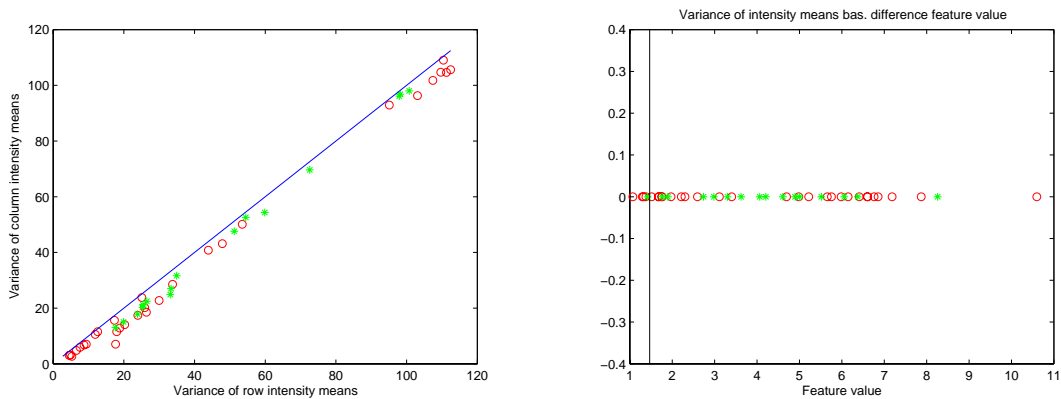


Figure A.9: Results for row noise detection using the variant of the variance of row intensity means feature. The classification performance when thresholding the final feature value was 0.5870 and the optimal threshold was found to be 1.468.

These results present one surprise. Assuming that the pixel intensities could be modeled as independent normally distributed random variables with a common expected value and standard deviation, a formula for the theoretical variance of row intensity means was stated in equation 5.2. The theoretical value was estimated and used to obtain a normalized feature value. Thus it was expected that the feature values should be close to one, but as can be seen in Figure A.8 they are far bigger than that.

The second feature proposed was the row intensity variance minimum. The results for this feature are shown in Figures A.10 and A.11. The feature values for the image sequences without row noise have been marked with (green) stars and the feature values for the image sequences

with row noise have been marked with (red) rings. In the first plot of Figure A.11, the column intensity variance minimum has also been calculated and each image sequence is represented by a point in the plane with the row variance minimum on the x axis and the column variance minimum on the y axis. As for the previous feature, it was expected that the image sequences without row noise be closer to the line $y=x$ in this plane. This expectation is fulfilled and the separation is better than for the previous feature. The second plot shows the absolute value of the row intensity variance minimum minus the column intensity variance minimum.

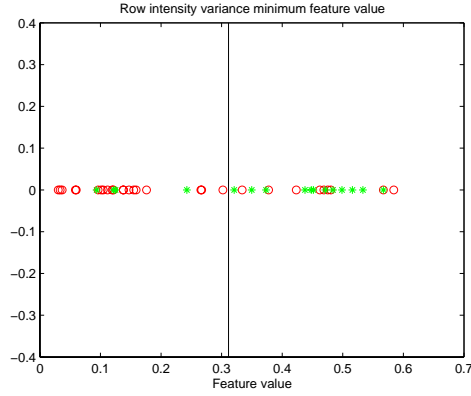


Figure A.10: The feature values for image sequences with and without row noise. The classification performance when thresholding the feature value was 0.7174. The optimal threshold was 0.3114.

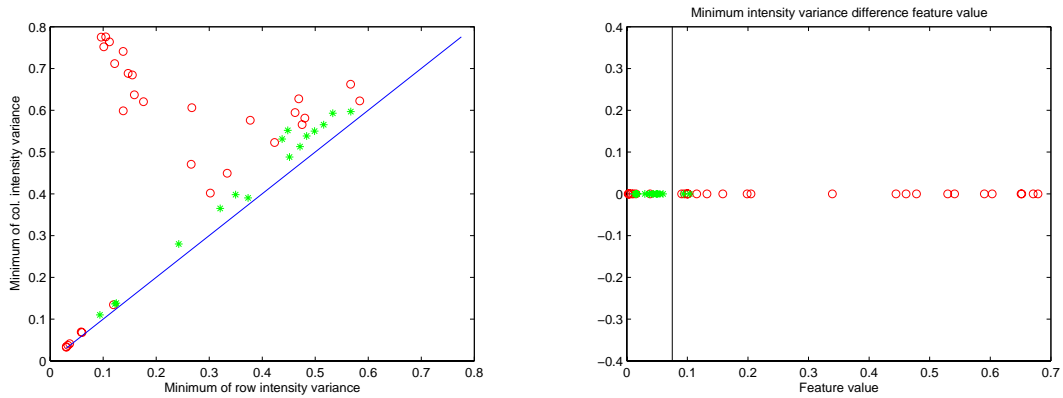


Figure A.11: Results for row noise detection using the variant of the row intensity variance minimum feature. The classification performance when thresholding the final feature value was 0.7826 and the optimal threshold was found to be 0.0752.

Figures A.12-A.18 shows performance plots and feature value plots for the remaining features proposed for row noise detection.

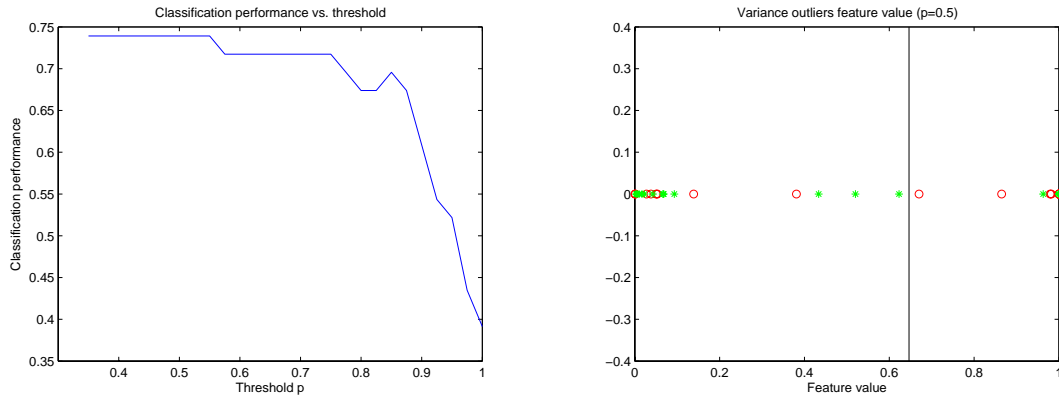


Figure A.12: The classification performance of the variance and mean outliers feature. The first plot shows the classification performance plotted against the value of the parameter p . The best performance was achieved for (among others) $p=0.5$. The second plot shows the feature values (calculated for $p=0.5$) for the image sequences with row noise, marked as red rings, and the image sequences without row noise, marked as green stars. The optimal threshold is 0.6467 and has been marked with a black line. The classification performance was 0.7391.

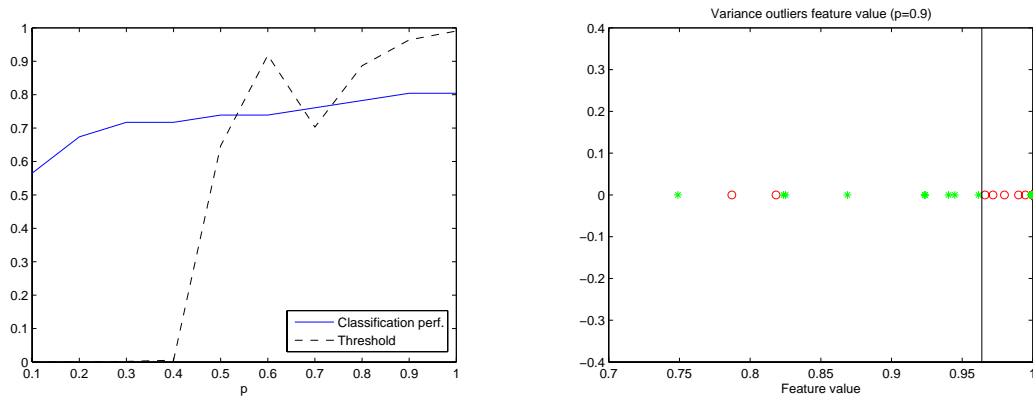


Figure A.13: The classification performance of the variance outliers feature. The first plot shows the classification performance plotted against the value of the parameter p . The best performance was achieved for $p=0.9$. The second plot shows the feature values (calculated for $p=0.9$) for the image sequences with row noise, marked as red rings, and the image sequences without row noise, marked as green stars. The optimal threshold is 0.9639 and has been marked with a black line. The classification performance was 0.8043.

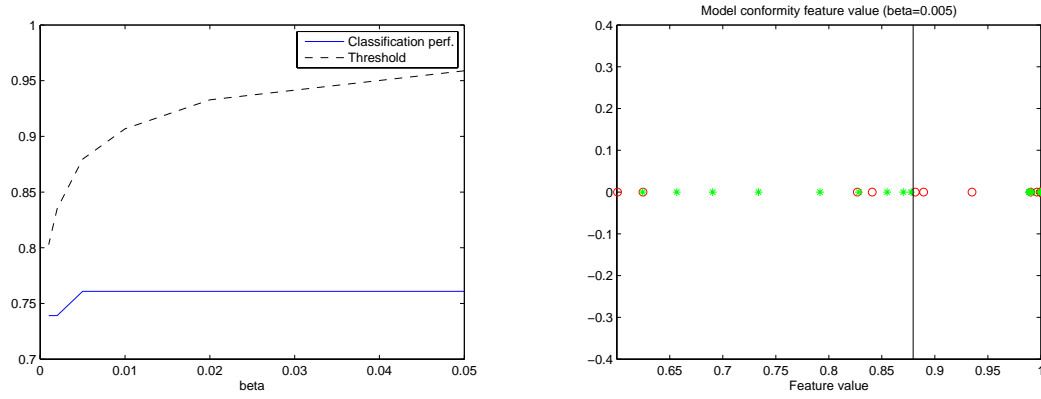


Figure A.14: The classification performance of the model conformity feature. The first plot shows the classification performance plotted against the value of the parameter β . The best performance was achieved when, for example, $\beta=0.01$. The second plot shows the feature values (calculated for $\beta=0.01$) for the image sequences with row noise, marked as red rings, and for the image sequences without row noise, marked as green stars. The optimal threshold was 0.8796 and has been marked with a black line. The classification performance was 0.7609.

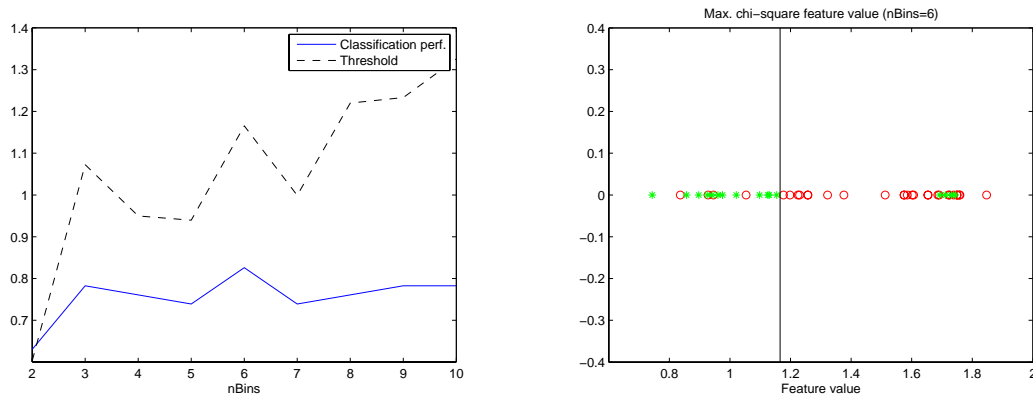


Figure A.15: The classification performance of the maximum χ^2 distance feature. The first plot shows the classification performance plotted against the number of bins. The best performance was achieved when the number of bins was set to 6. The second plot shows the feature values (calculated using 6 bins) for the image sequences with row noise, marked as red rings, and for the image sequences without row noise, marked as green stars. The optimal threshold was 1.1656 and has been marked with a black line. The classification performance was 0.8261.

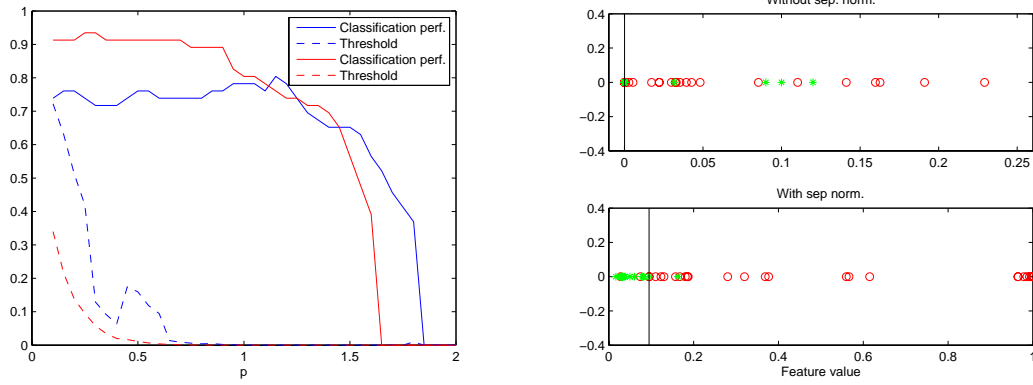


Figure A.16: The classification performance of the number of χ^2 distance outliers. The intensity histograms used for the χ^2 distance calculations had 6 bins. The first plot shows the classification performance and optimal threshold plotted against the value of the parameter p . The red curves represent the result after separate normalization of each frame. Then the best performance, 0.9348, was achieved for $p=0.25$. The optimal threshold was then 0.094917. The best performance without separate normalization of each frame was 0.8044, achieved for $p=1.15$ and with optimal threshold $4.1667e-005$. The second plot shows the feature values (calculated for $p=0.25$ and $p=1.15$ respectively) for the image sequences with row noise, marked as red rings, and for the image sequences without row noise, marked as green stars. The optimal thresholds have been marked by black vertical lines.

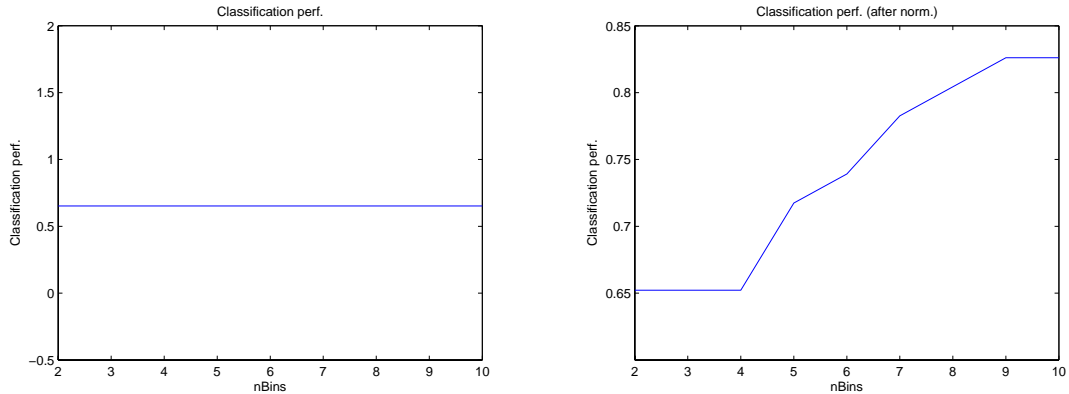


Figure A.17: The leftmost plot shows the classification performance of the χ^2 distance distribution feature (which is described as a variation of the χ^2 distance outliers feature in section X), plotted against the number of bins used in the χ^2 distance histogram. The intensity histograms used for the χ^2 distance calculations had 5 bins. The classification in this case required the use of a support vector machine. The classification performance was constant at 0.6522. The rightmost plot shows the classification performance when the each frame was normalized separately. In this case the classification performance improved to 0.8261 when using 9 bins.

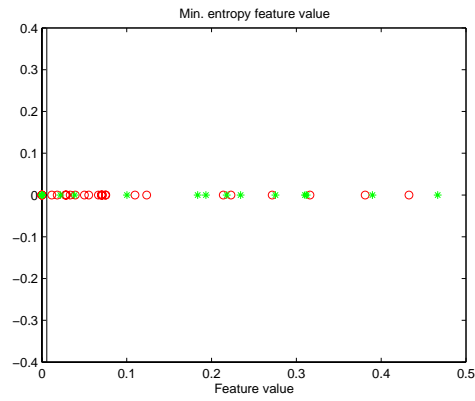


Figure A.18: The classification performance of the minimum entropy feature. The plot shows the feature values for the image sequences with row noise, marked as red rings, and for the image sequences without row noise, marked as green stars. The optimal threshold was 0.0058 and has been marked with a black line. The classification performance was 0.5870.