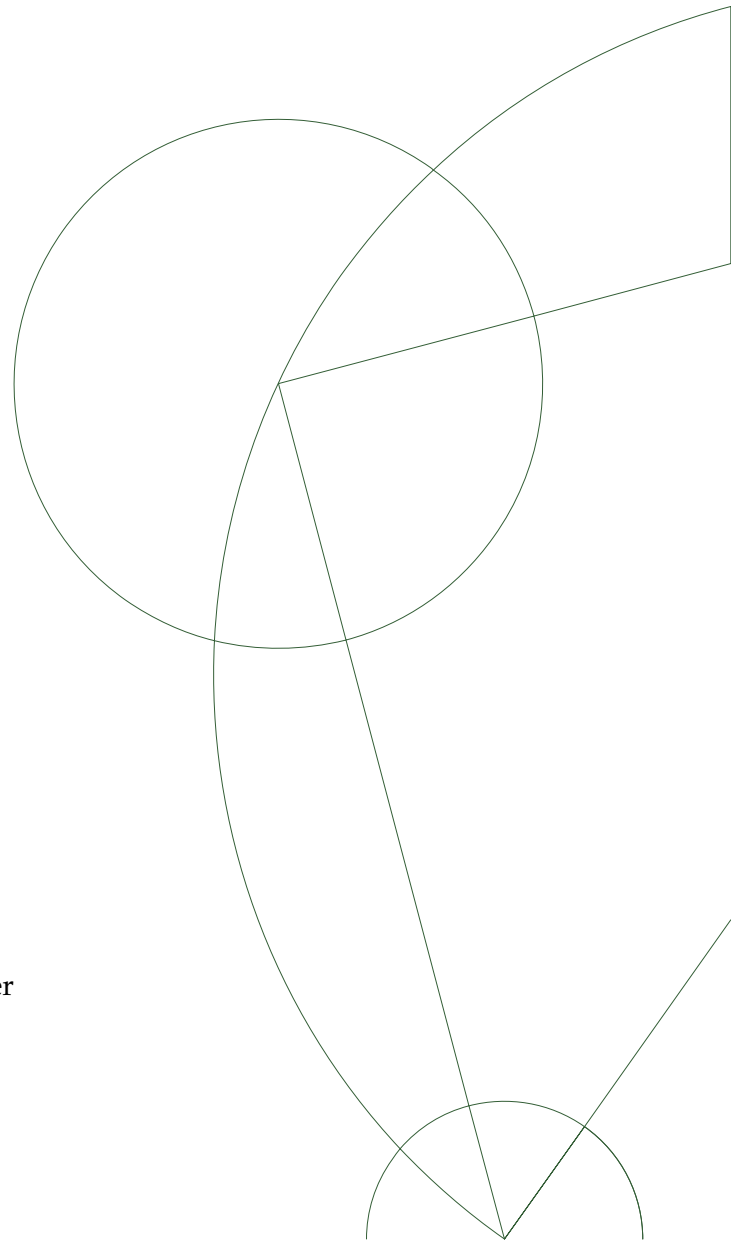
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Master's Thesis

Benjamin Michael Braithwaite – cpq608@alumni.ku.dk
Malte Stær Nissen – tgq958@alumni.ku.dk

A study of higher order image descriptors



Supervisors: Kim Steenstup Pedersen and Sune Darkner

August 18, 2014

Abstract

Descriptors are a tool used in computer vision for describing local structure in images. In this project we study the literature of descriptors, and develop our own descriptors based on this study. Our descriptors consist of localized kernel-based histograms, each describing parts of a region. Gradient orientation and shape index are used as histogram domains.

The descriptors are optimized and evaluated on two problems: image correspondence, where we test their ability to handle various image transformations, and pedestrian detection, where we test their ability to generalize the class of pedestrians. We compare the performance on each application against the SIFT and HOG descriptors. Extensive datasets of real world images are used.

Our descriptors perform marginally better than SIFT for image correspondence, and slightly worse than HOG for pedestrian detection. The addition of shape index to gradient orientation histograms is shown to be useful for pedestrian detection, but unnecessary for image correspondence.

Acknowledgements

We would like to express our gratitude towards our supervisors Kim Steenstup Pedersen and Sune Darkner for their help and level of enthusiasm throughout this master's thesis. Furthermore we would like to thank the image group at DIKU for supplying us with access to the computational power needed in order to complete our work.

Contents

1 Introduction **1**

2 Related work **3**
2.1 Gradient-based descriptors 3
2.2 Higher order differential descriptors 5
2.3 Image correspondence 5
2.4 Object detection 6

3 Methods **8**
3.1 Gaussian scale-space 8
3.2 Differential structure 9
3.2.1 Gaussian derivatives 9
3.2.2 Gradient orientation and magnitude 12
3.2.3 Shape index and curvedness 12
3.3 Histograms 14
3.3.1 Aperture kernel functions 15
3.3.2 Periodic histograms 15
3.3.3 Renormalization 16
3.4 Invariants and robustness 17
3.4.1 Translation 17
3.4.2 Rotation 18
3.4.3 Reflection 18
3.4.4 Illumination 19
3.4.5 Scale 19
3.4.6 Perspective 19
3.5 Interest point detection 19
3.5.1 Multi-scale difference of Gaussians 20
3.6 Sliding window 22
3.7 Descriptors 22
3.7.1 SIFT 22
3.7.2 Opponent SIFT 23
3.7.3 HOG 23
3.8 Binary classification measures 24

3.9 Confidence intervals . . . 26

4 Proposed descriptor **28**
4.1 Bin value and magnitude functions . . . 29
4.1.1 Local magnitude normalization . . . 29
4.2 Cell aperture function . . . 31
4.2.1 Interest point cell layouts . . . 32
4.2.2 Uniform cell layouts . . . 33
4.3 Center aperture function . . . 34
4.4 Binning aperture function . . . 34
4.4.1 Histogram normalization . . . 36
4.5 Descriptor parameters . . . 36
4.6 Example . . . 37

5 Image correspondence **45**
5.1 Matching strategies . . . 45
5.2 Similarity measures . . . 46
5.3 Performance measures . . . 46
5.4 Dataset . . . 47
5.4.1 Evaluation . . . 51
5.4.2 Pitfalls and deficiencies . . . 52
5.5 Application of descriptors . . . 53
5.6 Experimental setup . . . 54
5.7 Example . . . 55
5.8 Parameter study . . . 55
5.9 Results . . . 64

6 Pedestrian detection **69**
6.1 Support vector machines . . . 69
6.2 Performance measures . . . 71
6.3 Dataset . . . 71
6.3.1 Pitfalls and deficiencies . . . 72
6.4 Application of descriptors . . . 72
6.5 Experimental setup . . . 74
6.6 Example . . . 74
6.7 Parameter study . . . 77
6.8 Results . . . 81

7 Discussion **85**
7.1 Image correspondence . . . 85
7.2 Pedestrian detection . . . 87
7.3 Differences between applications . . . 88
7.4 Future work . . . 89

8 Conclusion **91**

A Histogram renormalization **92**
A.1 Box kernel 92
A.2 Triangle kernel 92
A.3 Gaussian kernel 93

B Image transformations **94**
B.1 Rotation 94
B.1.1 Gradient orientation 95
B.1.2 Gradient magnitude 95
B.1.3 Shape index 96
B.1.4 Curvedness 96
B.2 Illumination 97

C Grid layouts **98**
C.1 Log-polar grids 98
C.2 Concentric log-polar grids 99

D Gaussian derivatives **101**

E Confidence intervals for image correspondence **102**

Chapter 1

Introduction

In the field of computer vision, researchers have for many years worked on programming computers to interpret real world images as humans do. We focus on two problems within this field: image correspondence [DAP11], where we search for corresponding points between two different images taken of the same object, and pedestrian detection [FMR08], where locations of pedestrians are identified. These objectives are illustrated in Figure 1.1. The approaches to these problems are often based on creating descriptors to represent regions of the images, and comparing these descriptors by some method depending on the application. Popular and successful descriptors include SIFT [Low04], GLOH [MS05], DAISY [TLF08], and HOG [DT05; Fel+10].

The strategy for choosing image regions depends on the application. For pedestrian detection a systematic search across the image is needed, so we use a sliding window approach. For image correspondence we only want to describe the most distinctive regions, and hence we instead use an interest

(a) Image correspondence **(b)** Pedestrian detection

Figure 1.1: Illustrations of the objectives for the two chosen applications. In (a) we look for corresponding points between two images of the same object. In (b) we look for pedestrians.

point detector. Commonly used are the Harris corner detector, Hessian based detectors, the Difference of Gaussians (DoG) blob detector, and the Maximally Stable Extremal Regions (MSER) detector [ADP12; DAP11].

An important part of constructing descriptors is to make them invariant to image transformations such as rotation, translation, illumination, scale, perspective, and noise. This causes real world objects to be described similarly despite being captured under different conditions. The choice of invariant properties depends on the application of the descriptor. For example rotation invariance is desired for texture description but not necessarily for recognition of pedestrians.

The overall goal of this project is to study whether higher order information is able to improve descriptor performance. We will study the field of descriptors and based on this create our own, which we compare against state of the art. Like SIFT and HOG, our descriptors are based on derivative scale-space images, which describe differential structure in images at multiple scales. From these we compute histograms of gradient orientation and/or shape index [KD92]. The descriptor consists of histograms computed for a grid of cells over a region. To account for local illumination changes, we propose a pixel-wise normalization scheme of the derivative scale-space images.

The report is structured as follows: First we conduct a literature study of descriptors and their applications. Next we describe various methods used within the field of descriptors, which we need throughout the report. Hereafter we propose our descriptor framework, and apply our descriptors to the image correspondence and pedestrian detection problems. Then we discuss the application results, and finally we conclude upon our work.

Throughout this report we move non-essential details of derivations to appendices, and note when this is done.

Chapter 2

Related work

Descriptors have been constructed based on different requirements with respect to the use of storage and computations. Heinly, Dunn, and Frahm [HDF12] have described the overall field of descriptors using these two variables resulting in the following taxonomy: real value parametrization (high storage and computation), patch-based (high storage and low computation), binarized (low storage and high computation), and binary (low storage and computation). We are mainly interested in the best performance, and hence we will focus on the class of real value parametrization descriptors. This class of descriptors was revolutionized by Lowe [Low04], who inspired many later descriptors. We will summarize the most successful of these. The descriptors are divided into two categories: gradient-based and higher order descriptors. Since we have chosen the two applications: image correspondence and pedestrian detection, which is a related to the more general object detection problem, we briefly summarize the work within these problems.

2.1 Gradient-based descriptors

Gradients of an image are often used to describe the regions in images, since they describe the change of image intensities. The *scale-invariant feature transform* (SIFT) descriptor [Low04] is the most popular descriptor based on this approach. It works by dividing image regions around interest points into square cells and constructing a histogram of gradient orientations for the pixels in each cell. The interest points are detected using a multi-scale DoG detector. By computing the gradient orientations according to the detection scale, the descriptor becomes scale invariant. The descriptor is also rotation invariant as the cell grid is rotated according to the dominating gradient orientation. Finally the descriptor achieves illumination invariance by normalizing the histograms. Van de Sande, Gevers, and Snoek [VGS10] succesfully extended the SIFT descriptor into the *Opponent SIFT* descriptor, which computes and concatenates the SIFT descriptor in each of the

three channels of the opponent color space.

PCA-SIFT [KS04] is also based on combining oriented gradients within a region, but instead of dividing the feature into cell histograms, it uses *principal component analysis* (PCA) to reduce the dimensionality of the combined gradients. PCA computes the most significant linearly independent dimensions of a dataset, and discards the rest. This means that while SIFT is a general purpose descriptor, PCA-SIFT must first be trained on specific data. In order to obtain good results a large and diverse training dataset of images is required. The dimensionality of the PCA-SIFT descriptor is substantially lower than SIFT and the other following gradient-based descriptors.

The *gradient location and orientation histogram* (GLOH) descriptor [MS05] is a different extension of the SIFT descriptor. It differs from SIFT by using a log-polar grid of ring segment cells instead of a grid of rectangular cells. The dimensionality of the descriptor is like PCA-SIFT also reduced using PCA, but the analysis is performed on the gradient orientation histograms rather than the raw gradients.

The DAISY descriptor [TLF08] is a descriptor originally developed for dense wide-baseline matching, and it is therefore developed to create a descriptor of each pixel in an image very efficiently. The descriptor is created from Gaussian directional derivative convolutions of the image computed at points located in a circular grid resembling a daisy. The directional derivatives are similar to the histogram bins of SIFT, but they can be computed much faster. Further refinements of the algorithm and experiments with the actual layout of the daisy formation have been examined by Winder, Hua, and Brown [WHB09], who report that the DAISY descriptor performs better than SIFT when applied to the image correspondence problem.

The *Histograms of Oriented Gradient* (HOG) descriptor is another SIFT-based descriptor, proposed by Dalal and Triggs [DT05] for the purpose of performing pedestrian detection. Rather than using a detector, a sliding window approach is used for searching an image for pedestrians. For each window one HOG descriptor is constructed and a binary classification of the descriptor is performed. The window is divided into a grid of cells, in which histograms of oriented gradients are computed. Adjacent cells are then grouped together into blocks, which are normalized to accommodate for local illumination changes. The final HOG descriptor is a concatenation of the histograms of the window cells. Deformable part models [FMR08] improve upon the HOG descriptor by searching for parts at finer scales for each detected pedestrian. Furthermore Felzenszwalb et al. [Fel+10] extended the HOG descriptor to compute both directed and undirected gradients as well as a sparse representation of the block normalization.

2.2 Higher order differential descriptors

Another approach to descriptor design is to use higher order differential information. We use the term *local k-jet* to refer to a vector consisting of the derivatives up to order k at some point.

Crosier and Griffin [CG10] base their texture representation, which is just a descriptor but used for textures, on the local 2-jet. They partition the jet space into six Basic Image Features (BIFs) and compute these across a region of pixels at four different scales. The chosen BIFs are distinct texture elements such as dark spots and bright lines. Rather than computing the distribution of BIFs at each scale, the four BIFs at each point are combined into a BIF-column, and a histogram over all possible BIF-columns is computed. The descriptors are used for texture classification.

Larsen et al. [Lar+12] have had success using local 4-jets. Their $\mathcal{J}_4$-grid2 descriptor is computed from local 4-jets at four points spread out across a pixel region. A whitening process is used on the jet coefficients to scale normalize and decorrelate the descriptors allowing for Euclidean distance as a distance measure. The descriptor was evaluated against state of the art on the image correspondence problem and performed favourably, despite the simplicity of the descriptor.

Another recent higher order descriptor is the galaxy descriptor from Pedersen et al. [Ped+13], which is used to predict star-formation rate from galaxy texture. The descriptor consists of multi-scale histograms of gradient orientation as well as shape index, which is a simple 1D representation of second order differential information. The histograms are computed over a single region at eight scale levels.

2.3 Image correspondence

The image correspondence problem has been frequently used throughout the literature for testing invariants and robustness of descriptors [Low04; MS05; KS04; Lar+12; Cui+09; TW09]. This is due to the simplicity of the problem: Take two images of the same scene with a pre-defined transformation, and find correspondences between the two images. The problem can be solved by searching for similar descriptors across the two images. By comparing these matches against a ground truth, we can measure the descriptor's ability to create robust descriptors under transformations. In practice the image correspondence problem can be used for baseline stereo matching, where correspondences are found with the goal of creating a 3D reconstruction of a scene. Tola, Lepetit, and Fua [TLF08] solve this problem by developing and using the DAISY descriptor. We will however not go further into this problem, but instead we use the base image correspondence problem for descriptor evaluation.

Mikolajczyk and Schmid [MS05] evaluate 10 descriptors including SIFT, GLOH, and PCA-SIFT on the image correspondence problem. Two descriptors are classified as a match by some matching strategy based on the distance between them. The result depends on a threshold t, which is varied to obtain a performance curve. Matching regions having an overlap error below 0.5 under the given image transformation (a homography) are classified as correct matches. A curve showing the *recall* as a function of the $1 - precision$ is used as performance measure. Three different matching strategies were tested, which we will describe in detail in Section 5.1. Mikolajczyk and Schmid [MS05] conclude that GLOH performs best, closely followed by SIFT, and that SIFT obtains best results by matching based on nearest neighbour distance ratio.

Dahl, Aanæs, and Pedersen [DAP11] test the SIFT and DAISY descriptors with 7 state of the art detectors to find the best detector-descriptor combination. This is done by having a dataset with various image transformations and known 3D geometry, and hence having a known ground truth for patch correspondences. Compared to Mikolajczyk and Schmid [MS05], the dataset is substantially larger and more varied, which suggests that their results are more generalized. For this study only the nearest neighbour distance ratio is used as matching strategy. The Receiver Operating Characteristics (ROC) curve is computed for varying ratio-thresholds and finally the area under the curve (AUC) is computed. The average AUC over all the scenes of the dataset is used as performance measure. The ROC and its AUC are described further in Section 5.3. Dahl, Aanæs, and Pedersen [DAP11] conclude that SIFT and DAISY perform best when being paired with DoG or MSER detectors.

2.4 Object detection

As mentioned above, pedestrian detection is related to the more general object detection problem, and hence the approaches to object detection can be used for pedestrian detection as well.

Object detection has been conducted in various ways throughout the literature. Since we are interested in comparing our descriptor performance against state of the art descriptors, we will not be developing a complex object detection system, but rather stay with a simple one, for which we can change the descriptor. Therefore we limit this part of our literature study to simple object detection algorithms. As mentioned in Section 2.1 the HOG descriptor [DT05] was developed for pedestrian detection using a binary classification of windows across the image. An SVM is trained on the raw HOG descriptors and used for determining the outcome of the binary classification. Felzenszwalb, McAllester, and Ramanan [FMR08] improved this approach by expanding the HOG algorithm with a parts-model.

Using a sliding window for object detection reduces the problem to image classification. A common approach to image classification is the bag-of-visual-words (BoVW) strategy, which e.g. Csurka et al. [Csu+04] use in their *Bags of Keypoints* method for visual categorization. The BoVW strategy's training step is called codebook generation. Firstly, descriptors are computed for a training set of images. Secondly, the descriptors are clustered into k cluster centers called the vocabulary. Thirdly, the descriptors are projected to their closest match in the vocabulary, and a histogram (bag) of words is created for each image. Finally a classifier is trained on the bags-of-visual-words and their corresponding class. Prediction of an image is performed by computing the bags-of-visual-words (like training step 3) for the image and predicting the class using the trained classifier.

Fisher Vectors [Sán+13] are an alternative to the raw descriptor and BoVW classifiers. A Gaussian Mixture Model (GMM) with a diagonal covariance matrix is assumed for the distribution of descriptors of an image. Various statistics of the GMM are computed, from which we compute the Fisher vector signature of each image's descriptors and apply power (signed square-rooting) and L_2 normalizations. Sánchez et al. [Sán+13] compares the Fisher vectors against a BoVW histogram approach using a linear SVM and gain large performance improvements.

Chapter 3

Methods

In this chapter we go through basic computer vision theory and methods that we have found relevant throughout our work. We start out by explaining various components of descriptors such as Gaussian scale-space, differential structure, and histograms, as well as invariants and robustness of descriptors. Hereafter we describe interest point detection and the sliding window technique, which we will be using in our respective applications. This is followed by a short description of the two descriptors, SIFT and HOG, we will be comparing our descriptors against. Finally we explain two binary classification measures and explain the confidence intervals, which we will be using to access the difference in performance between our descriptors and SIFT on the image correspondence problem.

3.1 Gaussian scale-space

When taking pictures of the real world, the resulting images are of various sizes and resolutions, and include objects at different distances. This causes details to be present in the images at various scales. Using the Gaussian scale-space representation [Gri97; VH00] described in this section, we are able to create a multi-scale representation of the details in an image. The representation consists of a set of smoothed versions of the original image, obtained by applying Gaussian filters G of various sizes σ corresponding to the image scales. When an image is smoothed by a filter at scale σ, the details smaller than that scale are removed. For a point (x, y), the Gaussian filter at scale σ is defined as

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right) \tag{3.1}$$

and the corresponding scale-space image $L(x, y; \sigma)$ is defined as

$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y), \qquad \sigma > 0 \tag{3.2}$$

where $I(x, y)$ is the intensity of the original image I in (x, y) and $*$ denotes the convolution operator defined as

$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1, \tau_2) \cdot g(x - \tau_1, y - \tau_2) \, \mathrm{d}\tau_1 \, \mathrm{d}\tau_2$$

for the two functions f and g [GW08, p. 345].

Note that while these scale-space functions are defined on an infinite continuum, in practice we select a finite set of scales for which we wish to retrieve structural information in an image.

Figure 3.1 shows an example of a finite scale-space representation of an image using the five scales $\sigma = 1, 2, 4, 8, 16$. At smaller scales ($\sigma = 1, 2$) we are still able to see detailed structure in the gates, pavement, car, hedge, etc. When increasing the scale to $\sigma = 4$, the small details are smoothed out, and only the larger details of the mentioned objects are noticeable. At $\sigma = 8, 16$ the details are completely smoothed out, and only the larger overall structures are visible. This example clearly shows the power of the scale-space representation being able to model both small details as well as larger structures.

3.2 Differential structure

A typical approach to describing structure in an image is to compute local derivatives. We consider first and second order derivatives in the form of gradients and shape index. Real world images are usually filled with small-grained noise, which is why we often prefer to combine the derivative calculations with a smoothing of the image using a Gaussian filter. We first explain the details of this step, followed by a description of how to compute gradients and shape index.

3.2.1 Gaussian derivatives

The *Gaussian derivatives* $G_{x^m y^n}$, where m and n are the order of derivations with respect to x and y, are used for obtaining structural information from an image. Figure 3.2 illustrates the Gaussian derivatives up to second order.

Based on the Gaussian derivatives we define the Gaussian derivative scale-space images $L_{x^m y^n}$:

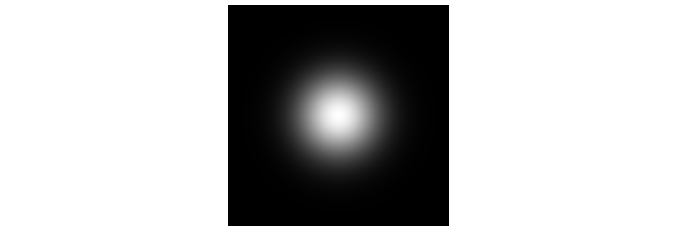
$$L_{x^m y^n}(x, y; \sigma) = G_{x^m y^n}(x, y; \sigma) * I(x, y) \tag{3.3}$$

The complexity of the convolution is not irrelevant when calculating scale-spaces. Given a filter f with k elements and an image I with l entries, the complexity of a convolution is $\mathcal{O}(kl)$ since we perform k multiplications and $k-1$ summations for each of the l elements. Here we have assumed that the

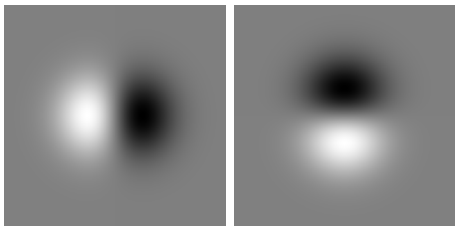
(a) Original **(b)** $\sigma = 1$ **(c)** $\sigma = 2$

(d) $\sigma = 4$ **(e)** $\sigma = 8$ **(f)** $\sigma = 16$

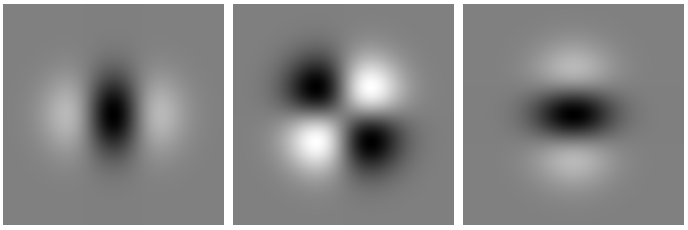
Figure 3.1: Example of a finite scale-space representation with scales $\sigma = 1, 2, 4, 8, 16$ of an image from the INRIA dataset (Section 6.3).



G



G_x G_y



G_{xx} G_{xy} G_{yy}

Figure 3.2: Gaussian derivatives up to second order.

boundaries of the image are padded to allow a full filter computation for all l entries. In theory the Gaussian function has infinite support causing the Gaussian filter to be infeasibly large. In practice one often uses the 3-sigma rule to determine the support radius of the Gaussian filter and hence the computational load of the convolution. The rule states that approximately 99.7% of the area under a 1D Gaussian function lies within 3σ of the center of the function, and hence using 3σ as the Gaussian function support radius suffices.

When computing a single Gaussian derivative image, a typical approach is to solve $G_{x^m y^n}$ analytically and sample the 2D filter from the analytical solution. Given a square filter of width and height 6σ we get $k = (6\sigma)^2$ filter elements. By solving G_{x^m} and G_{y^n} analytically, sampling two 1D filters of size 6σ and performing two chained convolutions of I, we get the same resulting image but at the cost of two significantly smaller filter convolutions. The same strategy can of course be applied to other separable 2D filters such as the ordinary Gaussian smoothing filter.

Since the Gaussian derivative scale-spaces are computed on a set of increasing scales, we need to compute all the needed derivatives for each scale. This give us a large amount of convolutions of the same image using various sizes Gaussian filters, which is computationally inefficient. Given these circumstances, we now describe an alternative and computationally cheaper way of computing the Gaussian derivative scale-spaces. The idea is to compute one smoothing per scale, and then calculate all the derivatives at that scale using finite differences filters. We first define the central differences filter f:

$$
\begin{aligned}
I_x = \frac{\partial I}{\partial x} &\approx f_x * I = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * I \\
I_y = \frac{\partial I}{\partial y} &\approx f_y * I = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^{\mathrm{T}} * I
\end{aligned}
$$

The f_{x^m} filter is made by convolving m f_x-filters in a chain and the f_{y^n}-filter is constructed similarly. The combined derivative is calculated by following the alternative approach above, where we chain two 1D convolutions of G together:

$$
\begin{aligned}
L_{x^m y^n}(x, y; \sigma) &= G_{x^m y^n}(x, y; \sigma) * I(x, y) \\
&\approx f_{x^m} * (f_{y^n} * (G(x, y; \sigma) * I(x, y))) \\
&= f_{x^m} * (f_{y^n} * L(x, y; \sigma))
\end{aligned}
$$

Since we have a nice set of increasing scales for our scale-space images, we are able to compute $L(x, y, \sigma_2)$ by using the previous scale-space $L(x, y, \sigma_1)$ where $\sigma_2 > \sigma_1$ as described by Tola, Lepetit, and Fua [TLF08]. The idea is to split up the smoothing into two operations and then use the associativity

of convolution to use the previous scale-space image for the new smoothing:

$$\begin{aligned} L(x, y; \sigma_2) &= G(x, y; \sigma_2) * I(x, y) \\ &= (G(x, y; \sigma_\Delta) * G(x, y; \sigma_1)) * I(x, y), \qquad \sigma_2 > \sigma_1 \\ &= G(x, y; \sigma_\Delta) * L(x, y; \sigma_1) \\ \sigma_\Delta &= \sqrt{\sigma_2^2 - \sigma_1^2} \end{aligned}$$

These equations give us a faster way of computing the Gaussian derivative scale-spaces than the original formulation.

3.2.2 Gradient orientation and magnitude

The *gradient orientation* $\Theta \in [-\pi, \pi]$ is the orientation of the local first order derivatives. Θ is independent of the size of the derivatives, which is given by the *gradient magnitude* M. Θ and M are defined in terms of the image derivatives as

$$\begin{aligned} \Theta &= \text{atan2}(L_y, L_x) \\ M &= \sqrt{L_x^2 + L_y^2} \end{aligned}$$

where atan2 is the arctangent function spanning the whole range of angles $[-\pi, \pi]$. Figure 3.3 illustrates the relationship between the gradient orientation Θ, gradient magnitude M, and x- and y-derivatives. Note that Θ is the directed gradient. Alternatively we could compute undirected gradients by using the $\tan^{-1}$ function such as Dalal and Triggs [DT05] do. This is however not done in our work, since the directed gradients are more descriptive.

3.2.3 Shape index and curvedness

The *shape index* $S \in [-1, 1]$ [KD92] is a scalar characterization of local second order derivatives. Like Θ and M, S is independent of the size of the derivatives, which is given by the *curvedness* C. Additionally the shape index is independent of orientation.

S is defined by the orientation of the eigenvalues κ_1 and κ_2, also called the principal curvatures, of the local Hessian matrix $\nabla^2 L$. It can also be expressed by the local image derivatives:

$$S = \frac{2}{\pi} \tan^{-1} \left(\frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2} \right) \tag{3.4}$$

$$= \frac{2}{\pi} \tan^{-1} \left(\frac{-L_{xx} - L_{yy}}{\sqrt{4L_{xy}^2 + (L_{xx} - L_{yy})^2}} \right) \tag{3.5}$$

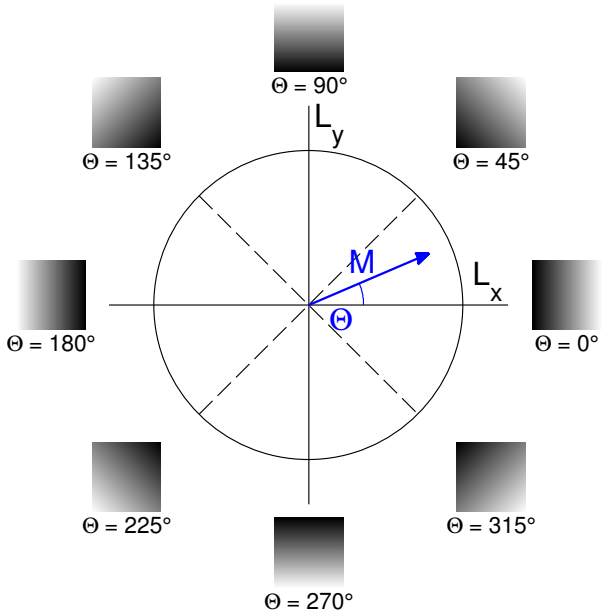


Figure 3.3: Gradient orientation and magnitude shown in the coordinate system (L_x, L_y) of first order image derivatives.

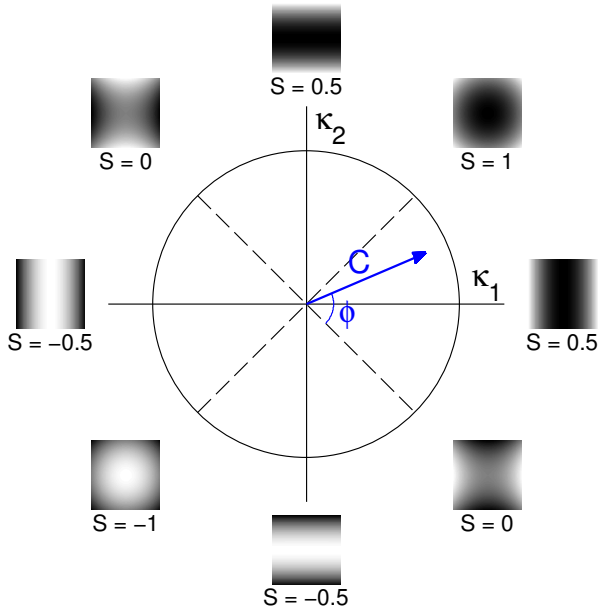


Figure 3.4: Shape index and curvedness shown in the coordinate system (κ_1, κ_2) of principal curvatures. The blue vector illustrates a shape index $S = \frac{2}{\pi}\phi$ and curvedness C, where ϕ is the signed angle between the vector and the line $\kappa_2 = -\kappa_1$. Note that the order of κ_1 and κ_2 does not matter, as shape index is independent of the orientation of the local geometry.

C is defined by the size of κ_1 and κ_2, and can also be expressed by the image derivatives:

$$C = \sqrt{\frac{\kappa_1^2 + \kappa_2^2}{2}} \tag{3.6}$$

$$= \frac{1}{\sqrt{2}} \sqrt{L_{xx}^2 + 2L_{xy}^2 + L_{yy}^2} \tag{3.7}$$

Figure 3.4 illustrates the relationship between the principal curvatures, shape index and curvedness, as well as the geometry of various shape indices. As seen in the figure, S ranges over light blobs ($S = -1$), ridges ($S = -0.5$), saddle points ($S = 0$), valleys ($S = 0.5$), and dark blobs ($S = 1$) with continuous deformations in between.

3.3 Histograms

Histograms play an essential role in SIFT-like descriptors, where they are used for aggregating gradient orientation and magnitude information over local areas. Traditional histograms have been constructed using hard boundaries between bins. These types of histograms are however neither robust to small interest point translation/detection errors or gradient orientations close to the bin boundaries. SIFT uses a trilinear interpolation of the gradient magnitude based on the radial distance between the gradient orientation and the bin centers to cope with gradients close to the bin boundaries. In order to improve upon the robustness of the histograms, we here describe a kernel-based histogram framework, which is able to produce both ordinary (hard bounded) as well as smooth histograms. The framework is based on locally orderless images (LOI) [Kv99] and Parzen windows [Par62].

The idea behind smooth histograms is to remove the sharp boundaries of ordinary histograms (bin boundaries). Given a position we wish to create a histogram for the surrounding local area/region of interest (ROI). This is done by letting all the values from the ROI contribute to all the bins of the histogram.

Given a bin value function f with a range of $[f_{\min}, f_{\max}]$, and a spatial ROI center $\mathbf{x}_0$, we wish to compute the histogram H value at bin center f_i. The bin value function defined as $f(\mathbf{x}, \sigma)$ corresponds to calculating the function f in position $\mathbf{x}$ at (inner) scale σ. In terms of images this could correspond to calculating the gradient orientation at a position $\mathbf{x}$ in a scale-space image at scale σ. We use the distance between the bin center f_i and $f(\mathbf{x}, \sigma)$ to compute the bin aperture function B using the (tonal) scale β as "width" of B. Finally we compute a spatial aperture function A of the distance between $\mathbf{x}_0$ and $\mathbf{x}$ using the (outer) scale α as the "width" of A. The histogram of the bin center f_i is defined from f, B, and A as

$$H(f_i; \mathbf{x}_0, \sigma, \beta, \alpha) = \int A(\mathbf{x}; \mathbf{x}_0, \alpha) B(f(\mathbf{x}; \sigma) - f_i; \beta) \, \mathrm{d}\mathbf{x} \tag{3.8}$$

By specifying the variables $\mathbf{x}_0,\ \sigma,\ \beta,\ \alpha$ in the functions A and B, we get the following shorthand notation for a histogram:

$$H(f_i) = \int A(\mathbf{x})B(\mathbf{x}; f_i, f)\,\mathrm{d}\mathbf{x} \tag{3.9}$$

The specifics of the histograms used in our work will be explained in Equation (4.1).

3.3.1 Aperture kernel functions

We consider three choices of kernel functions for A and B: box, triangle, and Gaussian. In 1D, the respective kernels are defined as:

$$Box(x;\beta) = \begin{cases} \frac{1}{2\beta}, & \text{if } |x| \leq \beta \\ 0, & \text{otherwise} \end{cases}$$

$$Tri(x;\beta) = \begin{cases} \frac{1}{2\beta}\left(1 - \frac{|x|}{2\beta}\right), & \text{if } |x| \leq 2\beta \\ 0, & \text{otherwise} \end{cases}$$

$$G(x;\beta) = \frac{1}{\sqrt{2\pi}\beta}\exp\left(-\frac{x^2}{2\beta^2}\right)$$

In practice we truncate these kernels to a range $|x| < r$ given by a support radius r. For box and triangle kernels, we naturally set $r = \beta$ and $r = 2\beta$ respectively. For the Gaussian kernel, which is always non-zero, we limit $r = 3\beta$. This range contains 99.7 percent of the total area as mentioned in Section 3.2.1.

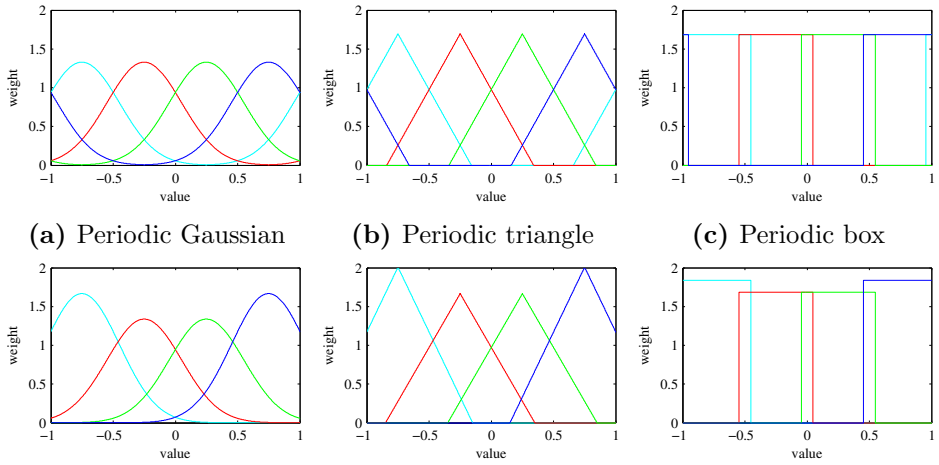
For bin aperture functions B we can use these kernels directly. The spatial aperture functions A are however defined on a 2D domain; this is handled by multiplying two of the same kernel functions together, e.g:

$$\begin{aligned} G(\mathbf{x};\alpha) &= G(x;\alpha)G(y;\alpha) \\ &= \frac{1}{2\pi\alpha^2}\exp\left(-\frac{x^2+y^2}{2\alpha^2}\right) \end{aligned}$$

where $\mathbf{x} = [x\ \ y]^T$. A single α for both dimensions is sufficient since we don't require a skew in the x- and y-axes for our purposes.

3.3.2 Periodic histograms

When our bin domain $[f_{\min}, f_{\max}]$ is periodic, there are multiple distances between any two points. Since we are computing distance bin aperture functions, we need to find the minimum distance d_{wrap} from each point value to each bin center in order to correctly compute the function values.



(a) Periodic Gaussian **(b)** Periodic triangle **(c)** Periodic box

(d) Renorm. Gaussian **(e)** Renorm. triangle **(f)** Renorm. box

Figure 3.5: Examples of periodic and renormalized 1D Gaussian, triangle, and box functions. Each figures contains four filters centered at values -0.75, -0.25, 0.25, and 0.75 and with $\beta = 0.3$.

This is done by a simple wrap-around using the known period of the domain as maximum distance:

$$d_{wrap} = \min(d, (f_{\max} - f_{\min}) - d)$$

where d is the distance $x - \mu$ from a point x to a bin center μ, and min is the function that chooses the lowest of its arguments.

Figures 3.5a to 3.5c show 1D examples of four periodic Gaussian, triangle and box kernels respectively on the interval $[-1, 1]$. Notice how the wrap-around affects the kernels at both ends of the periods.

3.3.3 Renormalization

When making histograms over a non-periodic domain, the weight function will be cropped by the histogram bounds $f_{\min}$ and $f_{\max}$. We therefore risk biasing the weights towards the central bins of the domain, since the cropped filters will have a smaller area. In order to avoid this, we view the bins as probability densities, and hence the area under each bin's weight function has to integrate to 1. This is done by renormalizing the functions based on their areas. Given a bin aperture function B and a bin center f_i, for which $f_{\min} \leq f_i \leq f_{\max}$ we calculate the bounds a and b when B is centered around f_i:

$$a = f_{\min} - f_i, \qquad b = f_{\max} - f_i,$$

We notice that $a \leq 0 \leq b$. The renormalized bin aperture function B_{renorm} is obtained by dividing B with its definite integral:

$$B_{\text{renorm}}\left(f(\mathbf{x};\sigma) - f_i;\beta\right) = \frac{B\left(f(\mathbf{x};\sigma) - f_i;\beta\right)}{\int_a^b B\left(z;\beta\right)\,\mathrm{d}z}$$

Notice how the bounds a and b of the integral are offset by the bin center f_i. The definite integrals of our chosen kernel functions are computed in Appendix A.

Figures 3.5d to 3.5f show 1D examples of four renormalized Gaussian, triangle, and box filters respectively on the interval $[-1, 1]$. Notice how the two weight functions of the outer-most bins have higher peaks than the central bins in order to integrate to 1 when being cut off at the interval bounds. For periodic histograms the renormalization step is excluded since no cut off of the filters occurs.

3.4 Invariants and robustness

When describing a local region in real world images, we experience a lot of transformations of the actual physical objects. We are however interested in being able to describe the object without any (or a predefined set) of these transformations applied, such that we get similar descriptions of the same object from images with different transformations.

Apart from invariance to various transformations we also want to be robust to small changes in some of the transformations. This robustness should give us nearly equal descriptors despite having small errors in our estimation of the transformations. In this section we will go through a list of transformations that we wish for our descriptor to be invariant and/or robust to.

3.4.1 Translation

Translation occurs when images are taken from different positions or objects are moved, and hence the region is translated in the image plane, but without affecting the perspective. Translation invariance is often achieved by combining a descriptor with an interest point detector or a sliding window, which tells the descriptor where all the regions of the image are located with respect to both the position and scale. Translation invariance is almost always desired since even tiny vibrations at the time of capture can cause an object to change position in an image.

An interest point detector or a sliding window is only able to give an estimate of positions of interest points based on the detail available in the pixels of the image. We therefore also wish to have a descriptor, which is robust to small changes in interest point positions allowing for a small error in detection point without having a large impact on the descriptor.

Measure	Rotation	Illumination
Θ	$\Theta - \theta$	Θ
M	M	aM
S	S	S
C	C	aC

Table 3.1: The transformation of various measures by rotation and illumination. The derivations are included in appendix B. The rotation column corresponds to an image, which is rotated by θ, and the illumination column corresponds to an affine transformation $aI + b$ of each pixel in image I.

3.4.2 Rotation

Rotation naturally occurs when either the camera or the object in question is rotated. Even though a precise position is located, the outcome of descriptors with spatial pooling will be affected by rotation, if no effort is done to achieve rotational invariance. For SIFT-like descriptors the general approach to achieving rotational invariance is to estimate the direction of the feature and rotate the descriptor grid before computing the descriptor. Rotation invariance is often desired, but it is not necessary for the datasets we use for our applications (Sections 5.4 and 6.3). Thus we refrain from pursuing rotation invariance for our descriptors.

Looking at the differential measures $\Theta, M,\ S$, and C, we are able to derive the resulting measure under rotation by the angle θ. We have left out the details of the derivations here, but they can be found in Appendix B.1. The final derived results are shown in Table 3.1. We see that both $M,\ S$, and C are rotational invariant whereas Θ is rotated by the transformation angle θ.

3.4.3 Reflection

Reflection occurs when either an object is flipped along some axis or an entire image is flipped. An example could be two sideshot images of a car driving in opposite directions, since a car typically looks the same from either side. Reflection is typically not handled directly in the descriptor but rather in the applications of the descriptor. An example of this is the pedestrian detection application, where [DT05] make sure to include both the original positive pedestrian images as well as their horizontally flipped versions in order to obtain a reflection invariant pedestrian detector.

3.4.4 Illumination

Illumination transformations both occur as global as well as local transformations of light. The global transformation model the overall diffuse light level, which is visible on e.g. a sunny and a cloudy day. The local transformations model the local differences in light such as e.g. cast shadows and directed light.

By assuming the image $\tilde{I} = aI + b$ is an affine illumination transformation of the underlying image I with no light transformations, we are able to derive the effect of the transformation on the differential measures Θ, M, S, and C. Once again we have left out the details here, which can be found in Appendix B.2. The final results can be seen in Table 3.1. We see that Θ and S are illumination invariant whereas M and C are multiplied by the linear constant a of the affine transformation. These coefficients are typically estimated and corrected for by using some either local or global normalization scheme. Note that non-linear illumination changes can also occur due to e.g. reflections from 3D objects. These changes are not handled by the normalizations mentioned here.

3.4.5 Scale

The scale of objects in images vary depending on multiple factors as described in Section 3.1. In order to achieve scale invariance, descriptors are typically defined relative to a feature scale. Depending on the application, the feature scale is typically defined by an interest point detector or a sliding window as described in Section 3.4.1.

3.4.6 Perspective

Perspective distortions and occlusions have major impact on images. Even small changes of camera position can distort objects and occlusions might occur. The perspective transformations are hard to estimate without prior knowledge of the scene and its attributes. Therefore only perspective robustness to a certain degree has been achieved in descriptors so far.

3.5 Interest point detection

As mentioned in Chapter 1 interest point detectors are used for finding distinctive points in images having various desired properties. These properties vary between blobs, corners, and edges depending on the detector used. Furthermore some detectors are multi-scale, meaning that they search through a scale-space pyramid in order to find interest points at multiple scales. Combining a multi-scale detector with a descriptor is a typical way of achieving scale invariant descriptors as mentioned in Section 3.4.5.

The focus of this report is to study higher order descriptors and hence we simply choose to use the multi-scale DoG blob detector from SIFT [Low04] described below.

3.5.1 Multi-scale difference of Gaussians

Recall from Section 3.1 that our scale-space representation removes image details below a given scale σ. The idea of the difference of Gaussians (DoG) blob detector is that by computing the difference of two scale-space images for σ_1 and σ_2, we also remove the details at a higher scale, leaving only details between σ_1 and σ_2. Formally we define

$$\begin{aligned}\Gamma(\mathbf{x};\sigma_1,\sigma_2) &= L(\mathbf{x};\sigma_2) - L(\mathbf{x};\sigma_1) \\ &= (G(\mathbf{x};\sigma_2) - G(\mathbf{x};\sigma_1)) * I(\mathbf{x})\end{aligned}$$

where $\sigma_2 > \sigma_1$. Minima and maxima of this function indicate dark and light blobs, respectively, at a scale between σ_1 and σ_2.

In order to create a multi-scale DoG detector, Γ is computed across adjacent pairs of scale-space images. In SIFT, the scales are created in octaves as illustrated in Figure 3.6. When we extract multi-scale interest points, we find extrema that are not just extrema in their respective DoG images, but for adjacent DoG images as well. In practice this is done by finding extrema in a 26-pixel neighbourhood. As seen in the illustration, each octave has a number of DoG images, all with the same image size.

The octave scales are defined as 2^n for $n = l, \ldots, h$, where l is the lowest exponent and h is the highest exponent needed for a desired range of octave scales. An octave is split into several scales s, for which we wish to extract interest points. This results in Gaussian images positioned at scales $2^{m/s}$ where $m = l, \ldots, s \cdot h$. The DoG images are however differences of these Gaussians, and hence the detection scales are offset in between the Gaussian scales at $2^{(m+0.5)/s}$. In practice we need to create $s+3$ Gaussian images of the same size for each octave. The first extra Gaussian image $(s+1)$ is needed in order to compute the s DoG images. The next two Gaussian images are needed for two additional DoG images surrounding the s initial DoG images, allowing for 26-pixel neighbourhood computations within the entire octave. Figure 3.6 illustrates this for $s = 2$.

To prevent extracting too many interest points, we can select only the most significant points by placing a threshold on the magnitude of Γ.

An example of DoG images is shown in Figure 3.7, which is based on the scale-space images from Figure 3.1. Note that the two different scales locate dark and light blobs of different sizes.

We use the VLFeat implementation of the multi-scale DoG detector Vedaldi and Fulkerson [VF08].

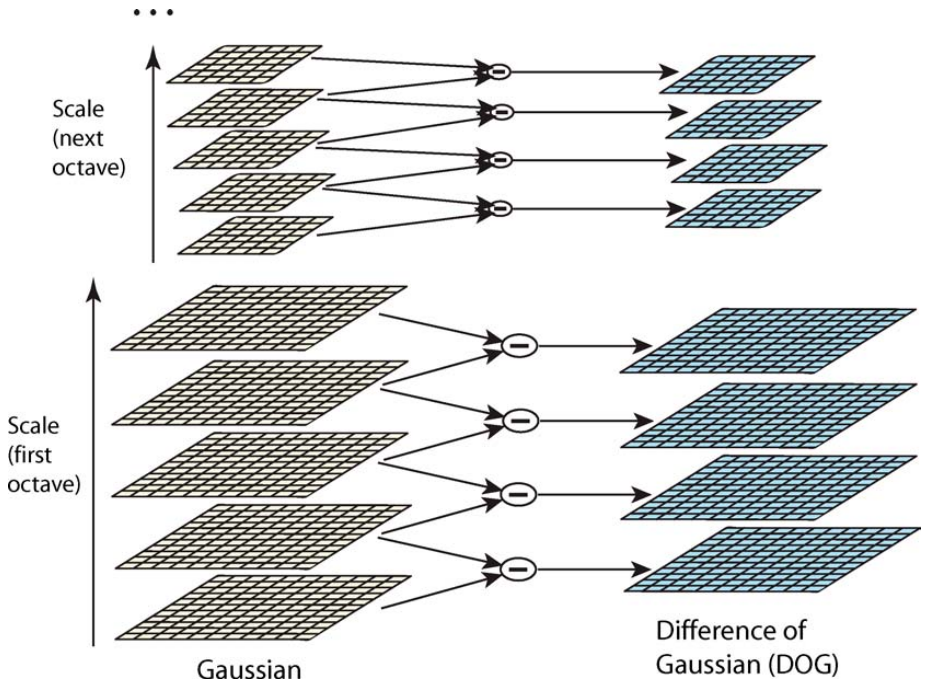
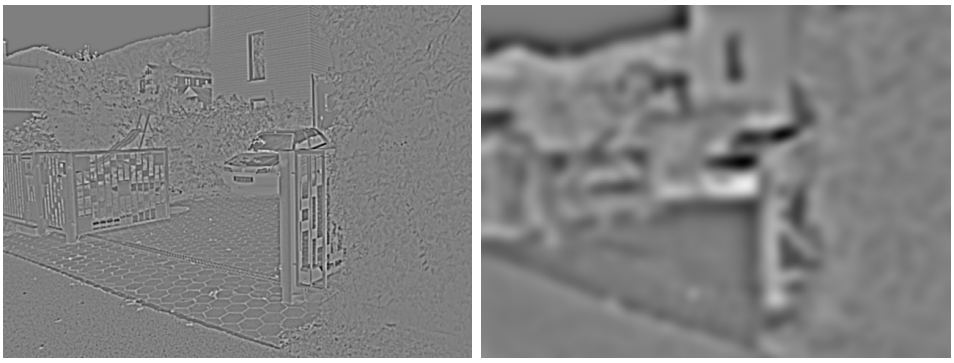


Figure 3.6: Illustration of how DoG images are computed by subtracting adjacent Gaussian scale-space images. The resulting images contain details between the two respective scales. This figure is taken from Lowe [Low04, figure 1,pp. 95].



(a) $\sigma_1 = 1, \sigma_2 = 2$ **(b)** $\sigma_1 = 8, \sigma_2 = 16$

Figure 3.7: Example of DoG images computed for the set of scale-space images from Figure 3.1 at two fine scales $\sigma_1 = 1, \sigma_2 = 2$ and two course scales $\sigma_1 = 8, \sigma_2 = 16$. Light and dark areas correspond to dark and light blobs in the respective scale intervals.

3.6 Sliding window

A sliding window is an alternative approach to selecting image regions for which to compute descriptors. Instead of centering regions around interest points, we systematically extract regions from varying positions and scales. Generally this is achieved by placing a square grid in each scale-space image. Regions are then extracted around each grid point at a size respective to each scale. We use a somewhat dense grid with a pre-defined stride, which causes a large overlap between regions. The sliding window method is used when we require an exhaustive search of an image, which is the case for pedestrian detection.

3.7 Descriptors

In our two applications we choose the SIFT and HOG descriptors to compare our descriptors against, since these are successful and widely used. These descriptors were introduced in the literature study, and in this section we will go into the details of how they work, and how we apply them to our applications.

3.7.1 SIFT

SIFT utilizes gradient information on grayscale images to compute distinctive interest point descriptors. SIFT was originally proposed in combination with the multi-scale DoG detector described in Section 3.5.1. After the DoG detection steps, the algorithm first determines the orientations of the interest point in question, in order to make the descriptor invariant to rotation. This is done by computing gradient magnitudes and orientations for a local area around the interest point in the associated scale-space image L. From the orientations a histogram of 36 bins is created, where each orientation is weighted by its magnitude and scaled by a Gaussian weight computed from the distance to the interest point. This Gaussian window has σ equal to 1.5 times the detection scale σ_0. A descriptor is created for the highest peak of the histogram as well as for each of the peaks that have a weight within 80% of the maximum histogram weight. Finally, the chosen orientations are refined by fitting of a parabola to the three points around each orientation. If rotation invariance is not needed for the application, this step can be skipped.

The local area surrounding the interest point is divided into a grid of 4×4 cells with the determined orientation, each spanning 4×4 pixels in the σ_0 scale-space image. For each of these cells a gradient orientation histogram with 8 bins is created. Each sample is weighted by its gradient magnitude. The magnitude is however subject to a trilinear interpolation between adjacent cell- and bin centers. Furthermore all magnitudes are

multiplied by a Gaussian window centered at the interest point and having σ half the width of the descriptor window.

Once the histograms are computed, the descriptor vector is normalized, all values thresholded to a maximum value of 0.2, and normalized to unit vector length again. The normalization is done to make the descriptor illumination invariant, and the thresholding is done to decrease the importance of gradients with very large magnitudes. The result is a descriptor of 16 histograms with 8 bins using a total of 128 dimensions. We use the VLFeat implementation of SIFT [VF08].

3.7.2 Opponent SIFT

The *Opponent SIFT* proposed by Van de Sande, Gevers, and Snoek [VGS10] is an extension of the original SIFT. Where SIFT is computed on the gradients of a grayscale image, the Opponent SIFT is computed on the opponent colour space:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

To compute the Opponent SIFT descriptor for a point in an RGB image, the opponent colour space is first computed. The SIFT descriptor is then computed for each of the three colour channels O_1, O_2, and O_3 and combined into a single descriptor. In total this descriptor is three times as big as the original SIFT and hence has 384 dimensions. Van de Sande, Gevers, and Snoek [VGS10] report that Opponent SIFT performs better than SIFT. We have verified that this holds for our descriptor as well in Section 5.8, and hence we use opponent colour space images for both our own descriptor and SIFT in our comparison.

3.7.3 HOG

The HOG descriptor is similar to the SIFT descriptor, but it describes a rectangular region of an image rather than a local area around an interest point. It has a number of parameters for various components of the descriptor; we will here describe the original HOG descriptor optimized for pedestrian detection.

Gradient orientation and magnitude is computed for the image by central differences. Unlike SIFT, the scale-space images are not smoothed, but only resized according to scale. Regions in these images are divided into 8×8 pixel cells, each for which a gradient orientation histogram weighted by gradient magnitude is created. Like SIFT, pixels contribute to multiple histogram bins by trilinear interpolation with respect to the cell and bin

centers. Though, HOG differs by using 9 histogram bins of unsigned gradient orientations.

Neighbouring cells are then grouped into overlapping 2×2 blocks. An additional weight is applied to each pixel in a block before assembling the histograms, based on a Gaussian window centered at the block center with $\sigma = 8$. The purpose of this is to reduce the importance of pixels near the edges of the block. Each block is then normalized to $L2$-norm unit length, followed by clipping values to 0.2 from SIFT. The resulting block normalized histograms are concatenated to make up the final HOG descriptor. Note that the block overlap causes each cell to be represented 4 times with different local normalizations. Dalal and Triggs [DT05] report that while this may seem redundant, it is critical to obtaining good performance. The block normalization scheme makes the descriptor robust against local illumination changes between cells.

As mentioned in the literature study, Felzenszwalb et al. [Fel+10] extended the HOG descriptor successfully. Their extended descriptor computes both directed and undirected gradients and deals with the normalization redundancy differently. The histogram bins for the 4 block normalizations are summed together to form a 27-dimensional vector. However, the information from these normalizations are conserved in 4 additional dimensions by summing across the 9 undirected bins. This variant achieves slightly better pedestrian detection results, and thus we will use it for our comparison. As mentioned by Dalal and Triggs [DT05], the HOG feature is computed on RGB images by computing the gradients of each colour channel and choosing the gradient with the highest magnitude in each position. This is reported to produce better results than for grayscale images using the original HOG. We have verified that this holds for our descriptor as well in Section 6.7, and hence we use RGB images for both our own descriptor and HOG in our comparison.

We also use the VLFeat implementation of the extended HOG, which is denoted the UoCTTI variant in VLFeat.

3.8 Binary classification measures

Binary classification is the problem of predicting which of two classes (conditions) an element belongs to: positive $(+1)$ or negative (-1). Formally, we consider elements $x \in X$ with respective condition labels $y \in \{-1, 1\}$. This is also called the ground truth or golden standard. Without the knowledge of y, we wish to assign elements $x \in X$ with a prediction or classification $\hat{y} \in \{-1, 1\}$, ideally equal to y.

Many binary classification methods do not directly output such a classification, but rather a more descriptive *classification score* $s \in \mathbb{R}$ for each $x \in X$, that denotes the confidence of a classification. A higher s means that

we find it more likely that x has the label $y = 1$. Thus as an intermediate step we can select a threshold $t \in \mathbb{R}$, and classify x with $\hat{y} = 1$ if $s \geq t$, or $\hat{y} = -1$ if $s < t$. Varying this threshold allows us to modify the ratio between elements classified as positive and negative.

To measure the performance of a binary classification method, we could simply compute the ratio of correctly classified elements. However, this measure is flawed if the distribution of elements is skewed toward one of the two conditions, or if it is more important to classify one of the conditions correctly than the other. Instead, we base our measures on the confusion matrix:

	Condition positive	Condition negative
Assigned positive	True Positive (TP)	False Negative (FN)
Assigned negative	False Positive (FP)	True Negative (TN)

Table 3.2: Confusion matrix for binary classification

The confusion matrix places classification outcomes into one of four cells, depending on both the condition and assigned label. We define TP, FP, FN, and TN to be the number of respective outcomes across a given set of elements. We also define the following three measures: *recall* or *true positive rate* (TPR), *fall-out* or *false positive rate* (FPR), and *precision* (P):

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$
$$P = \frac{TP}{FP + TP}$$

In other words, recall is the percentage of condition positives that we have classified correctly, fall-out is the percentage of condition negatives that we have classified incorrectly, and precision is the percentage of assigned positives that are indeed condition positives.

Recall that these measures depend on the threshold t used for classification. There are two common ways to visualize the effect of t: the ROC-curve defined as recall vs. fall-out and the PR-curve defined as precision vs. recall. They are constructed by varying t across $\mathbb{R}$ and plotting the measures as a curve in the respective spaces. A ROC-curve describes the ability to classify both condition positives and negatives correctly, whereas a PR-curve describes the relation between the "quality" of assigned positives and the "quantity" of assigned condition positives.

In order to get a performance measure independent of the choice of t, we can integrate the ROC- and PR-curves giving us the area under the

curve (AUC). We refer to these measures as the ROC AUC and PR AUC. The ROC AUC has the property of being the probability that a randomly chosen condition positive element has a higher classification score s than a randomly chosen condition negative element. The PR AUC is also called the average precision, as it is the average precision across all possible recall ratios. Figures 5.11a and 5.11b show examples of ROC and PR curves as well as their AUCs for an example of the image correspondence problem.

Note that PR-curves are used when the positive class is the main class of interest, or when a large skew towards one of the conditions of the classification is present as according to Davis and Goadrich [DG06]. Furthermore Davis and Goadrich [DG06] prove that an algorithm which optimizes ROC AUC doesn't necessarily optimize PR AUC, and hence optimization algorithms should utilize the final choice of evaluation measure to achieve optimal results. It is however also shown that given two curves, the first curve dominates the second in ROC space if and only if the first curve dominates the second in PR space [DG06, Theorem 3.2], and hence an algorithm which always beats the competitors in PR also beats the competitors in ROC. In this context the notion of a curve dominating another curve means that the second curve is beneath or equal to first (dominating) curve.

The choice between using the ROC- or PR-curves and their AUCs are discussed in the individual application sections in Chapters 5 and 6.

3.9 Confidence intervals

Assume we have two sets of samples X and Y from two normal distributions $\mathcal{N}(\mu_X, \sigma_X^2)$ and $\mathcal{N}(\mu_Y, \sigma_Y^2)$ with possibly unequal variances. We wish to estimate $\mu_X - \mu_Y$ by a $(1-\alpha)100\%$ confidence interval:

$$\bar{X} - \bar{Y} \pm t_{\alpha/2,v}\sqrt{\frac{s_X^2}{n} + \frac{s_Y^2}{m}}$$

where $\bar{X}$ and $\bar{Y}$ are the sample means of X and Y, s_X^2 and s_Y^2 are the unbiased estimated variances, n and m are the number of samples from the respective distributions, and $t_{\alpha/2,v}$ is the critical t-value of Student's t-distribution with v degrees of freedom. The t-value is computed numerically using the degrees of freedom estimated by:

$$v = \frac{\left(\frac{s_X^2}{n} + \frac{s_Y^2}{m}\right)^2}{\frac{\left(s_X^2/n\right)^2}{n-1} + \frac{\left(s_Y^2/m\right)^2}{m-1}}$$

α is usually set to 0.05, in which case we are 95% confident that the difference between the means of two distributions is in our confidence interval. If

0 is not contained in the confidence interval, we say that the two distributions are significantly different. However we cannot conclude that the two distributions are equal if 0 is contained in the confidence interval.

Chapter 4

Proposed descriptor

Given a region centered at a point $\mathbf{x}_0$ in an image at scale σ_0, we wish to effectively describe the region. Our proposed descriptor consists of weighted kernel-based histograms, explained in Section 3.3, computed for m *cells* arranged in a grid. We define the j'th cell histogram H_j as

$$H_j(f_i) = \int F(\mathbf{x})A_j(\mathbf{x})P(\mathbf{x})B(\mathbf{x}; f_i, f)\,\mathrm{d}\mathbf{x} \tag{4.1}$$

where

f_i for $i = 1 \dots n$ are bin centers,

f is a bin value function on the domain $[f_{\min}, f_{\max}]$,

F is a magnitude function,

A_j for $j = 1 \dots m$ are cell aperture functions,

P is a center aperture function, and

B is a binning aperture function.

We will use the above terminology throughout this chapter. Note that we have kept this definition similar to the galaxy descriptor [Ped+13], except we consider several histograms with their own aperture functions A_j and add the additional center aperture function P. The descriptor $\mathbf{d}$ is constructed by concatenating all n bin values of all m cell histograms:

$$\mathbf{d} = \Big((H_1(f_1) \cdots H_1(f_n)) \cdots (H_m(f_1) \cdots H_m(f_n))\Big)^{\mathrm{T}} \in \mathbb{R}^{nm} \tag{4.2}$$

The five functions above can be seen as the components that make up our descriptor, and we consider different choices for each. The functions depend on numerous additional parameters, which we define and describe throughout this chapter. Most of these parameters will be subject to a parameter

study for each of our applications in order to find the optimal descriptor for the specific application. The parameter studies are described in Sections 5.8 and 6.7. After describing the descriptor parameters, we show a detailed example of the descriptor being computed for an image.

4.1 Bin value and magnitude functions

The choice of bin value and magnitude functions are closely connected, as we wish the magnitude function to weight the histogram values according to their importance for the local image structure. By far the most common choice in literature [Low04; KS04; MS05; TLF08] is using gradient orientation Θ as value function and gradient magnitude M as magnitude function. Another good choice from the galaxy descriptor [Ped+13] is to use shape index S weighted by curvedness C. Finally we have experimented with the l, b, a re-parametrization of second order information weighted by 2-jet norm $\|\mathbf{j}\|$ [Gri07]. This was however unable to produce good results through initial testing, and we decided not to pursue the idea further.

Our strategy is thus to create a descriptor based on Θ and M (referred to as GO), one based on S and C (referred to as SI), as well as a concatenation of these descriptors (referred to as GO+SI). These functions are defined in Sections 3.2.2 and 3.2.3.

Recall that in order to make our descriptor scale invariant, we wish to compute the descriptor according to the scale σ_0 of its region center $\mathbf{x}_0$. This is achieved by computing a scale space pyramid of both value and magnitude images across all region scales. The values and magnitudes needed to compute the cell histograms can then be found by lookup in these scale space images. Before making this lookup, we propose a pre-processing step on the magnitude images to make our descriptor robust to local illumination changes.

4.1.1 Local magnitude normalization

In Section 3.4.4 we described the problem of handling illumination in images under the assumption of an affine illumination transformation $\tilde{I} = aI + b$. The block-normalization scheme from the HOG descriptor [DT05] is a way of locally (in blocks of 2×2 cells) estimating the affine coefficient a. We wish to make this estimation even more local by estimating a on a pixel-level. We propose a normalization scheme called pixel-wise normalization, which we apply to the magnitude function $F(\mathbf{x})$. The pixel-wise normalized magnitude at a point is defined as the magnitude relative to a local area, which we compute by convolution with a Gaussian filter:

$$F_{\text{norm}}(\mathbf{x}) = \frac{F(\mathbf{x})}{\int F(\mathbf{y}) G(\mathbf{y} - \mathbf{x}; \eta\sigma_0)\, \mathrm{d}\mathbf{y}} \tag{4.3}$$

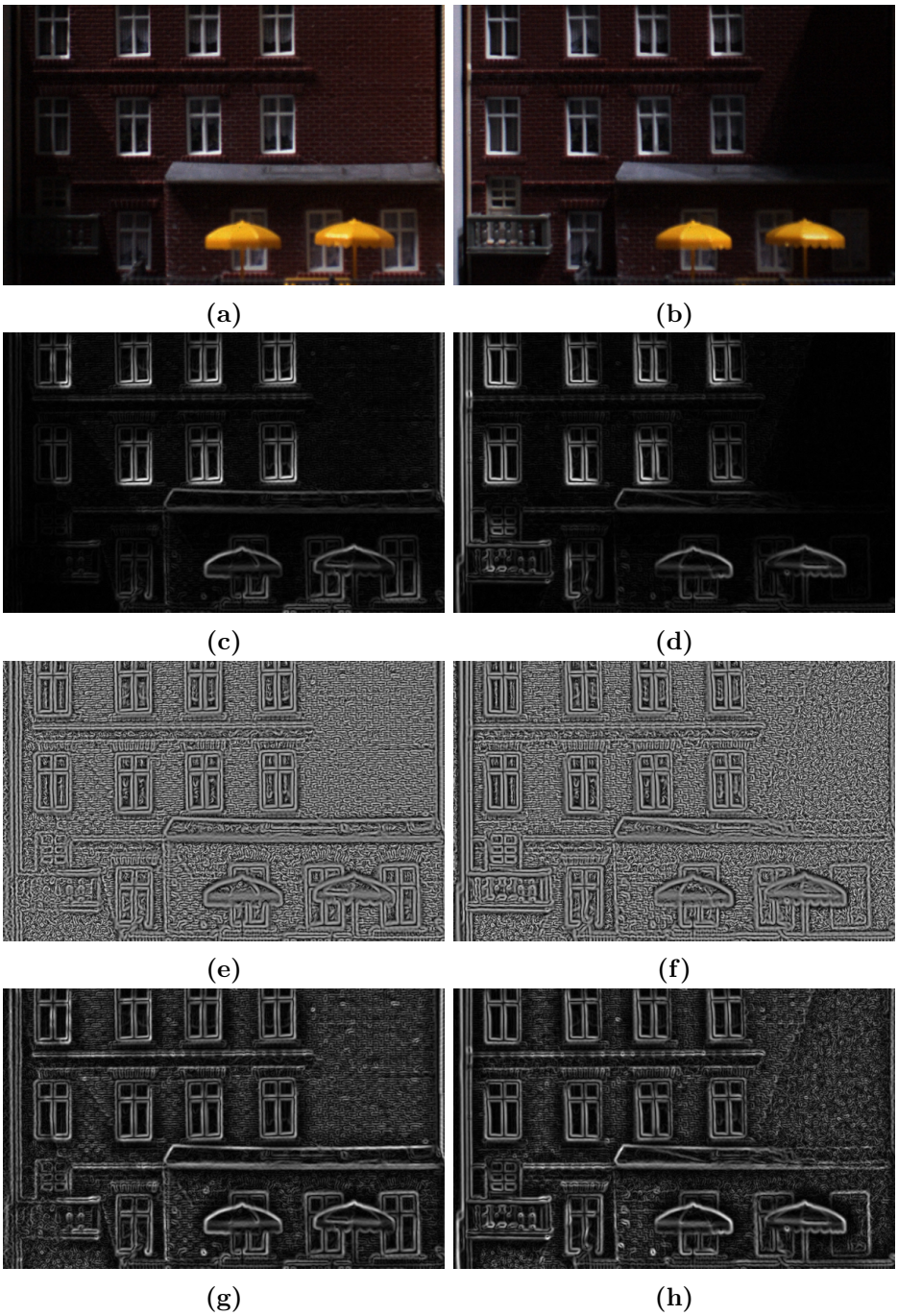


Figure 4.1: Images (a) and (b) show cut-outs of an image with artificial lighting from the left and right, respectively. Images (c) and (d) show gradient magnitudes of these images. Images (e) and (f) show pixel-wise normalized magnitudes for normalization scale $\eta = 2$. Images (g) and (h) show pixel-wise normalized magnitudes for $\eta = 10$.

where η is the normalization scale, defined relative to the region scale σ_0. The idea behind this scheme is that an image is locally affected by light changes according to the affine illumination model. In order to estimate a on a pixel-level, we use intensity information from the surrounding pixels. Using a Gaussian filter we are able to weight the surrounding intensities based on their spatial distance to the pixel in question. The rationale for this is, that pixels close to one another has a higher probability of being affected by equal light conditions than pixels further away from one another. By dividing the pixel value by the sum of all these weighted intensities (estimation of a), we remove the affine coefficient from the pixel.

Having applied the pixel-wise normalization to the magnitude scale space images we are done pre-processing the images and able to perform lookups of values and magnitudes for each region. Figure 4.1 shows two examples of images containing shadows, their gradient magnitudes at $\sigma_0 = 1$, their pixel-wise normalized magnitudes with $\eta = 2$, and their pixel-wise normalized magnitudes with $\eta = 10$. These types of shadows are the ones we will try to handle by using pixel-wise normalization. As seen in the gradient magnitude images, the shadows have a dampening effect on the gradient magnitude. By using a small η for the pixel-wise normalization, we see that a lot of the noise is enhanced but the shadows (especially the edges) are a lot less significant than for the normal gradient magnitudes. When using a larger η the noise is smoothed out and less significant. Furthermore there seem to be more of the structure highlighted from within the area affected by the shadows, but the shadow edges are more significant as well. The choice of optimal η is part of our parameter studies.

An alternative technique for handling local illumination is normalizing each cell histogram, which can be done instead of or alongside pixel-wise normalization. This was used successfully in the DAISY descriptor [TLF08]. We have tried various ways of incorporating this technique, but none were able to beat simple pixel-wise normalization through preliminary tests, and hence we have refrained from pursuing these strategies further.

4.2 Cell aperture function

The purpose of the cell aperture functions A_j is to weight points closer to each cell center higher in their respective cell histograms. We define a cell aperture function as

$$A_j(\mathbf{x}) = K(\mathbf{x} - (\mathbf{x}_0 + r\sigma_0\mathbf{c}_j); \alpha k_j r\sigma_0), \tag{4.4}$$

where

K is a kernel function as described in 3.3.1,

$\mathbf{c}_j$ is the relative cell center position for cell j,

α is the cell scale across all cells,

k_j is a cell scale specific to cell j, and

r is a radius parameter defined below according to the cell layout, scaling both cell scales and relative positions.

Note that the cell aperture function is simply a kernel function centered at each cell j for a specific region center $\mathbf{x}_0$ and scaled according to the layout radius r and region scale σ_0.

The cell aperture function is defined for all cells of the region. Having defined A_j we are now able to define the *cell layouts* that define the cell centers $\mathbf{c}_j$ for each region. The layouts depend on the application, and hence we split our proposed cell layouts into two categories: interest point and uniform cell layouts.

4.2.1 Interest point cell layouts

When describing an interest point the structure surrounding the interest point is of great importance. Interest points are usually found using a detector as described in Section 3.5. The innermost cells of the cell layouts are therefore spatially smaller than the cells further away, in order to capture the structural variations close to the interest point in greater detail. Given the nature of the layouts as structured grids, we refer to them as *grid types*.

Figure 4.2 shows examples of the six different grid types that we propose for describing the local area around an interest point: polar, polar central, log-polar, concentric polar, concentric polar central, and concentric log-polar. These layouts are inspired by the GLOH [MS05], Irregular SIFT [Cui+09], and DAISY [TLF08] layouts. Circular grids are chosen in order to retrieve an even amount of information around the interest point. The four polar grids use polar cell kernels, where points are converted from Cartesian coordinates into polar coordinates before applying the kernel. The two log-polar grids both use standard Cartesian kernels. The examples are shown for the *grid size* 8×2 meaning that each layout has 8 cells in each of their 2 rings. The two "central" and log-polar grids additionally have a single central cell. The two types of cells are illustrated in Figure 4.3 for a Gaussian kernel.

All these layouts are defined relative to the *grid radius* r and interest point scale σ_0. In other words the grid radius defines the span of the feature. The grid types are defined such that by default the 1 standard deviation curves of the outer-most cells in the feature touch the grid radius. Each ring consists of n cells dividing the ring equally in the angular direction. The cells have varying size depending on their position in the grid, and hence each cell j has its 1 standard deviation defined by a cell scale parameter k_j relative to r and σ_0. The following holds for each of the specific grid types:

For log-polar grid types the radial span of each ring is defined such that the 1 standard deviation of each cell j touches the 1 standard deviation curves of the neighbouring rings and neighbouring angles by default. This results in different default cell scales for the normal and concentric log-polar grid types, since the concentric log-polar grid type can pack the cells more densely without overlapping cells as seen in Figures 4.2c and 4.2f. For polar grid types the radial span is divided equally between the rings. The polar central grid types are furthermore having a central cell of half the radial span of the rings. The polar Gaussian cell aperture function combined with positioning of the cell centers $\mathbf{c}_j$ are shown in Figures 4.2a, 4.2b, 4.2d and 4.2e. See Appendix C for more information on how to compute the grid type cell positions.

As mentioned in the descriptions above, the grids are defined with default cell scales. These cell scales can be scaled both up and down by varying α to allow for additional or less overlap between the cells. This will however not change the positioning of the cell centers $\mathbf{c}_j$.

For the grid type examples in this section, we have simply chosen a grid size and a grid radius. These two variables have a major impact on the dimensionality and final outcome of the descriptor along with the cell kernel and its scale α, and hence these parameters are tuned in our parameter studies in order to get the optimal descriptor grid type and overlap.

4.2.2 Uniform cell layouts

The goal of a uniform cell layout is to describe a specified region of an image with no bias towards any reference point. These layouts are typically used in conjunction with a sliding window approach as described in Section 3.6. The shape of the region depends on the application and dataset. Regions are specified by a central point $\mathbf{x}_0$ with scale σ_0 similar to the interest point layouts. Like the interest point cell layouts, the uniform cell layouts are structured in grids and hence we likewise refer to these as grid types. Figure 4.4 shows examples of the two grid types that we propose for describing rectangular regions: square and triangle.

By default the 1 standard deviations of neighbouring cells are tangent to each other. The size of the 1 standard deviation, and hence also the spacing of the cells, is defined as the cell spacing r. Following the logic from the interest point grid types, the 1 standard deviations can be scaled to allow for additional or less overlap between the cells using the cell scale parameter α. The layouts are however only defined for a limited region, which causes problems when using the grid types in practice. Upscaling the overlap changes the support radius of each cell, which means that some cells span outside the defined region. In order to avoid this problem, we define another scaling parameter k_j for each cell j. For cells with support radius inside the defined region, we define $k_j = 1$. For the remaining cells we

define k_j such that the support radius of each cell is tangent to the closest region boundary. By increasing α, the boundary cells will therefore keep their original scale defined by r, whereas the cells further away from the boundary will have increased overlap.

4.3 Center aperture function

Similarly to A_j, the purpose of the center aperture function P is to weight points closer to the detected interest point higher, since they are of greater importance to the local structure. The function is thus omitted for uniform cell layout descriptors. We define it as

$$P(\mathbf{x}) = K(\mathbf{x} - \mathbf{x}_0; \rho r \sigma_0) \tag{4.5}$$

where K is a kernel function as described in 3.3.1, and ρ is the scale of the center aperture function. When $\rho = 1$, the standard deviation of the center aperture function is equal to the grid radius scaled by the detection scale. This function is inspired by SIFT which use a similarly defined center aperture function. Their standard deviation of the corresponding function is pre-defined to have a scale half its square-grid width and height, whereas we search for the optimal ρ by adding it to our parameter studies.

4.4 Binning aperture function

The binning aperture function decides the shape of the histograms that make up our descriptor. We define it as

$$B(\mathbf{x}; f_i, f) = K\left(f(\mathbf{x}) - f_i; \beta \frac{f_{\max} - f_{\min}}{2n}\right) \tag{4.6}$$

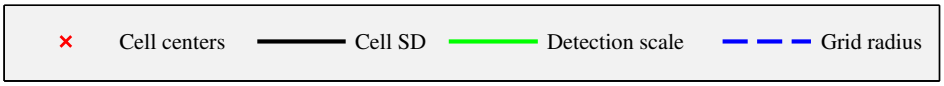
where K is a kernel function as described in 3.3.1, and β is the bin scale, defined relative to a uniform bin layout. That is, when $\beta = 1$, the standard deviations of each bin are tangent to neighbouring bins.

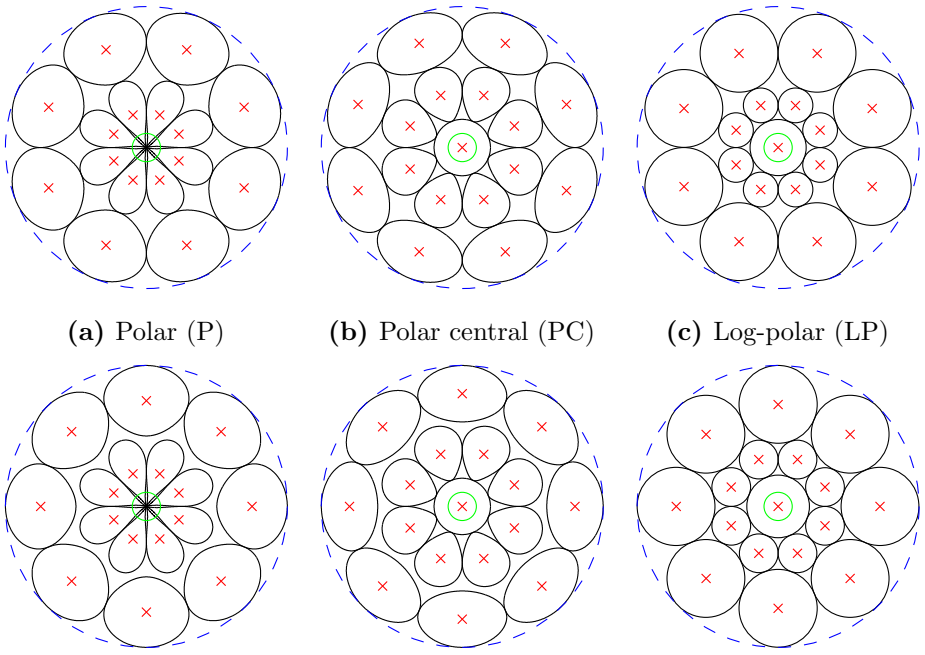
For our proposed descriptors the value function f is either the gradient orientation or the shape index. The number n of bin centers f_i for $i = 1, \ldots, n$ is variable and hence we need to define the bin centers as a function of n. Given an interval $[f_{\min}, f_{\max}]$, we define the bin centers f_i

$$f_i = f_{\min} + \frac{f_{\max} - f_{\min}}{n}\left(i - \frac{1}{2}\right) \tag{4.7}$$

which corresponds to splitting the interval into n equally sized parts and placing the bin centers f_i in each of the n centers of these parts.

The shape index has values spanning the interval $[-1, 1]$, and is thus subject to renormalization as described in Section 3.3.3. The gradient orientation has values spanning the periodic interval $[-\pi, \pi]$, which is handled by wrap-around of the distance $f(\mathbf{x}) - f_i$ as described in Section 3.3.2.

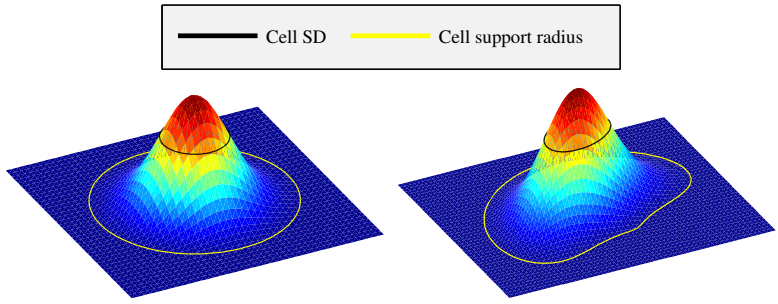




(a) Polar (P) **(b)** Polar central (PC) **(c)** Log-polar (LP)

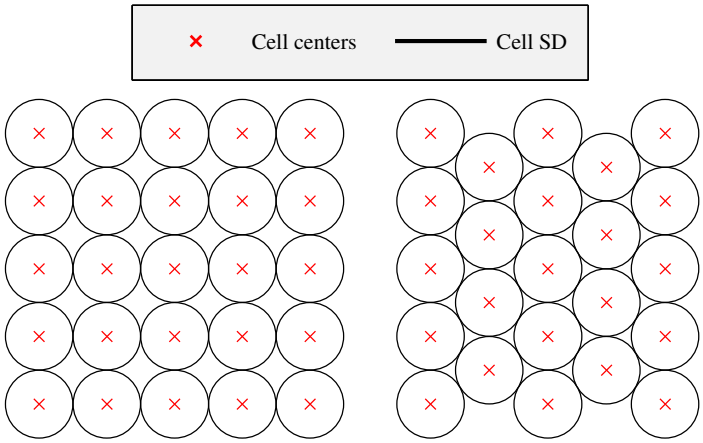
(d) Concentric polar (CP) **(e)** Concentric polar central (CPC) **(f)** Concentric log-polar (CLP)

Figure 4.2: Examples of cell grid types centered around a detected interest point, showing cell centers, 1 standard deviation curves (see Figure 4.3), detection scales, and grid radii. The grid size here is 8×2 (8 cells in each of 2 rings) and the grid radius r is set to 10, which is relative to the detection scale. The polar grids utilize polar Gaussian aperture cell functions while the log-polar use Cartesian Gaussian aperture cell functions.



(a) Gaussian **(b)** Polar Gaussian

Figure 4.3: The two types of Gaussian cell aperture functions plotted in 3D. The black curves are placed 1 standard deviation α from the cell centers. The yellow curves are placed 3α from the cell centers and confine the regions used to compute the cell histograms.



(a) Square grid **(b)** Triangle grid

Figure 4.4: Examples of uniform cell layout grid types used for sliding windows, where we are interested in describing a larger area rather than the local area surrounding a detected point.

4.4.1 Histogram normalization

We have now covered all the functions needed in order to compute the j'th histogram H_j, Equation (4.1), for each cell j as well as the positioning of each of these cells. To get our descriptor $\mathbf{d}$, Equation (4.2), we must concatenate all of the H_j histograms. An optional final step is to view $\mathbf{d}$ as a single vector, and normalize it by using the L_2-norm as Lowe [Low04] does in his SIFT descriptor. This could potentially cause problems, e.g. when dealing with occluded interest points. We investigate the effect on our optimized descriptors for each application in the corresponding parameter studies.

4.5 Descriptor parameters

To summarize, our descriptor has the following parameters that will be part of our parameter studies:

- Normalization scale η
- Cell kernel
- Cell scale α
- Interest point cell layout specific parameters:
  - Grid type
  - Grid size
  - Grid radius r
  - Center scale ρ

- Uniform cell layout specific parameters:
  - Grid type
  - Cell spacing r
- Bin kernel
- Bin scale β
- Bin count n

4.6 Example

In order to visualize the construction of our descriptor, we will show an example of the whole process using an interest point detector. The example would however not differ much if we had used a sliding window approach with a uniform cell layout instead of an interest point layout. We illustrate both the GO and SI descriptor introduced in Section 4.1. The parameters chosen for this example are chosen for illustrative purposes and not necessarily optimal. At first we convert the image to grayscale and extract interest points from an image with a multi-scale DoG detector, as shown in Figure 4.5.

The next step is to construct the scale space images as described in Section 4.1 based on the DoG scales. This results in scale space images which are smoothed and downsampled versions of the original image. Both operations are done according to the respective scales, which causes all features to have the same size in pixels regardless of their respective scales. We compute the new feature coordinates in the scale space images, and remove those points that are too close to the edge: if any cell support radius (shown in Figure 4.3) belonging to a feature spans outside the image border, we discard the feature. Figure 4.6 shows a handful of these images with their respective features, and which of the features that are discarded.

From the blurred and downsampled images we apply derivative filters in order to compute value and magnitude images by their respective functions. In the case of GO, x- and y-derivatives are needed, whereas xx-, xy-, and yy-derivatives are needed for SI. The derivatives are used to compute bin value and magnitude functions over the entire image. We then apply our pixel-wise normalization scheme to the magnitudes with normalization scale $\eta = 2$. These images are shown in Figures 4.8 and 4.9 for GO and SI respectively.

We need to compute the cell and center weights for pixels in every cell according to the chosen cell layout parameters. In this case we use Gaussian cell kernels, where the support radius is confined to 3 times the cell radius. Pixels outside this distance to each cell center are ignored. For pixels inside, we compute the spatial weights according to Sections 4.2 and 4.3. The

products of these weights are illustrated in Figure 4.7, but since many cells overlap we show the maximal weight for each pixel.

The last type of weights needed are the bin values. Figure 4.10 shows bin value images for two bin centers with opposite gradient orientations. The purpose of these images is to illustrate the bin weighting and hence we have left out the corresponding SI examples. The images clearly show their corresponding parts of the beer can structures as well as some background noise. Though not shown in these images, to save computation time we only compute bin values once for each pixel with a non-zero spatial weight, and we then simply look up the value when needed in a cell.

Finally, the four weights are multiplied in order to construct the cell histograms. Figures 4.11 and 4.12 illustrate one of these grids for a single feature for GO and SI respectively. We see that there is a correspondence between the dominating orientations of the GO histograms and the gradient field of the underlying image. The weighted histogram values are then concatenated and normalized for each feature, resulting in the final descriptor. For SI the correspondence between the histograms and the underlying image is harder to see.

Figure 4.5: Interest points (green) found by a multi-scale DoG detector on an example image. The circle radii illustrate the detection scale of each point. 529 points are detected in total.

Figure 4.6: The smoothed and downsampled images at various scales with their corresponding kept interest points (green) and removed interest points (red).

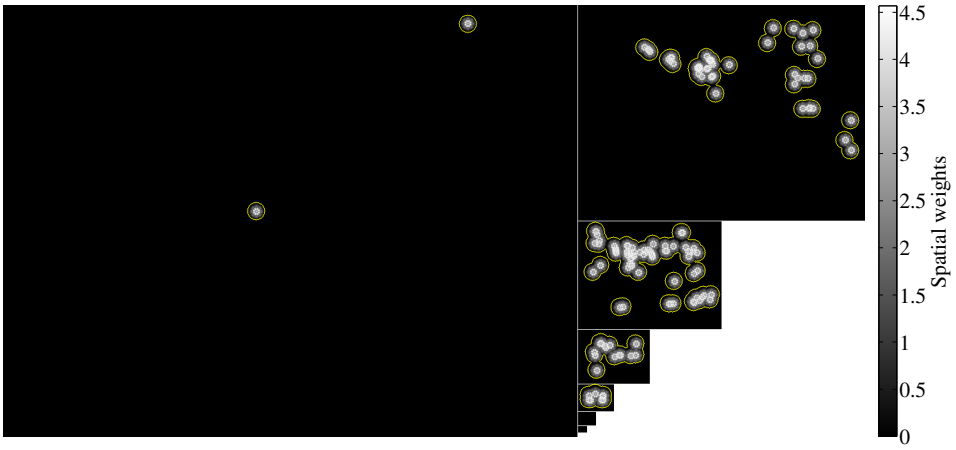
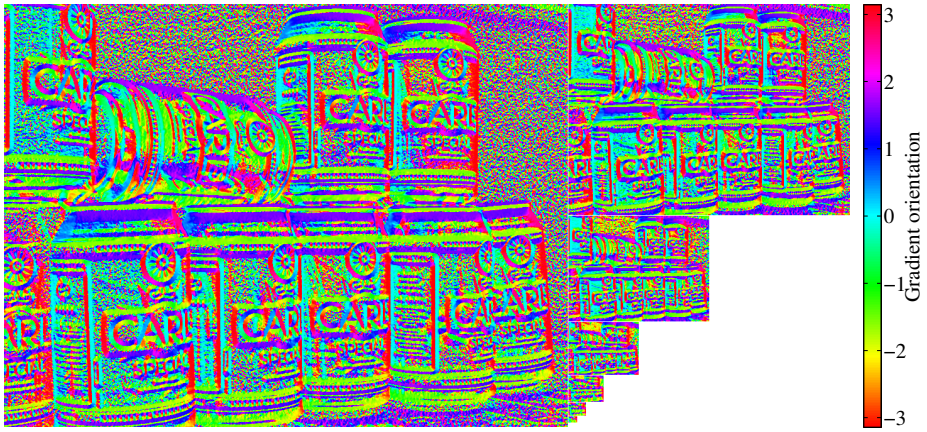
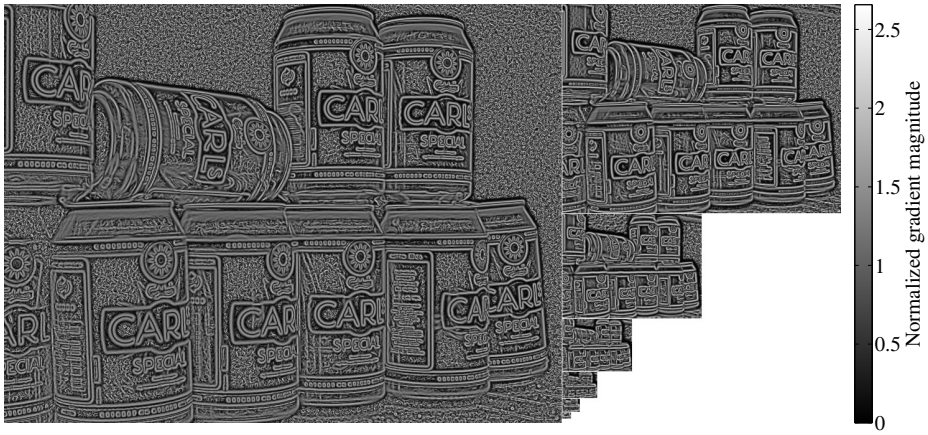


Figure 4.7: Product of cell and center weights. When cells overlap, the maximal weight is shown. The yellow borders indicate the scope of pixels within the support radii of the cells. Only these are used to construct the cell histograms.



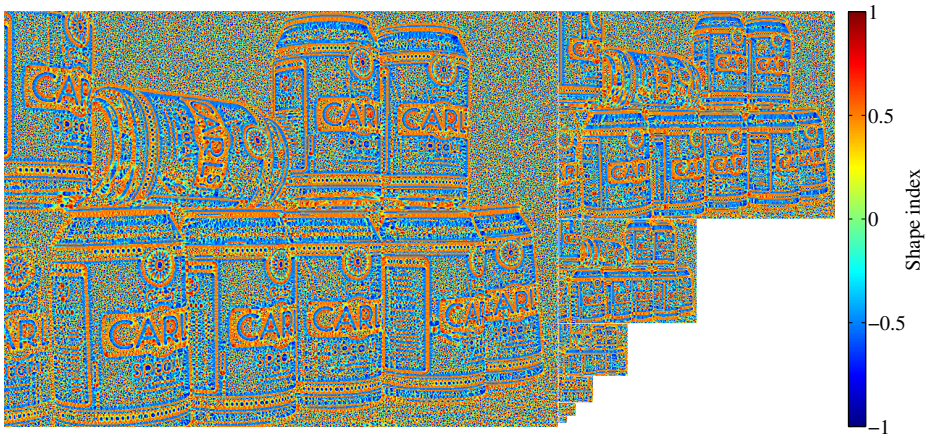
(a) Gradient orientation images

(b) Gradient magnitude images



(c) Normalized pixel magnitude images

Figure 4.8: Images (a) and (b) show gradient orientation and magnitude images computed at various scales. Image (c) shows the pixel-wise normalized magnitude images, where the magnitudes are relative to a small local area.

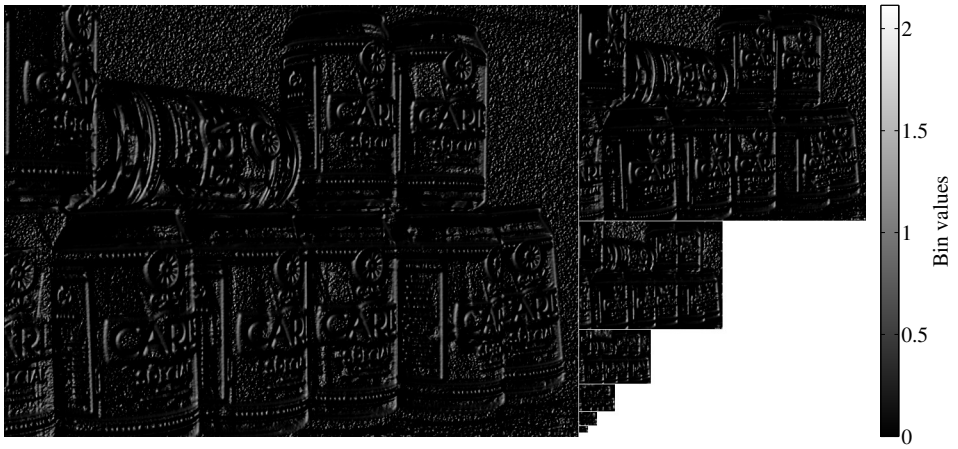


(a) Shape index images

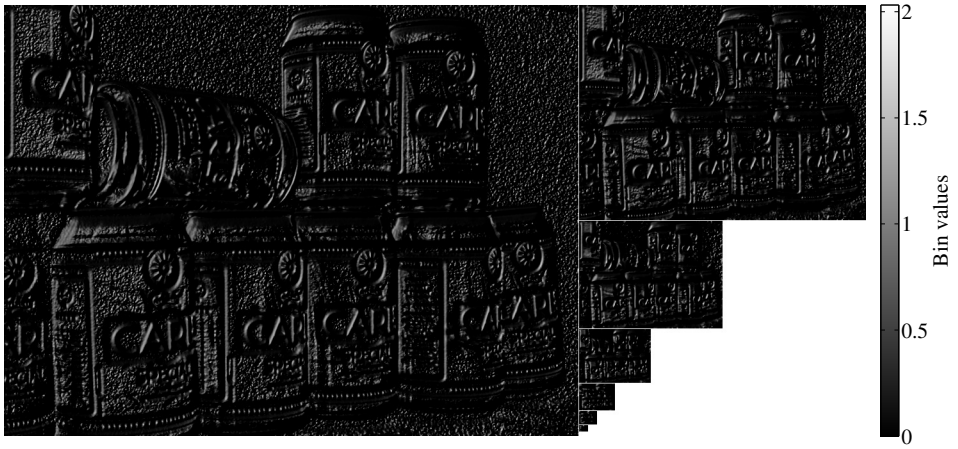
(b) Curvedness images

(c) Normalized curvedness images

Figure 4.9: Images (a) and (b) show shape index and curvedness images computed at various scales. Image (c) shows the pixel-ise normalized curvedness images, where the curvedness is relative to a small local area.



(a) Bin 1 of 8



(b) Bin 5 of 8

Figure 4.10: Bin value images for two of the histogram bins. Images (a) and (b) correspond to the opposite bin centers $\Theta = -157.5°$ and $\Theta = 22.5°$, and we see that opposite sides of the beer cans are given the greatest bin values for each image.

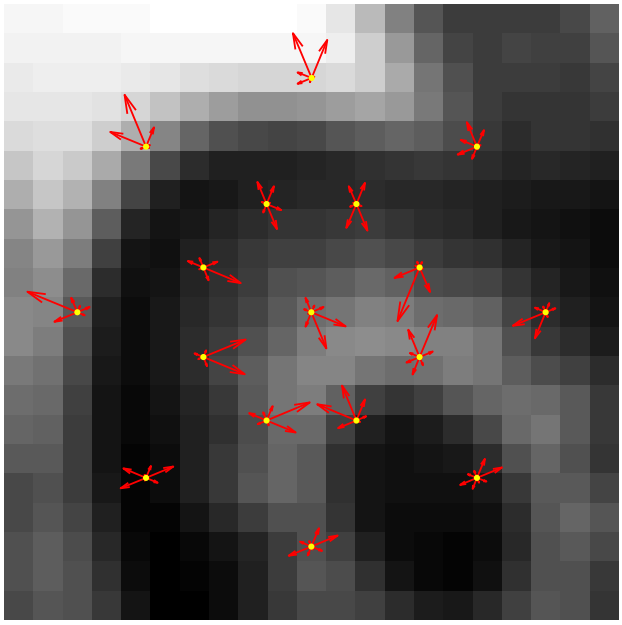
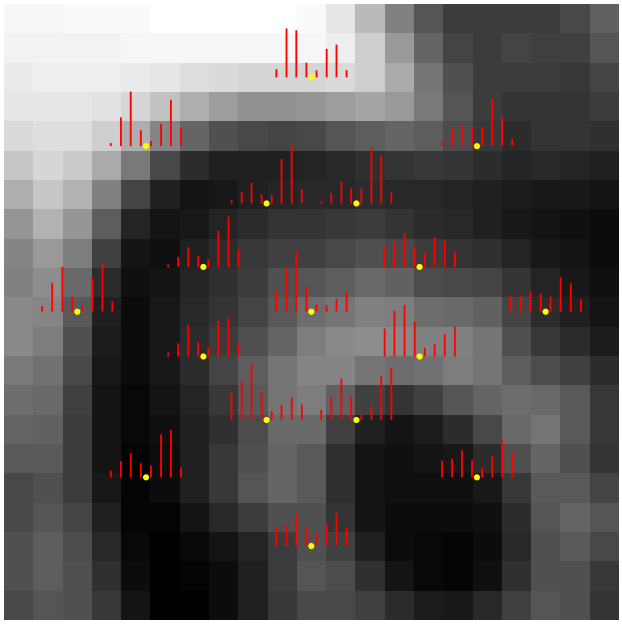
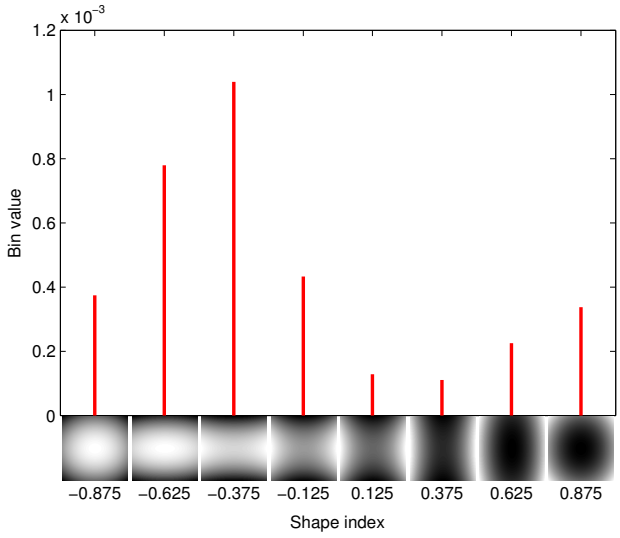


Figure 4.11: The cell histograms for a single feature that make up our GO descriptor. For each cell center (yellow), the weighted histogram values (red) are drawn in the direction of the corresponding bin center orientations. We see that the cell histograms generally point towards larger intensities.



(a)



(b)

Figure 4.12: The cell histograms for a single feature that make up our SI descriptor (a). For each cell center (yellow), the weighted histogram values (red) are drawn. (b) illustrates the central histogram of (a) where the images at each bin show examples of the corresponding structure.

Chapter 5

Image correspondence

In this chapter we explain the image correspondence problem and the application of our descriptor to solve it.

Image correspondence is the problem of matching features across two images A and B of the same object. A match should ideally indicate that the two features correspond to the same physical point or object. The two images of the object can be captured under different lighting conditions, taken from different positions, and have varying tilt, rotation, zoom, and focus; hence the images will differ based on these variables. By using an interest point detector in combination with a descriptor, we are able to find interest points in A and B, describe each interest point with a descriptor algorithm, and compare these descriptors across the two images by a given similarity measure to estimate whether the points match.

The chapter is structured as follows. First we describe the steps of our approach: the strategy to select matches between images, the similarity measure used for this, and the the measure used to evaluate matching performance. We then describe the dataset and how to use it to determine correctness of matches. Next, we specify how we apply our descriptor to the problem as well as our experimental setup. In order to visualize our approach, we then extend the earlier example with image correspondence. Finally, we present our parameter optimization and test results, which include a comparison with SIFT.

5.1 Matching strategies

A matching strategy is a method of picking potential matches between images and assigning them a score denoting how confident we are in the match. If the confidence score is below a chosen threshold t, the match is positive. Otherwise it is a negative match and rejected. mikolajczyk2005performance described the following three matching strategies that are used when solving the image correspondence problem: Simple *thresholding* compares each

descriptor in image A with each descriptor in image B to find potential matches. The confidence score is defined as the mutual distance between each pair of descriptors. *Best-thresholding* uses the same confidence score as the thresholding strategy, but only the best matching descriptors in B for each descriptor in A are considered. *Ratio-thresholding* also considers only the best matching descriptors. The distances B_1 and B_2 between each descriptor in A and the two best matching descriptors in B are computed, and the ratio $\frac{B_1}{B_2}$ between the two distances is used as confidence score.

We choose the ratio-thresholding strategy since this is the one favoured in previous image correspondence evaluations [MS05; DAP11; Lar+12; Low04]. Additionally this matching strategy is more general since it uses the ratio between the distance to the two best matching features instead of the absolute values giving less dependence on the descriptor in question. The two following cases give a good rationale for choosing this matching strategy: The first case is when two descriptors are good matches. In this case the descriptors will be very similar, and their difference could be caused by noise or other small variations. Due to this fact the algorithm shouldn't choose one over the other, and hence the matching is negative since the distance ratio is close to 1. The second case is when only poorly matching descriptors are present. In this case the distance to both of the best matching descriptors will be high and the distance ratio close to 1 giving us a negative match as desired.

When using the approach in practice, a threshold t is chosen depending on the precision needed, and only the positive matches are extracted.

5.2 Similarity measures

We need a similarity measure in order to compute the distance between two descriptors. Recall that descriptors are vectors of real numbers. The Euclidean distance is widely used as similarity measure in the literature [Low04; KS04; MS05]. Larsen [Lar12] evaluated the following different similarity measures for histogram based descriptors similar to ours: L_1 distance, Euclidean distance, χ^2 distance, Kullback-Leibler divergence, and Jensen-Shannon divergence, and he found no notable difference in performance. Furthermore preliminary tests on our descriptor show that Euclidean distance (L_2) slightly outperforms L_1-distance. Therefore we choose to use Euclidean distance.

5.3 Performance measures

Having established the matching strategy and similarity measure, we now describe the two performance measures which we use for evaluation of descriptor algorithms.

Assume that we have computed the descriptor distances, found the two best matching descriptors in B for each descriptor in A and computed their distance ratio r. We now wish to compute the performance of the descriptor algorithm utilized. The problem has been reduced to a binary classification problem of whether two descriptors match or not. Following the approach described in Section 3.8, using the distance ratio r as the classification score s, we are able to compute the PR- and ROC-curves as well as their AUCs to get a measure of the performance of our descriptor.

Recall the discussion of the benefits of ROC and PR in Section 3.8. The PR measure is beneficial to use when the positive class is of greater importance than the negative class. In the image correspondence problem we wish to find as many corresponding points as possible. However negative matches are not used, and therefore we don't want to emphasize the true negatives in our performance measure. This means that we would rather compare the number of false negatives with true positives, as recall does, instead of with true negatives, as false positive rate does. Therefore the PR measure has greater importance than the ROC measure.

5.4 Dataset

The dataset, which we use for training and evaluation of our descriptor, is called the *DTU Robot 3D dataset* [ADP10b] (from now on called the DTU dataset). It consists of images of objects taken in a closed black box with varying illumination using 19 fixed position LED lights. In total there are 60 scenes with varying objects. Figure 5.1 shows 4 example scenes. Aanæs, Dahl, and Perfanov [ADP10a] classify the scenes into categories as shown in Table 5.1.

The camera is positioned around each scene using an industrial robot arm, which has automatically captured each scene from 119 positions. These positions are defined from a fixed frontal view varying the viewpoint θ in three arcs at different distance d to the scene. Arc 1 has $d = 0.5\,\text{m}$ to the scene and θ spans $\pm 40°$, arc 2 has $d = 0.65\,\text{m}$ and θ spans $\pm 25°$, and arc 3 has $d = 0.8\,\text{m}$ and θ spans $\pm 20°$. Furthermore a linear path is captured by moving the camera away from the scene, which corresponds to zooming or scaling the scene. This is done at $\theta = 0°$ and d spans $[0.5\,\text{m}; 0.8\,\text{m}]$. At each of the 119 camera positions 19 individual images I_i are taken with each of the LED lights i, for $i = 1, \ldots, 19$ turned on. Using the given camera positions we get four camera paths: three arc paths and one linear path. Figure 5.2 shows an overview of these camera paths. When computing the performance across the paths we wish to solve the image correspondence problem for matching each image in each path with the *key frame* image (middle position on arc 1), since this gives us the performance when varying the viewing angle and scale in a structured manner.

Figure 5.1: Example scenes from the DTU dataset

Class	Scene numbers	Total
House	1, 4, 8, 31, 32, 49, 50, 55	8
Books	2, 11, 20, 21	4
Fabric	5, 6, 45, 46, 47, 48	6
Greens	23, 24, 25, 26, 27, 51, 52, 53, 54, 56	10
Beer	15, 16	2
Teddy Bears	9, 10, 43, 44	4
Building Materials	33, 34, 35, 36, 37	5
Decorative Items (Art)	38, 39, 40, 41, 42	5
Groceries	12, 28, 29, 30	4
Twigs and Leaves	17, 57, 58, 59, 60	5

Table 5.1: Scene object classifications from [ADP10a, Table 1]

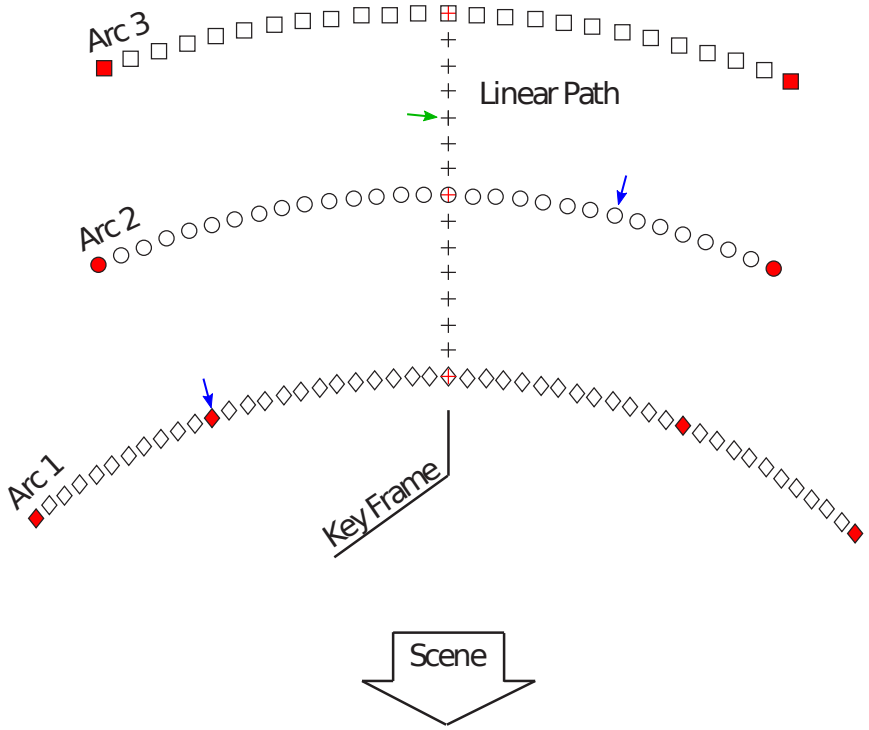


Figure 5.2: Overview of camera positions and paths in the DTU dataset. The red positions mark training camera positions, the arrows mark camera positions of the light path experiments where only the ones marked with blue are used for training. Reproduced and improved illustration from Aanæs, Dahl, and Pedersen [ADP10b, Figure 3, pp. 3].

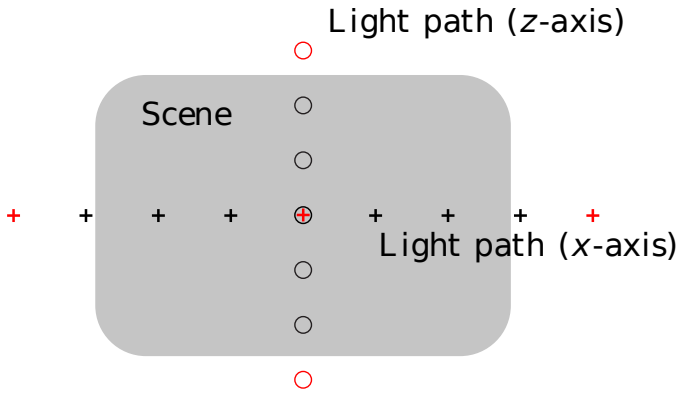


Figure 5.3: Light positions of the x- and z-axis light paths. Training light positions marked with red. Reproduced and improved figure from Larsen [Lar12, Fig. 1, pp. 644].

The choice of capturing the scenes with individual LED lights created images with a high amount of cast shadows. Larsen et al. [Lar+12] created a set[1] of artificial diffuse and light paths from the individual LED images which we now briefly explain. The artificial diffuse light images are created from the individual lightings in order to only evaluate the performance of our descriptor under viewpoint changes and to get more natural images. These diffuse images are created by averaging over the individual light images for each camera position:

$$I_{\text{diffuse}} = \frac{1}{19} \sum_{i=1}^{19} I_i \tag{5.1}$$

Since the dataset consists of individual LED images, one is able to construct images simulating two light source paths going from right to left and back to front respectively. Given a light position $\mathbf{x}$ in the spatial domain of the LED positions, the image $I_{\mathbf{x}}$ is constructed by weighting each LED image by the Gaussian of the distance to $\mathbf{x}$:

$$I_x = \sum_{i=1}^{19} G(\mathbf{x} - \mathbf{x}_i, \sigma) I_i \tag{5.2}$$

Figure 5.3 shows the selected light positions. In order to get a somewhat general measurement for the robustness against light variations, there are light paths generated for the following four image positions: 12 (arc 1), 25 (arc 1), 60 (linear path), and 87 (arc 2). The light path images are compared to the key frame image with diffuse light, which gives a difference in lighting conditions. See Aanæs, Dahl, and Pedersen [ADP10b], Aanæs, Dahl, and Perfanov [ADP10a], and Larsen et al. [Lar+12] for more information about the dataset and the generated light paths.

Figure 5.4 show 6 of the different light images for scene 4 at camera position 60. Figures 5.4a to 5.4c show the images taken with individual LED lighting (numbers 2, 17, and 8 respectively), and Figure 5.4d shows the diffuse light image. We here notice the significant difference in cast shadows using the three LEDs individually compared to the diffuse light image. The diffuse light image is however quite dark compared to the LED 8 light image, which could potentially cause problems in some scenes generating too few interest points. Figures 5.4e and 5.4f show the left- and rightmost positions of the x-axis light path. By comparing these to their LED counterparts (Figures 5.4b and 5.4c respectively), we see that the cast shadows are less significant and therefore look more natural. Figure 5.5 shows 3 different camera positions for scene 4 including the key frame (position 25).

[1] dataset available at `http://roboimagedata.imm.dtu.dk/data/condensed.tar.gz`

(a) LED 8 light **(b)** LED 2 light **(c)** LED 17 light

(d) Diffuse light **(e)** Leftmost position of the x-axis light path **(f)** Rightmost position of the x-axis light path

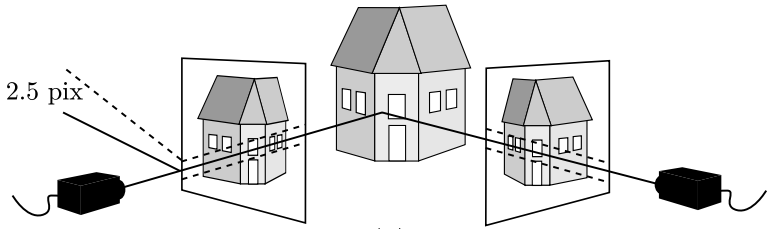
Figure 5.4: Examples of light images in scene 4 at camera position 60.

(a) Camera position 12 (arc 1) **(b)** Camera position 25 (key frame) **(c)** Camera position 64 (linear path)

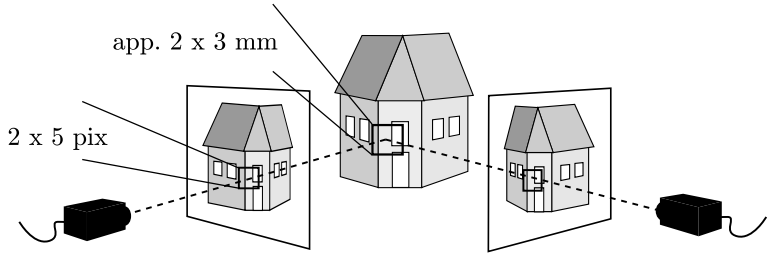
Figure 5.5: Examples of various camera positions of scene 4 using diffuse light.

5.4.1 Evaluation

In order to be able to classify matches between two images A and B as true or false, we need an evaluation method. In other words we need to generate the match ground truth. The DTU dataset has been scanned using *structured light* generating point clouds for the surfaces in each image. Using these points and the known camera positions for each image, we are able to check if a matching of two points from A and B indeed correspond to 3D points close enough to one another to be true. Aanæs, Dahl, and Pedersen [ADP10b] defined three evaluation criteria, of which we will be using the two first as chosen in Larsen [Lar12]: epipolar and surface geometry. We first define the property of being 3D reconstructable: An image point is said to be 3D reconstructable if there exists a point from the surface point cloud within a 10 pixel window in the image plane. This corresponds to a pixel within



(a) Epipolar geometry consistency: Corresponding points should have an orthogonal distance to each others epipolar lines of maximum 2.5 pixel.



(b) Surface geometry consistency: Corresponding descriptors should be 3D reconstructable and be within 3 mm of each other in the surface point cloud.

Figure 5.6: Illustration of the two ground truth evaluation criteria. Figure reproduced from [ADP10b, Figure 5 (a-b),pp. 4].

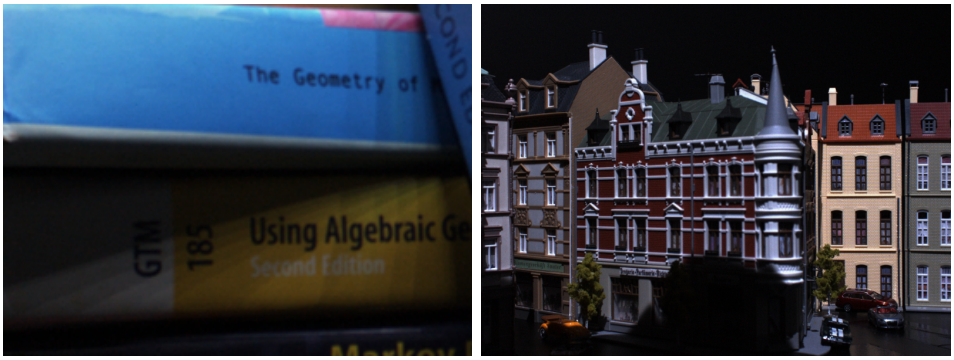
approximately 6 mm in the surface point cloud. Sometimes a matching is made for a point which is not 3D reconstructable. In this case no ground truth can be calculated, and the matching is ignored in the performance evaluation. The following two criteria (illustrated in Figure 5.6) determine the matching correspondence ground truth of the two points:

- **Epipolar geometry consistency:** Given a point in image A, the point in image B should be within a 2.5 pixel orthogonal distance to the epipolar line of the first point as illustrated in Figure 5.6a.
- **Surface geometry consistency:** The points need to be 3D reconstructable, and they need to be within 3 mm of each other in the surface point cloud as illustrated in Figure 5.6b.

The evaluation described in this section has been implemented by Aanæs, Dahl, and Pedersen [ADP10b]. In this thesis we have used and improved their implementation for our evaluation of descriptor performance.

5.4.2 Pitfalls and deficiencies

The DTU dataset is built to test the ability of computer vision systems to cope with viewpoint changes, scaling, light changes, and to a certain extent



(a) Diffuse lighting in set 2, image 20, with shadow artifacts

(b) Robot arm shadow in set 4, image 7, LED light 3

Figure 5.7: Examples of problematic images in the DTU dataset

occlusions as well. Since all images are captured with the same camera tilt, no rotation of objects occurs, and hence rotational invariance and robustness cannot be evaluated using this dataset.

Figure 5.7 shows the problems we have found within the DTU dataset, which we will go through here.

The artificial diffuse light is, as mentioned in the previous section, created by combining images captured under individual LED lighting. This is not optimal as some of the images are visually suffering from the spatial layout of the LED lights. Figure 5.7a shows image 20 (arc 1) from set 2 of the dataset. The diffuse light problem is clearly visible in the cast shadows seen on the 1st and 3rd book in the stack of lying books. Ideally these shadows would form either a smoothed or a single hard shadow edge instead of a number of gradually fading hard edges.

The automatic capturing of images using a robot arm combined with the individual LED lighting has the side-effect of the robot arm casting shadows in some of the images. This is seen when using the 3rd LED light and having captured the scenes from the far left of arc 1. Figure 5.7b shows an example of a cast shadow originating from the robot arm. This is however only a problem with one of the 19 LED light images, and hence the effect on the final test images is minimal.

5.5 Application of descriptors

Recall that we introduced the GO, SI, and GO+SI descriptors in Section 4.1. We will use the DTU dataset to evaluate these descriptors on the image correspondence problem. Our descriptors naturally use the interest point cell layouts for this application, described in Section 4.2.1. Additionally we will evaluate SIFT in order to compare our results. Since DTU images have a

fixed orientation, we disable SIFT's orientation estimation step, which our descriptors don't have either. We have tested and verified that using orientation estimation decreases performance. As mentioned in Section 3.7.2, the descriptors will be computed for opponent colour channels and concatenated, which slightly improves the results. Each descriptor will be computed for the same interest points from a multi-scale DoG detector, described in Section 3.5.1. This ensures that only the choice of descriptor can affect the result. Training is done on grayscale images, whereas testing is done on the images converted to opponent colour space as mentioned in Section 3.7.2.

5.6 Experimental setup

The DTU dataset is constructed using four different camera movement paths: Arc 1 through 3 and the linear path described in Section 5.4. We therefore choose to perform one experiment for each of these paths, where we average across the different scenes for each camera position. These four experiments are all conducted using the diffuse light images. Furthermore we conduct two light experiments, where we move the light in the x- and z-axis respectively. For these experiments we both average across scenes and a fixed set of camera positions. In total we have six different experiments with perspective, scale and light transformations.

Since the DTU dataset is not split in a test and a train part, there is no pre-defined way of training and testing an algorithm on the dataset. Larsen et al. [Lar+12] used the Oxford Affine Covariance Region dataset[2] for manual parameter optimization of their Jet-based local image descriptors. This dataset is however very small and limited, and hence we risk severe overfitting to the dataset if used for training. Instead we choose to split the DTU dataset with respect to the scenes into six parts for the purpose of 6-fold cross-validation. For each fold we thus have 50 training scenes and 10 testing scenes.

Testing on a scene is done on the complete camera and light paths, whereas we only perform training on a small subset of these, since the dataset contains quite a large amount of images to be processed. Figure 5.2 shows the camera positions of the dataset as described in Section 5.4. In this illustration the training camera positions for the three arcs and the linear path are marked with red. The camera positions for the light experiments are marked with arrows, where only the ones marked with blue arrows are used for training. Figure 5.3 shows the light positions for the light experiments. The red pluses and circles mark the x- and z-axis light path images used for training.

[2] `http://www.robots.ox.ac.uk/~vgg/research/affine`

5.7 Example

In this section we will continue the example from Section 4.6 by extending it to show the results of solving the image correspondence problem between the image of the example and the corresponding key frame image from the DTU dataset. For this extension of the example we only use the GO descriptor.

Figure 5.8 shows matches between the descriptors from the two images with score s below threshold $t = 0.8$. The green and red lines indicate true and false positive classifications respectively. The images contain several similar beer cans, and hence many of the false positive classifications are caused by patterns that are visible in multiple of the cans.

Recall from Section 5.1 that the chosen matching strategy finds a single correspondence candidate for each feature, and the match is evaluated based on the distance ratio r between the best and second best correspondence candidates. Figures 5.9 and 5.10 show two examples of a true positive match and a true negative match respectively. The green line shows a condition positive match, and the red lines show condition negative matches. In Figure 5.9 we see that there indeed is a correspondence between the best match of the two images with distance 0.083, and that our matching strategy results in a distance ratio (score) s of 0.92. In Figure 5.10 we see that there is no match between the best matching descriptors of each image with distance 0.081. For this match s is computed to 0.97. If we wanted to classify both matches correctly, we should therefore set our threshold t between 0.92 and 0.97. If we had used the distance as similarity measure, we would not have been able to classify both correctly, since the distance of the condition negative match is lower than the distance of the condition positive match.

When having computed the distance ratios of all matches, we are able to construct the ROC- and PR-curves giving us two measures of the performance of the descriptor matching. Figure 5.11 shows the ROC-curve (a) and PR-curve (b) as well as the area under the two curves. The match has a ROC AUC of 0.72 and PR AUC of 0.63. From the ROC plot we see that the descriptor matching performs better than a random classification since the ROC-curve (blue line) lies above the curve of no predictive value (dashed line).

5.8 Parameter study

We wish to optimize the descriptor parameters listed in Section 4.5 with the objective of maximizing average matching PR AUC. We do this separately for GO and SI as they may have different optimal parameters, and simply combine these to create the GO+SI descriptor. As mentioned in the experimental setup, we have split the DTU dataset into six parts to reduce overfitting by 6-fold cross-validation. For each fold we optimize parame-

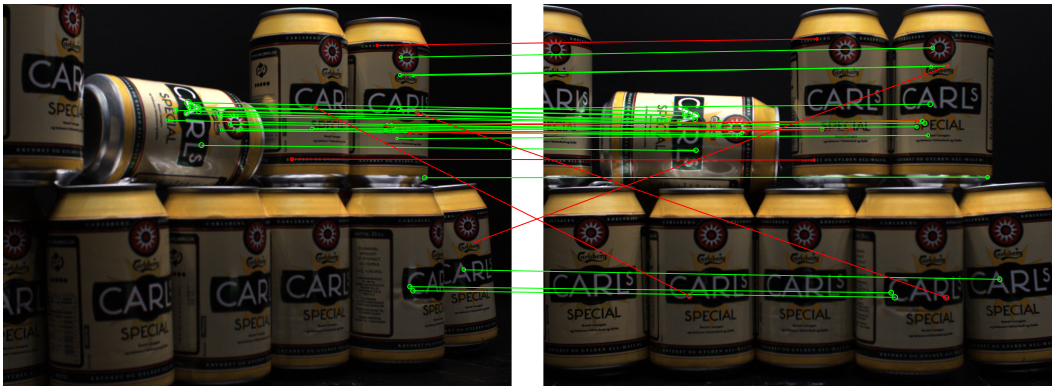


Figure 5.8: Matches with score s below threshold $t = 0.8$. Condition positive and negative matches are shown in green and red, respectively.

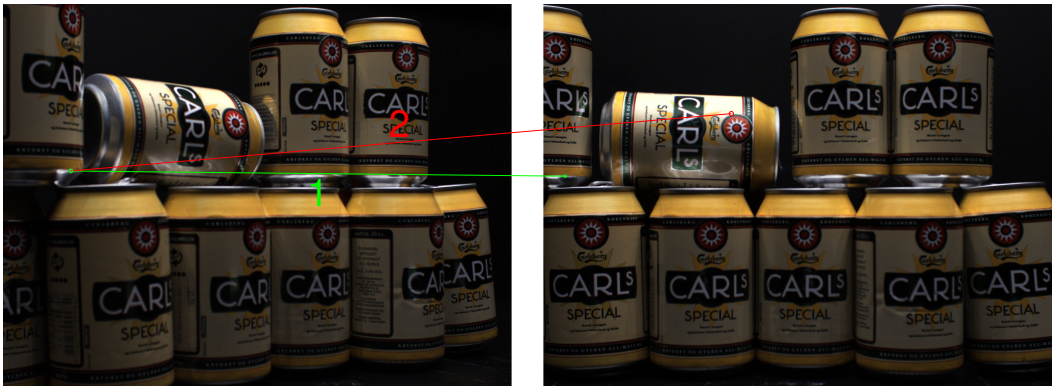
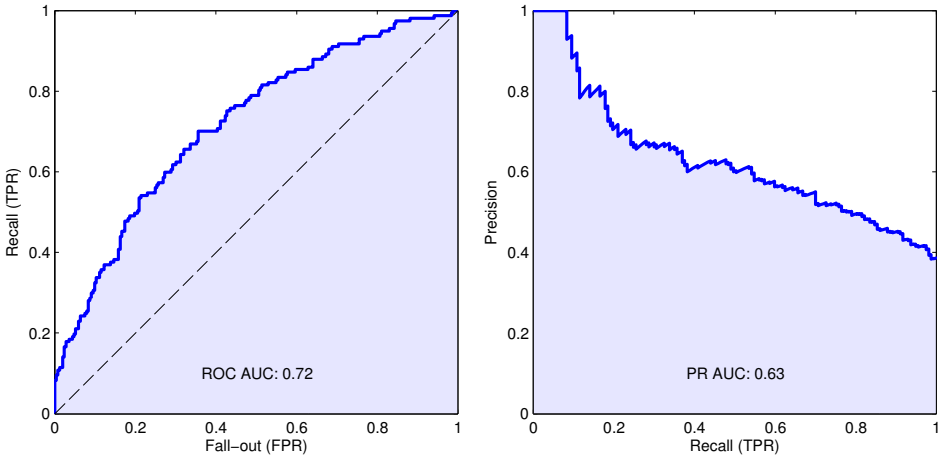


Figure 5.9: Example of a condition positive match. First (1) and second (2) best matches have descriptor distances 0.076 and 0.083, respectively, resulting in a distance ratio of 0.92.

Figure 5.10: Example of an condition negative match. First (1) and second (2) best matches have descriptor distances 0.079 and 0.081, respectively, resulting in a distance ratio of 0.97.



(a) ROC-curve (blue line), curve of no predictive value attained by classifying randomly (dashed line), and area under the ROC-curve (light blue area)

(b) PR-curve (blue line), area under the PR-curve (light blue area)

Figure 5.11: Examples of evaluation measure curves

ters on five parts and test the optimized descriptor on the sixth part. This means we obtain six sets of parameters and a single PR AUC for each image, where the images are evaluated without optimizing across their respective scenes. To save time evaluating parameters, we scale the images to half size and convert them to grayscale. The testing is done on the full scale images converted to opponent colour space as mentioned in Section 3.7.2.

Our actual parameter optimization is problematic because we need to optimize ten parameters at once, and each evaluation is computationally heavy. Our basic strategy is thus to cycle through each parameter and optimize it separately while fixing the other parameters. We do this iteratively until convergence. The risk of this strategy is getting stuck in local optima. To avoid this, recall that we have defined the parameters to reduce correlation: e.g. when bin count n is increased, the width of the bins is automatically reduced to compensate. We also select certain correlated parameters to optimize together: grid type with grid size, as well as cell and bin kernels with their respective scales. Furthermore we optimize continuous parameters in two steps: first a coarse search and then a finer search around the best result.

The optimal parameters of our parameter study for our GO and SI descriptors are shown in Tables 5.2 and 5.3 respectively. From these results we see that most of the parameters are quite stable across the folds. While we use these parameter sets for respective parts of the dataset to avoid overfitting, we also choose a set of recommended parameters based on the parameter study, shown in Table 5.4. We include the parameters for SIFT

translated into our framework for easy comparison.

We first compare the grid type and size of the GO and SI descriptors. SI and a single of the GO folds use the concentric polar central (CPC) grid, whereas the five other GO folds use the polar central (PC) grid. Both descriptors use grid size 12×2. Note that the amount of angular cells is higher than for GLOH and DAISY which use 8 angular cells. Tables 5.5 and 5.6 show the average PR AUC on training data for GO and SI respectively for various choices of grid parameters. We see that the 12×2 PC and CPC grids have nearly equal average PR AUC for both GO and SI, and hence the choice between these two grid types only gives a marginal performance difference. By looking at the grid results, we see that the performance difference between a concentric grid and the corresponding non-concentric grid is very small. The difference between log-polar (LP) and concentric log-polar (CLP) is however significant, which is caused by a more dense grid when using CLP than LP. The polar grids with a central cell perform slightly better than their corresponding polar grids without a central cell. Furthermore log-polar grids are always worse than the corresponding polar grids.

Figures 5.12 and 5.14 show average PR AUC on training data for GO and SI respectively when adjusting the parameters relative to the optimized parameters. Furthermore Figures 5.13 and 5.15 show the dimensionality of GO and SI respectively when varying the bin count n. On each of these graphs, the optimal choice of each parameter is marked with a cross. From the figures we see that the grid radius r as well as cell and bin kernels and scales are the parameters that affect performance the most for both GO and SI.

Looking at the cell and bin kernels, we see that both GO and SI use Gaussian kernels. From Figures 5.12c and 5.12d we see that the differences in maximum performance between Gaussian and triangle kernels are very small. By choosing triangle kernels instead of Gaussian ones, we could save computations, since the two kernels have different support radius in practice. This alternative choice is viable since the optimal α and β for the triangle kernels are only slightly higher than for the Gaussian kernels. This holds for SI as well as seen in Figures 5.14c and 5.14d.

For both descriptors the center scale ρ has little importance once it reaches a certain minimal value for each descriptor as can be seen in Figures 5.12e and 5.14e for GO and SI respectively. This is the reason why the optimal ρ is unstable across the folds for GO. We have also experimented with various other parts of our descriptor that are not listed as parameters. We report some of these results for our optimal GO descriptor on training data, which has a mean PR AUC of 0.781: It is marginally better to normalize the final descriptors before searching for matches. Omitting this step reduces the AUC by 0.002. Recall that we resize images to half size and convert them to grayscale for training. Omitting the down-scaling step improves the AUC by 0.009, and computing the descriptor on opponent colour

	Cross-validation fold					
Parameter	1	2	3	4	5	6
Grid type	PC	CPC	PC	PC	PC	PC
Grid size	12×2	12×2	12×2	12×2	12×2	12×2
Grid radius r	13.5	13.0	13.5	13.5	13.5	13.5
Center scale ρ	1.4	1.9	1.5	1.4	1.7	1.5
Cell kernel	G	G	G	G	G	G
Cell scale α	0.8	0.9	0.8	0.8	0.8	0.8
Bin count n	12	12	12	14	12	14
Bin kernel	G	G	G	G	G	G
Bin scale β	1.3	1.3	1.3	1.3	1.3	1.3
Norm. scale η	1.6	1.6	1.6	1.6	1.6	1.8

Table 5.2: Optimized parameters from our parameter study for GO. Separate sets of parameters are found for each fold.

	Cross-validation fold					
Parameter	1	2	3	4	5	6
Grid type	CPC	CPC	CPC	CPC	CPC	CPC
Grid size	12×2	12×2	12×2	12×2	12×2	12×2
Grid radius r	13.5	14.0	13.5	14.0	14.0	14.0
Center scale ρ	1.9	2.0	2.0	2.0	2.0	2.0
Cell kernel	G	G	G	G	G	G
Cell scale α	1.1	1.0	1.1	1.0	1.0	1.0
Bin count n	8	8	8	8	8	8
Bin kernel	G	G	G	G	G	G
Bin scale β	2.0	2.0	2.0	2.0	2.0	2.0
Norm. scale η	2.6	2.8	2.6	2.6	2.6	2.6

Table 5.3: Optimized parameters from our parameter study for SI. Separate sets of parameters are found for each fold.

Parameter	GO	SI	SIFT
Grid type	PC	CPC	Square
Grid size	12×2	12×2	4×4
Grid radius r	13.5	14.0	8.0
Center scale ρ	1.6	2.0	1.0
Cell kernel	G	G	*Tri*
Cell scale α	0.8	1.0	1.0
Bin count n	12	8	8
Bin kernel	G	G	*Tri*
Bin scale β	1.3	2.0	1.0
Norm. scale η	1.6	2.6	-

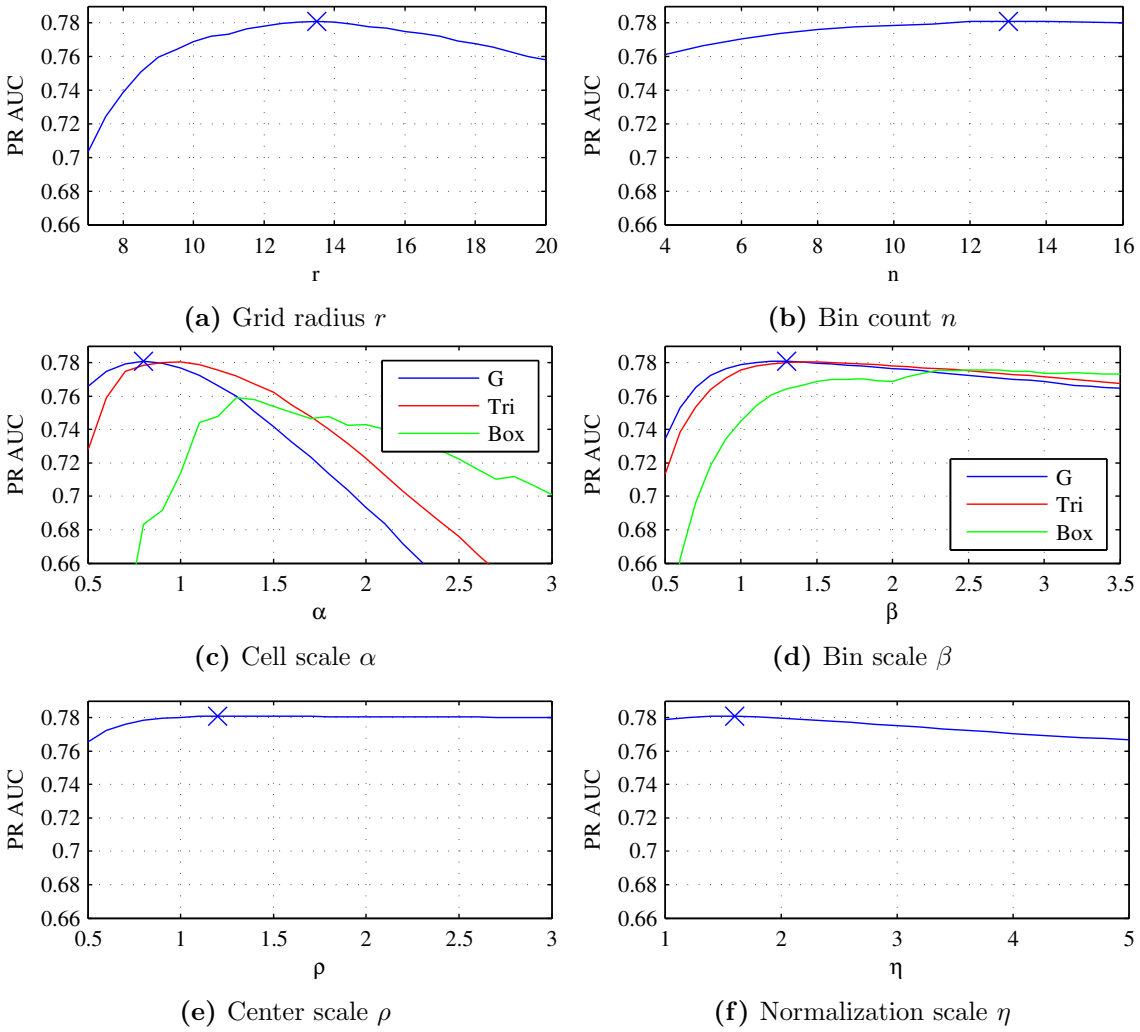
Table 5.4: Based on our parameter study results above, we manually choose a set of recommended parameters for GO and SI. We also include the SIFT parameters translated into our framework when possible. SIFT uses a simple 4×4 square grid, and does not use a local normalization scheme.

Grid size	P	CP	PC	CPC	LP	CLP
6×2	0.756	0.756	0.759	0.759	0.700	0.732
8×2	0.767	0.767	0.770	0.770	0.759	0.763
10×2	0.772	0.772	0.776	0.776	0.769	0.769
12×2	0.775	0.775	**0.779**	0.779	0.770	0.767
6×3	0.763	0.764	0.770	0.770	-	-
8×3	0.771	0.771	0.777	0.777	-	-
6×4	0.758	0.759	0.767	0.767	-	-

Table 5.5: Average matching performance (PR AUC) on training data for GO when adjusting the grid parameters. Optimal performance is marked in bold.

Grid size	P	CP	PC	CPC	LP	CLP
6×2	0.684	0.685	0.691	0.692	0.633	0.664
8×2	0.712	0.713	0.720	0.721	0.707	0.712
10×2	0.729	0.731	0.739	0.739	0.726	0.727
12×2	0.738	0.739	0.748	**0.749**	0.732	0.729
6×3	0.711	0.713	0.717	0.718	-	-
8×3	0.731	0.733	0.736	0.738	-	-
6×4	0.706	0.707	0.721	0.720	-	-

Table 5.6: Average matching performance (PR AUC) on training data for SI when adjusting the grid parameters. Optimal performance is marked in bold.



(a) Grid radius r **(b)** Bin count n

(c) Cell scale α **(d)** Bin scale β

(e) Center scale ρ **(f)** Normalization scale η

Figure 5.12: Average matching performance (PR AUC) on training data for GO when adjusting the parameters. Optimal parameter values are marked with a cross.

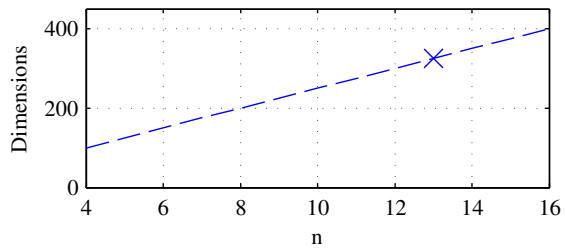
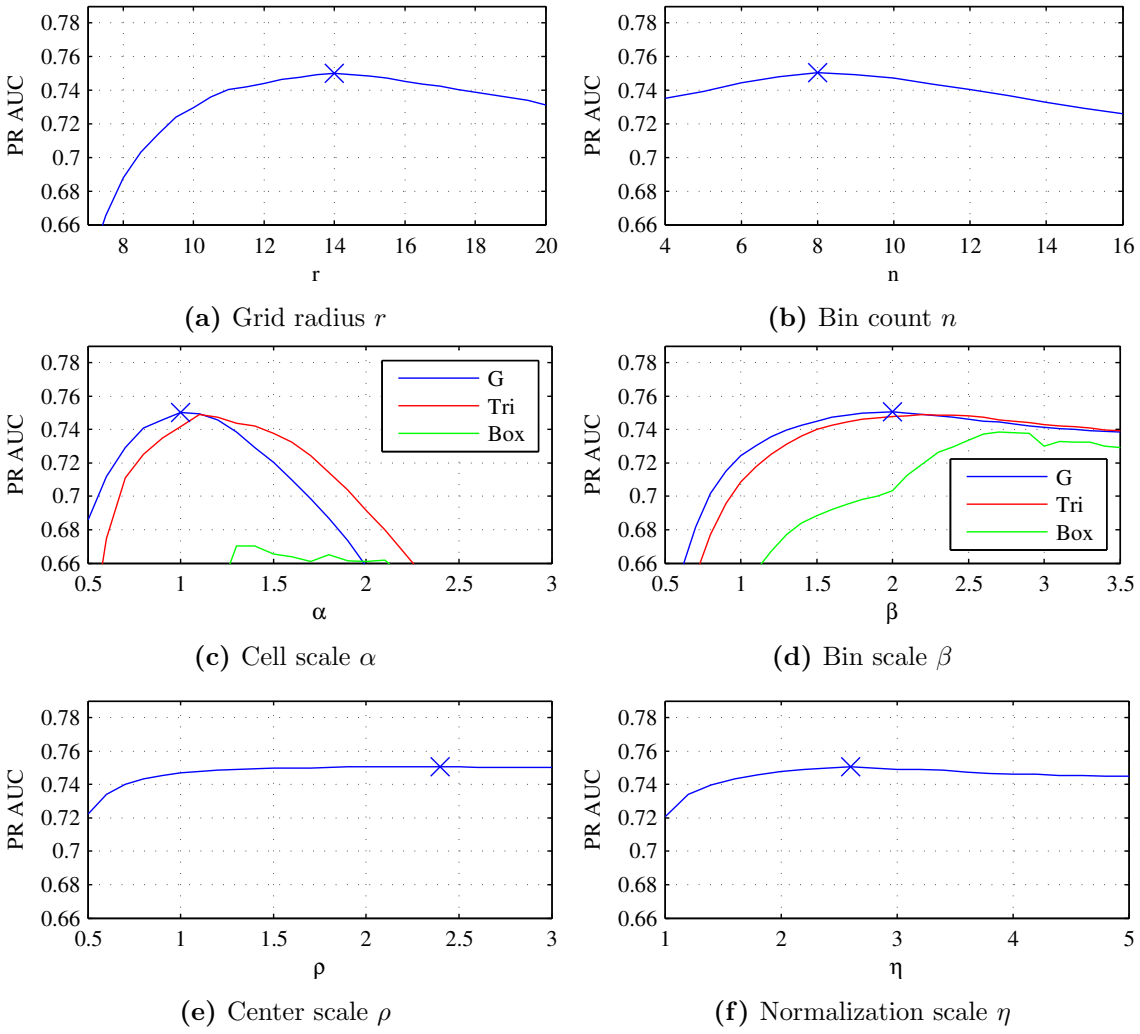


Figure 5.13: Dimensionality of GO when adjusting bin count n.



(a) Grid radius r **(b)** Bin count n

(c) Cell scale α **(d)** Bin scale β

(e) Center scale ρ **(f)** Normalization scale η

Figure 5.14: Average matching performance (PR AUC) on training data for SI when adjusting the parameters. Optimal parameter values are marked with a cross.

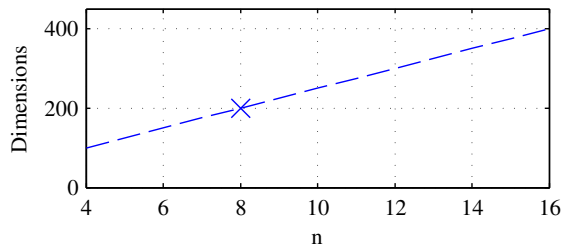


Figure 5.15: Dimensionality of SI when adjusting bin count n.

channels improves the AUC by 0.014. Omitting pixel-wise normalization reduces the AUC by 0.023. Omitting the center aperture function reduces the AUC by 0.002. These tendencies hold for SI as well.

5.9 Results

Having found the optimal parameters for each of the 6 folds in the previous section, we are now able to compute test results for each fold and report the performance of our descriptors. We have chosen to compare our descriptor performance against the SIFT descriptor since Dahl, Aanæs, and Pedersen [DAP11] found the combination of DoG and SIFT to perform well.

Table 5.7 shows average performance across all test images for our GO, SI, and GO+SI as well as SIFT. We see that GO+SI only has a very small advantage over GO despite adding 600 dimensions to the existing 900/1050 (varying across cross-validation folds), and that all our descriptors are marginally better than SIFT on average but have higher dimensionality as well.

The DTU dataset PR and ROC AUC test results for each path are shown in Figures 5.16 and 5.17 respectively. For both these measures the GO (red), SI (green), GO+SI (blue) and SIFT (dashed grey) results are shown. Recall from Section 3.8 that an algorithm which optimizes PR AUC does not necessarily optimize ROC AUC. Despite this fact, the mean ROC AUCs for our descriptors compared to the SIFT ROC AUC are somewhat equal to the corresponding comparisons of mean PR AUCs. SI however has a slight advantage over GO and GO+SI when visually inspecting the mean ROC AUC figures. In Section 5.3 we wrote that we gave the PR measure more importance than ROC, and hence the following comparison of performance between the descriptors is based on the PR AUC figures.

The first thing we notice when looking at all the graphs in Figure 5.16 is, that our GO+SI descriptor is marginally better than GO on arc 1, but they have identical performance on the rest of the variations. Furthermore we notice that our SI descriptor is worse than GO and GO+SI.

When looking at the three arc graphs (Figures 5.16a to 5.16c), we see that GO and GO+SI performance drops less than SIFT, when the camera is moved towards the outermost camera positions. The advantage over SIFT is however lower for arc 2 and 3 than arc 1. This difference could be caused by smaller perspective distortions at arc 2 and 3, since the distance to the camera is increased and the angular differences are smaller than for arc 1. This hypothesis is likewise supported by the fact that all four descriptors have roughly equal performance over scale variations as shown in Figure 5.16d.

For light variations (Figures 5.16e and 5.16f) our descriptors perform better than SIFT.

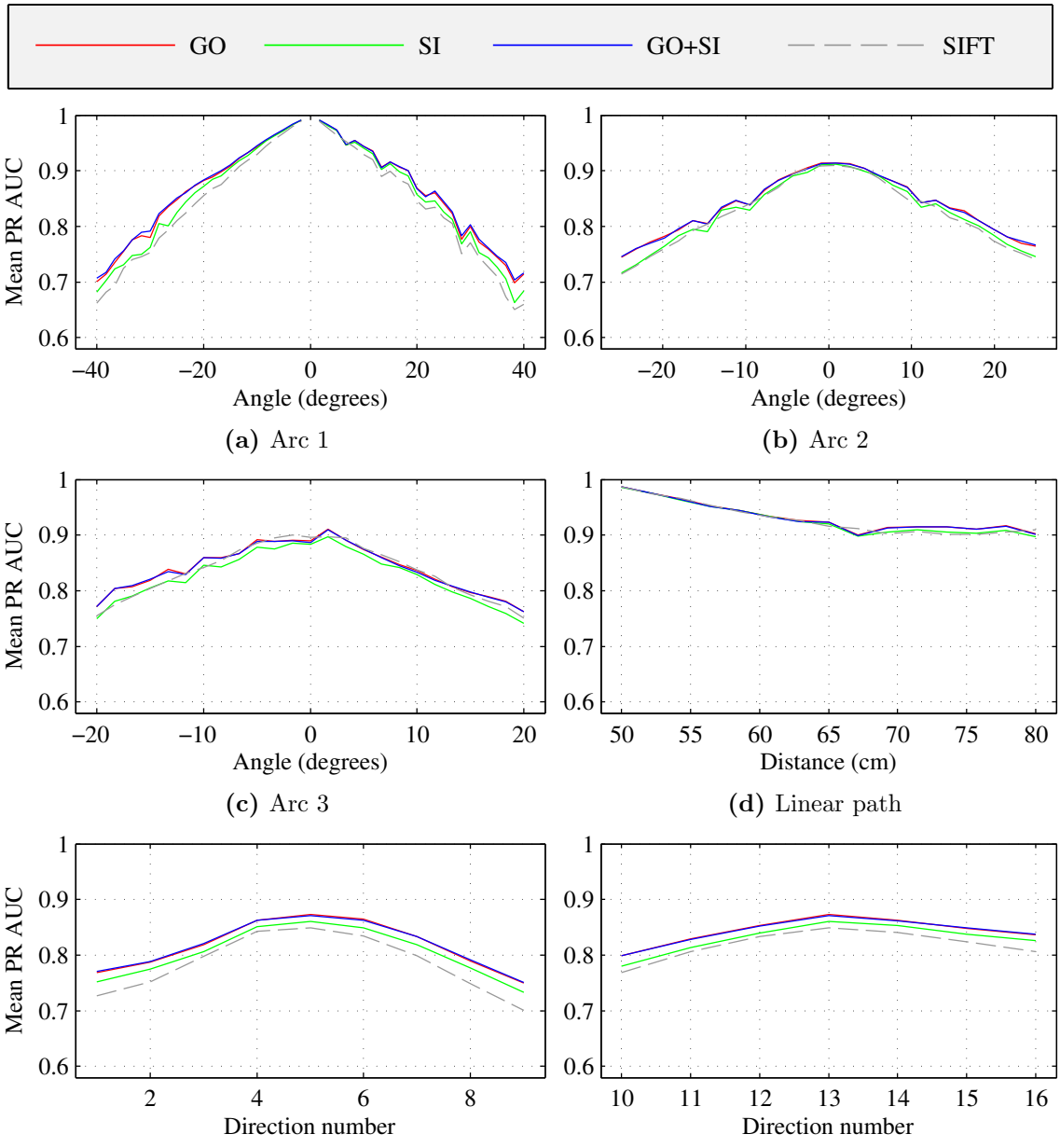
Having inspected the graphs for mean PR AUCs, we are now interested

Descriptor	Dimensions	PR AUC	ROC AUC
GO	900/1050	0.850	0.888
SI	600	0.839	0.890
GO+SI	1500/1650	0.851	0.889
SIFT	384	0.831	0.880

Table 5.7: Average performance across all test images for the various descriptors. Note that GO has a variable number of dimensions because different bin counts were found optimal across the cross-validation folds.

in knowing whether the descriptors are statistically significantly different. We test this separately for each camera and light position. Given a position, assume that the underlying distribution of PR AUCs over the scenes are normally distributed. We discuss whether this assumption is reasonable in Section 7.1. The difference between the mean PR AUC of two descriptors can be estimated by a 95% confidence interval as described in Section 3.9. The resulting 95% confidence interval graphs on the difference between our GO+SI descriptor and SIFT are show in Figure 5.18. As we see from the graphs, the confidence intervals contain 0 for all positions apart from position 1, 7, 8, and 9 of the x-axis light path and position 16 of the z-axis light path. In other words we cannot conclude that GO+SI and SIFT are significantly different except on these outermost positions of the light paths.

Performing these estimates between the remaining combinations of the GO, SI, GO+SI, and SIFT descriptors reveal that none of our descriptors are significantly different apart from GO, which is significantly better than SIFT on positions 6-9 of the x-axis light path and position 16 of the z-axis light path. Since the graphs for all these combinations are both space-consuming and not too informative once we have the above conclusions, we have omitted the graphs from the report itself, but can be found in Appendix E.

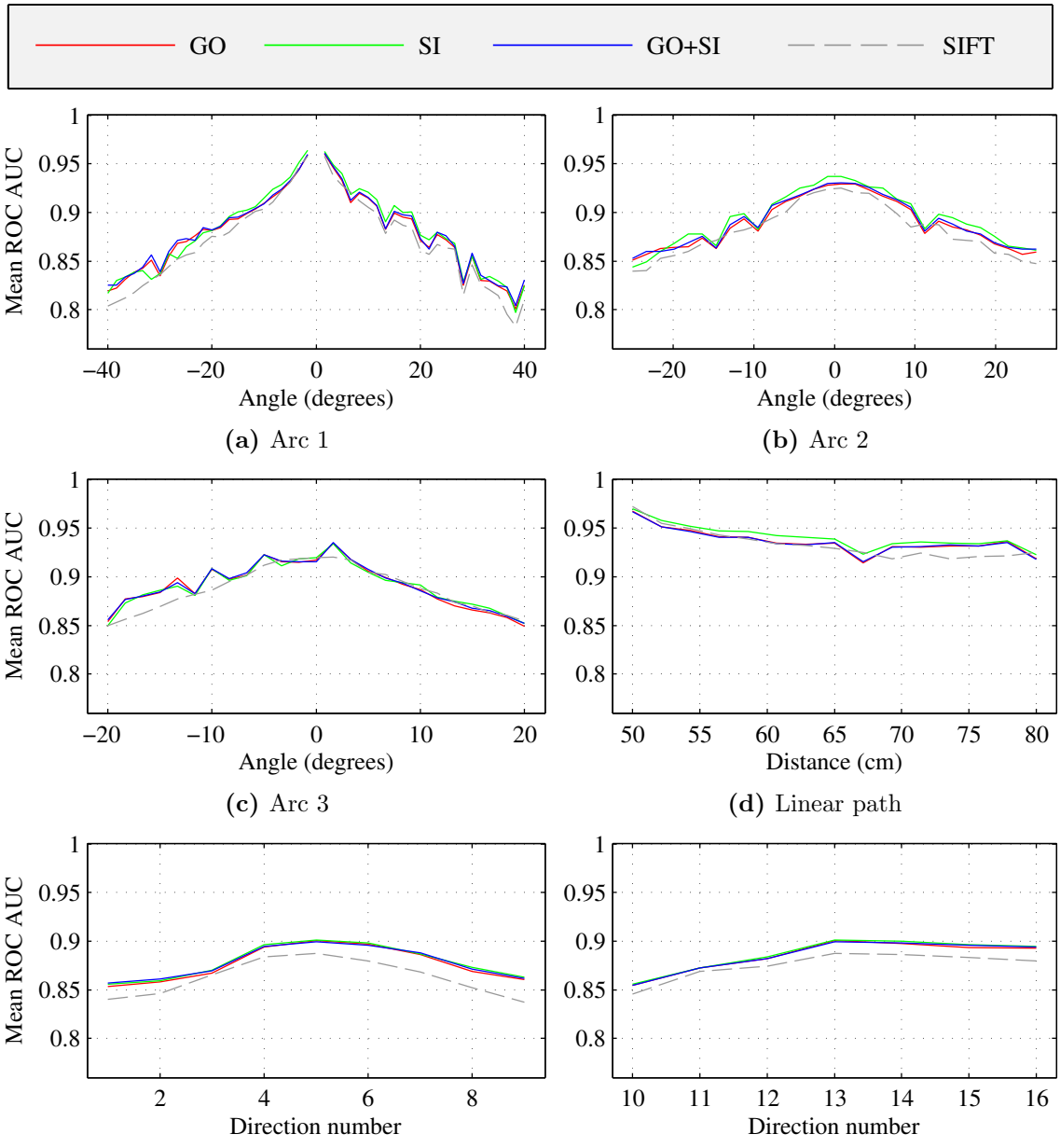


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure 5.16: Mean PR AUC test results on the DTU dataset.

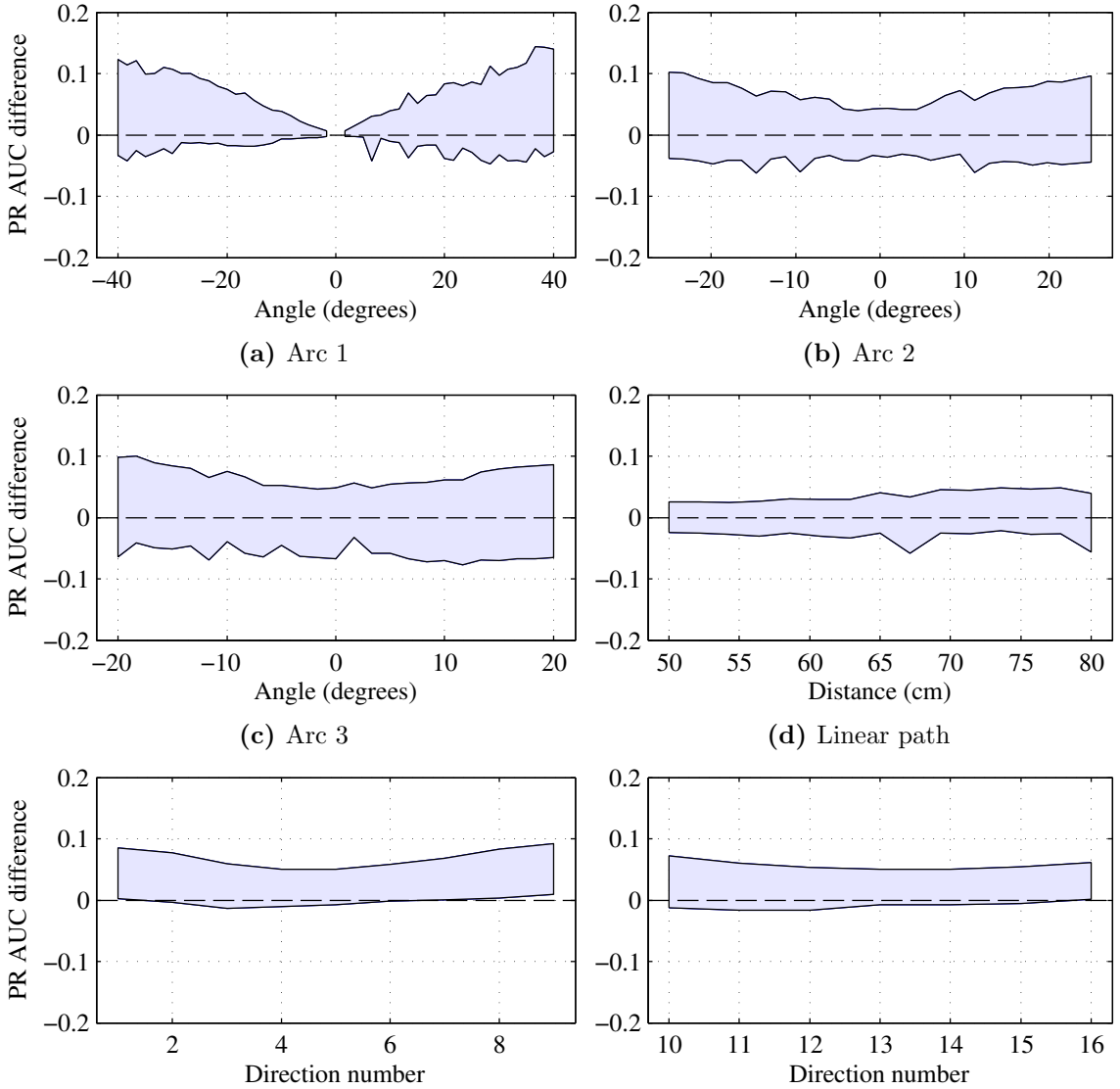


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure 5.17: Mean ROC AUC test results on the DTU dataset paths.



(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure 5.18: 95% confidence intervals on the PR AUC difference between our GO+SI descriptor and SIFT. A positive difference denotes that our GO+SI descriptor outperforms SIFT and vice versa. If the confidence interval for a position contains zero, there is no significant difference in performance.

Chapter 6

Pedestrian detection

In this chapter we describe the pedestrian detection problem and the application of our descriptor to solve it.

Object detection is the problem of locating specified objects in images. One application of this is pedestrian detection, which we have decided to focus on, as it is simple and has extensive datasets. A common approach to the problem is to split it into two steps. First, regions are extracted from the image, and binary classification of the presence of a pedestrian is performed for each region. Afterwards the best regions are picked in the case of overlapping regions around the same pedestrians. For the purpose of comparing descriptors we only focus on the first part. The classification of pedestrians is done by computing a descriptor for each region and using a linear support vector machine for prediction.

The chapter is structured as follows. First we describe support vector machines, and how to evaluate their classification performance. We then describe the dataset, and specify how we apply our descriptor to the problem as well as our experimental setup. Next, we show an example of the use of our descriptor for pedestrian detection and how it relates to HOG, including some example classifications of dataset images. Finally, we present our parameter optimization and test results, which include a comparison with HOG.

6.1 Support vector machines

Suppose we have some training data split into positive and negative elements. In our case these are n-dimensional descriptors of regions with and without pedestrians. We wish to construct a binary classifier which accurately assigns a classification score denoting how confident we are that there is or isn't a pedestrian present. The approach of a linear support vector machine (SVM) is to place a $(n-1)$-dimensional hyperplane in descriptor space which separates the two classes. The hyperplane $(\mathbf{w}, b)$ is defined as

the points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0.$$

If the data is linearly separable, we can search for the *maximum-margin* hyperplane defined as having the largest possible distance to the nearest point of each class. This distance defines the margin. Formally, this can be written as the constrained optimization problem

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{subject to} \qquad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1,$$

where $\mathbf{x}_i$ are training data points and $y_i \in \{-1, 1\}$ their respective condition labels.

However, linear separation of the two classes is almost never possible. Instead a *soft-margin* SVM is used. The idea is to allow some of the data points to be placed on the wrong side of the margin. The total distance of misplaced points to the margin is minimized, in addition to maximizing the margin. This is done by introducing a *cost* parameter C, which weights the importance of the misplacement minimization relative to the margin maximization. Formally, the soft-margin optimization problem is written as

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \right\} \qquad \text{subject to} \qquad \begin{array}{r} y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, \end{array} \tag{6.1}$$

where ξ_i are slack variables allowing for misplaced points.

This formulation can be extended to a non-linear SVM by replacing $\mathbf{w} \cdot \mathbf{x}_i$ with a kernel function $k(\mathbf{w}, \mathbf{x}_i)$. Examples of alternative kernels are polynomials $(\mathbf{w} \cdot \mathbf{x}_i)^d$ for $d \in \mathbb{R}$, and Gaussian radial basis functions $\exp\left(-\gamma \left\|\mathbf{w} - \mathbf{x}_i\right\|^2\right)$ for $\gamma > 0$. Dalal and Triggs [DT05] use a linear SVM and report that using a Gaussian kernel improves the results slightly but at a much higher computation time. Therefore we choose to use a linear SVM as well.

If there is a large skew in the number of elements in each class, the larger class will dominate the misplacement term, and we will be less likely to find a good separation. In order to correct this skew, the cost parameter C can be defined for each of the two classes individually. The simplest approach is dividing C by the number of elements in the respective classes.

When using the SVM for solving the pedestrian detection problem, we use the L_2-regularized L_2 loss support vector classifier from LIBLINEAR v. 1.94[1] [Fan+08]. The naming of this SVM type means that the L_2-norm is used for computing both $\|\mathbf{w}\|$ and ξ_i, as in Equation (6.1).

[1] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

6.2 Performance measures

Having described the method of using an SVM for binary classification, we are now able to define a performance measure for the pedestrian detection problem. Recall that an SVM gives us the signed distances $\mathbf{w} \cdot \mathbf{d} - b$ between each descriptor $\mathbf{d}$ and the hyperplane dividing the two classes: positive for pedestrians and negative for non-pedestrians. Since the SVM is trained to find the best separation between the classes, we can use this distance as the classification score s. Following the binary classification method described in Section 3.8, we are able to compute both the PR- and ROC-curves and their AUCs. In other words we vary a threshold on the distance to the hyperplane to get the two measures of the performance of a descriptor.

When looking for pedestrians in images, we are only interested in the regions where pedestrians present. This is much like matches in image correspondence, where we disregard the negative matches. Thus we follow the same reasoning as in Section 5.3 and use PR AUC as performance measure for our pedestrian detection evaluations. Dalal and Triggs [DT05] use the log-log plot of the ROC-curve as performance measure, and hence we will show test results for this performance measure as well. Since recall is very close to 1 and *FPR* is close to 0, the actual plot is defined as the logarithm of $1 - \text{Recall}$ versus the logarithm of *FPR*.

6.3 Dataset

The dataset, which we use for training and testing our descriptor on the pedestrian detection application, is called the *INRIA Person Dataset*[2] (from now on called the INRIA dataset) created by Dalal and Triggs [DT05]. The purpose of this dataset is to evaluate methods on more challenging images compared to the earlier MIT Pedestrian dataset [PP00].

It consists of various real world images grouped into two subsets: images with pedestrians (positives) and images without pedestrians (negatives). From the positive images, cut-outs of pedestrians are extracted and rescaled to a fixed size, such that the pedestrian in each cut-out is 96 pixels in height from their feet to shoulders. We extract our own cut-outs from the negative images. The size of the cut-outs are 64×128 pixels with a 3 pixel border to avoid boundary condition effects. In case a pedestrian cut-out is smaller than the defined size after resizing, the borders are replicated to achieve the desired dimensions. The positive set only contains images of somewhat upright persons that initially were at least 100 pixels in height. In order to improve the robustness of the dataset against reflected images, each positive cut-out has its horizontally flipped image included as well.

The dataset has been pre-split into training and test data in order to

[2] `http://pascal.inrialpes.fr/data/human/`

compare methods fairly. The training data has a total of 2416 positive and 1218 negative images, while the test data has a total of 1126 positives and 453 negatives.

Figure 6.1 shows examples of positive (a) and negative (b) images from the INRIA dataset. From the positive examples we clearly see the high variety of upright positions and surroundings in the dataset.

6.3.1 Pitfalls and deficiencies

We have found three minor problems with the INRIA dataset, which could affect the results of using the dataset. The first problem is duplicate images. Amongst the negative training images we have found 10 pair-wise duplicates. This could bias the SVM since it effectively means that these images are weighted twice as high as the others. Given the small percentage of duplicates in the dataset the effect should however be close to nothing. We have also found a single duplicated image in the negative test set.

The second problem is the presence of people in some of the negatives. Figure 6.2 show two examples of persons in the negative images of the INRIA dataset. The people are however either too small to fit correctly into the sliding window at any of the used scales or partially occluded by the image bounds to such an extent that we shouldn't be able to detect them anyway.

Finally, some of the positives are originally located near image borders. In order to get images of the correct size, the images are padded by repeating the border pixels. This technique generates synthetic gradients tangent to the border. Such gradients could be learned by the SVM and hence influence the final outcome. This could potentially overstate the actual performance of the classifier.

6.4 Application of descriptors

As in the image correspondence problem, we will evaluate the performance of the GO, SI, GO+SI and HOG descriptors. We also try combining HOG and SI into a HOG+SI descriptor, to see if adding shape index can improve on HOG. Furthermore we will propose a suboptimal compact GO+SI descriptor, which has lower dimensionality than HOG. Our descriptors are using the region cell layouts described in Section 4.2.2, since we are interested in a general region description.

When computing descriptors on images that are larger than the 134×70 pixel regions (including boundary pixels), we use a sliding window (see Section 3.6) with 10 pixel stride and scales at $2^{n/3},\ n = 1, \ldots, m$ where m is the largest number such that the entire window is inside the image.

Dalal and Triggs [DT05] showed that smoothing of the pedestrian images resulted in worse performance than without smoothing. We verify this for

(a) Positives

(b) Negatives

Figure 6.1: Example INRIA images.

Figure 6.2: Examples of pedestrians in negative images.

our descriptors in the parameter study in Section 6.7, and therefore we refrain from smoothing the images in our scale-space pyramids. As mentioned in Section 3.7.3, we use RGB images for HOG. During our parameter study we use grayscale images to decrease the amount of computations, whereas we use RGB images for the final classifier. This choice is verified in the parameter study as well.

6.5 Experimental setup

We have two experimental setups for evaluating descriptors on the INRIA dataset. The first is a validation setup for evaluating parameters during our parameter study using only the training data, and the second is a more complex test setup for evaluating the optimized or alternative descriptors on test data.

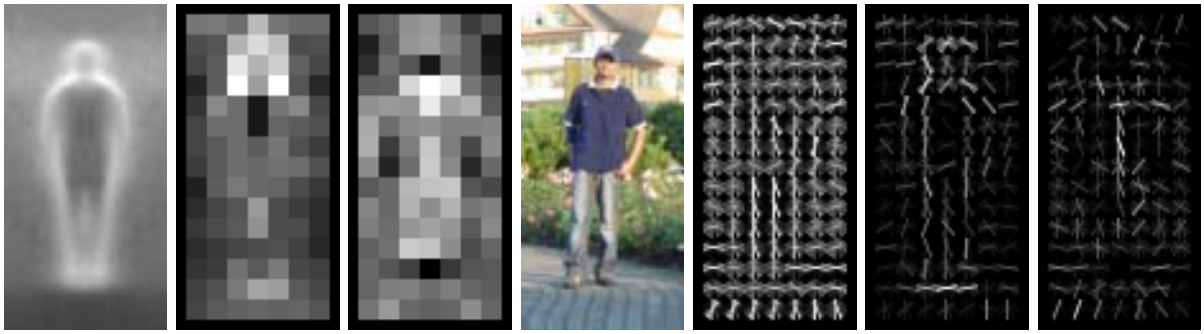
For our validation setup, we split the training data into six parts for the purpose of 6-fold cross-validation. This is done by, for each split, training an SVM on five parts and testing on the other. The SVM is trained using all the given positive images and 40 random cut-outs from each negative image. The results are consolidated by computing the mean PR AUC across the six splits.

Our test setup is a modification of the one described by Dalal and Triggs [DT05]. First, an SVM is trained using all positive training images and 40 random cut-outs from each negative training image. A sliding window is then run across these negative training images, and these windows are classified by the SVM in order to find hard negatives (false positives). The hard negatives are defined as the negative cut-outs with classification score $s > 0$. In other words, we search the negative training images for examples that have not been learned by the SVM from the initial random cut-outs. We add the hard negatives to the set of existing negatives and retrain an SVM with the new training set. This SVM is then used for testing on the positive test images and sliding windows across negative test images.

Dalal and Triggs [DT05] use 10 random cut-outs per image for initial training. We found that using this amount resulted in a high variance of performance when varying the seed for choosing the cut-outs. We therefore extract 40 cut-outs per image instead. The old amounts of training examples were caused by hardware limitations of only having 1.7 GB of RAM for SVM training.

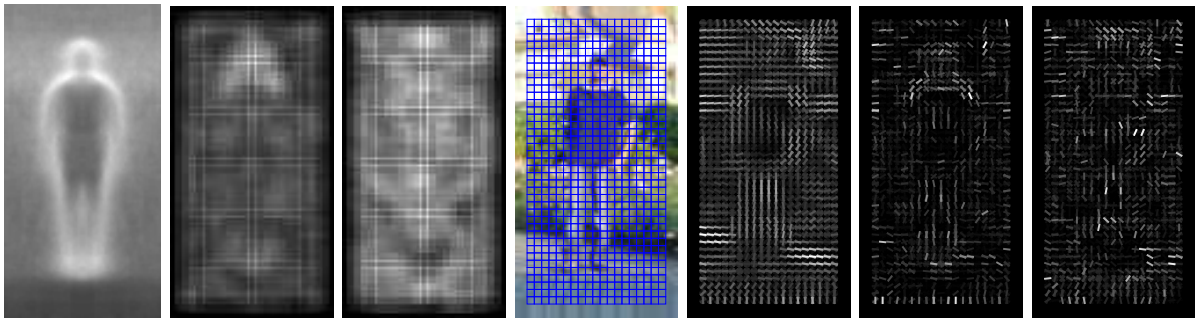
6.6 Example

In this section we will show an example of our proposed GO descriptor computed from a positive pedestrian image. Furthermore we illustrate the connection between the GO descriptor and the weights of an SVM classifier



(a) (b) (c) (d) (e) (f) (g)

Figure 6.3: Example of the HOG descriptor taken from [DT05, Figure 6]. (a) shows the average gradient image over training data. (b) and (c) show the maximal positive and minimal negative SVM weights for each block centered at the pixels, respectively. (d) shows a test image. (e) shows its computed descriptor. (f) and (g) show the descriptor weighted by positive and negative SVM weights, respectively.



(a) (b) (c) (d) (e) (f) (g)

Figure 6.4: Example of our GO descriptor following the same procedure as Figure 6.3. (b) and (c) are however shown for each cell instead of the HOG blocks, and upon (d) we have illustrated out grid layout using the cell spacing r.

trained using the test setup in Section 6.5. Finally, we will show examples of the positive and negative images whose SVM classification we have most and least confidence in.

Figure 6.3 shows a descriptor example from Dalal and Triggs [DT05], and Figure 6.4 shows the corresponding images reproduced for our GO descriptor. Since HOG has each histogram of gradients "repeated" by using block normalization, only the blocks centered at each pixel are illustrated. Our descriptor is illustrated for each cell of the descriptor. The average gradient images over the training data in (a) are similar to one another, as they both are computed using central difference filters without smoothing. Likewise we clearly see the silhouette of a pedestrian in these average images.

(a) True positives

(b) False negatives

(c) True negatives

(d) False positives

Figure 6.5: Example classifications of INRIA cut-outs by our GO descriptor. This shows the types of images our trained SVM is best and worst at classifying.

The maximal positive and minimal negative SVM weights in (b) and (c) differ quite a lot since our descriptor has a much more dense grid. There are however some overall similarities such as large weights around the head and foot regions of the positive weights, and low weights around the foot regions of the negative SVM weights. In other words, structure around the head and foot regions count towards a pedestrian detection.

Having described the SVM weights, we now look at the descriptors in (e) computed from the images in (d). The descriptors weighted by the positive and negative SVM weights are illustrated in (f) and (g) respectively. Both HOG and our descriptor visualize the head, shoulders, and foot regions of the pedestrian in (f). From (g) we see that vertical lines inside the pedestrian count against a pedestrian detection for HOG, whereas there is little to conclude for our descriptor. As mentioned in Section 6.3.1 repeated boundaries could be learned by the SVM. In the visualized descriptor (bottom row) we easily see the repeated border in the bottom of the image, and a couple of these contribute quite a lot to the positive SVM.

To get an idea of the types of images our trained SVM is best and worst at classifying, we find some of the pedestrian and non-pedestrian images with the highest and lowest classification scores. We call the pedestrian images with high classification scores true positives, as we are more likely to classify them correctly. Whether we do so of course depends on the exact threshold chosen. Similar statements can be said for false positives, true negatives, and false negatives. The images are shown in Figure 6.5.

Generally the true positive images show pedestrians in standard up-right poses with the front or back to the camera. The clothing is clearly distinctive from the background, and there are little to no occlusions. Furthermore 6 of the 10 true positives illustrated here have a repeated bottom boundary. The false negatives are quite the opposite: many occlusions, people facing sideways, and some dim lighting. The true negative examples are all sky with very little gradient structure. The false positives are quite varied; some have structure around the possible foot or head region, but it is not obvious to see how they could be classified as pedestrians.

6.7 Parameter study

We follow the same setup as our image correspondence parameter study in Section 5.8, with a couple exceptions. Since the dataset has a pre-defined split into training and testing data, we only run the parameter study once (with 6-fold cross validation) on training data and obtain a single set of parameters. The parameters are slightly different from the interest point cell layout parameters, and we additionally have the SVM-weight C, which we also wish to optimize for each descriptor.

The optimal parameters for our GO and SI descriptors are shown in Ta-

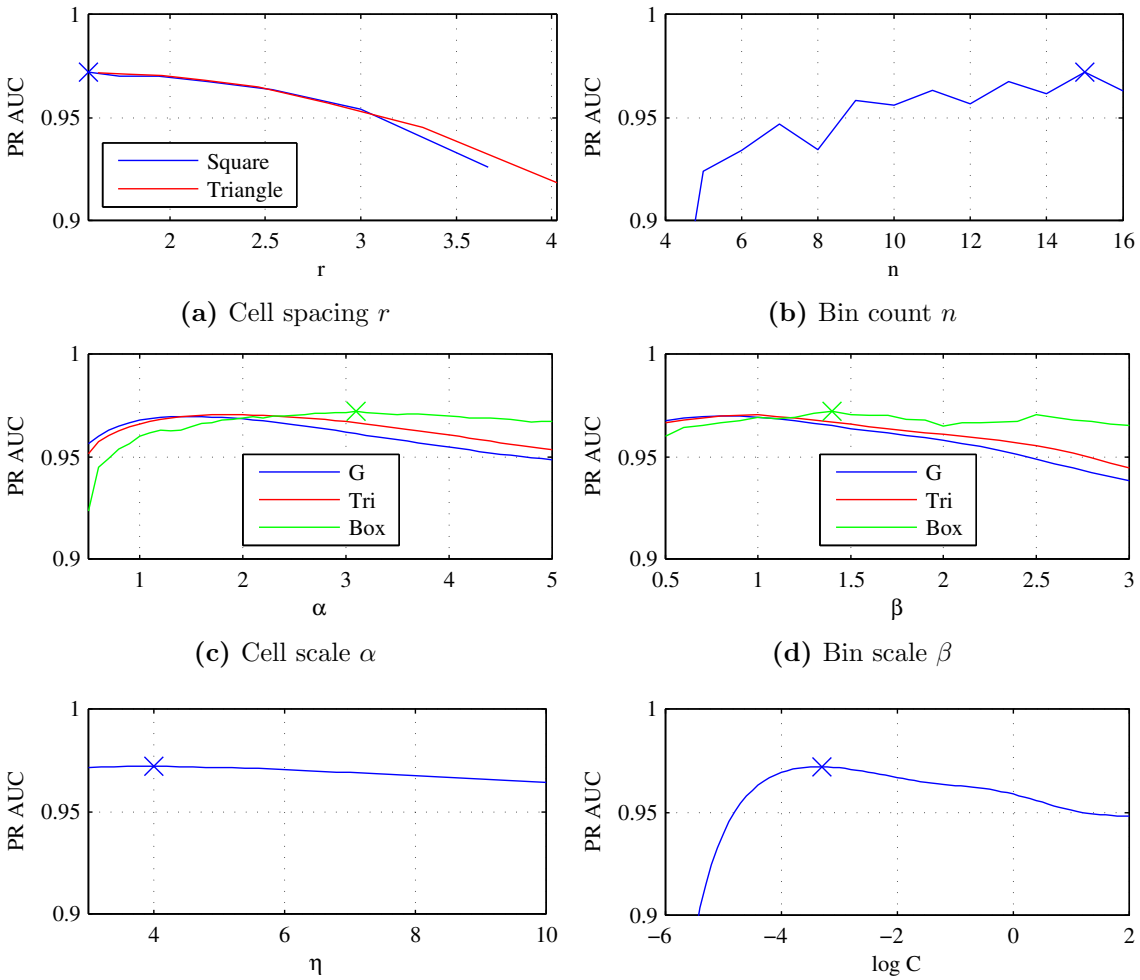
	Methods		
Parameter	GO	SI	HOG
Grid type	Square	Triangle	Square
Cell spacing r	1.57	1.62	4
Cell kernel	*Box*	*Box*	*Tri*
Cell scale α	3.1	3.5	1.0
Bin count n	15	13	$9 + 18$
Bin kernel	*Box*	*Tri*	*Tri*
Bin scale β	1.4	0.8	1.0
Normalization scale η	4.0	5.0	-

Table 6.1: Optimized parameters from our parameter study for GO and SI. We also include the HOG parameters translated into our framework when possible. HOG has 9 undirected and 18 directed orientation bins, and uses a block normalization scheme instead of our pixel-wise normalization.

ble 6.1, together with the parameters for HOG translated into our framework for easy comparison. The optimal C values for all descriptors are shown in Table 6.2. The effect of adjusting the parameters relative to the optimized parameters can be seen for GO in Figure 6.6 and SI in Figure 6.8. Cell spacing r and bin count n seem to be the two most important parameters: Decreasing r improves performance greatly, but the increased number of cells also results in a much higher dimensionality. For GO, n seems to produce better results for high odd numbers than the neighbouring even numbers, whereas 9 or 13 bins produce good results for SI. The optimized r values are the lowest in our search range for each grid type, because the dimensionality at this point is already almost too high to work with. Most of the other parameters have a large range of values that produce good results. Box filters marginally outperform the alternative kernels, except for the bin kernel of SI, where the three kernels perform almost equally well, and the triangle kernel is chosen.

Due to the high dimensionality we construct a compact version of our GO+SI descriptor based on these training results. We keep the same optimal parameters, except we increase cell spacing $r = 2.54$ in a square grid, and we reduce bin count to $n = 13$ for GO and $n = 5$ for SI. This reduces the dimensionality below HOG. For our HOG+SI descriptor, we reduce $r = 2.18$ and $n = 9$ to reduce the influence of the added shape index information compared to HOG. We will go into further detail about the descriptor dimensionality and performance in the results section.

We have also experimented with various other parts of our descriptor

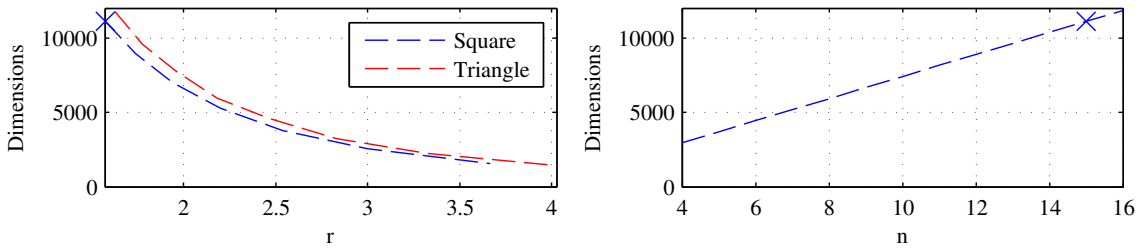


(a) Cell spacing r **(b)** Bin count n

(c) Cell scale α **(d)** Bin scale β

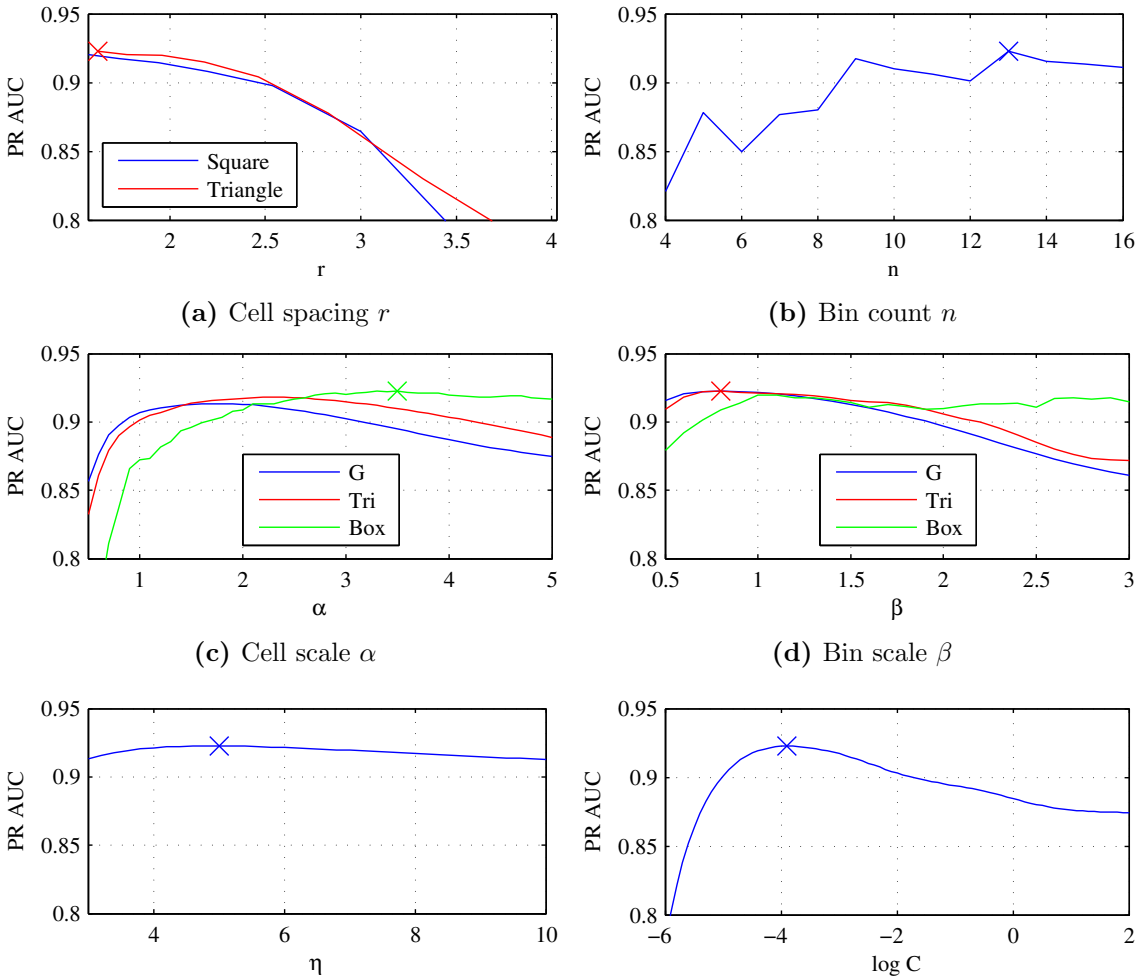
(e) Normalization scale η **(f)** SVM weight C

Figure 6.6: Average classification performance on training data for GO when adjusting the parameters. Optimal parameter values are marked with a cross.



(a) Cell spacing r **(b)** Bin count n

Figure 6.7: Dimensionality of GO when adjusting the parameters.

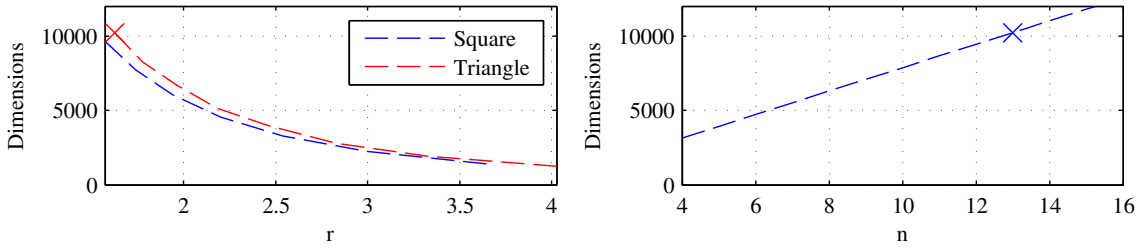


(a) Cell spacing r **(b)** Bin count n

(c) Cell scale α **(d)** Bin scale β

(e) Normalization scale η **(f)** SVM weight C

Figure 6.8: Average classification performance on training data for SI when adjusting the parameters. Optimal parameter values are marked with a cross.



(a) Cell spacing r **(b)** Bin count n

Figure 6.9: Dimensionality of SI when adjusting the parameters.

	Methods					
Parameter	GO	SI	GO+SI	Compact GO+SI	HOG	HOG+SI
$\log C$	$6.3 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$6.3 \cdot 10^{-4}$	$4.0 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$

Table 6.2: Optimized SVM weight C for GO, SI, GO+SI, Compact GO+SI, HOG and HOG+SI.

that are not listed as parameters. We report some of these results for our optimal GO descriptor on training data, which has a mean PR AUC of 0.972: Normalizing the final descriptors before training the SVMs reduces the AUC by 0.539. Computing the gradients on RGB colour channels and taking the largest magnitude improves the AUC by 0.003. Omitting pixel-wise normalization reduces the AUC by 0.066. Smoothing images at scale $\sigma = 1$ reduces the AUC by 0.015. These tendencies hold for SI as well.

6.8 Results

Now that we have computed optimal parameters on the training data, we can evaluate the performance of the descriptors on test data. As mentioned before, we compare our descriptor to HOG, which originally was trained and tested on the same INRIA dataset [DT05]. Recall that we use RGB images for testing.

Table 6.3 shows a test performance summary for the six descriptors, their dimensionality, and amount of hard negatives added for retraining. We also include the test performance (initial PR AUC) when omitting retraining to see the effect of this step. Based on PR AUC the ordering of the descriptors from best to worst is: HOG+SI, HOG, compact GO+SI, GO+SI, GO, SI. The same ordering holds for recall at 10^{-4} FPR, which is the measure used by Dalal and Triggs [DT05], whereas HOG and compact GO+SI switch positions when looking at ROC AUC.

Overall HOG outperforms our descriptors, but the addition of shape index is shown to be valuable for both HOG and our descriptors. The SI descriptor performs very poorly on its own. The difference between compact GO+SI and GO+SI is negligible despite the large difference in dimensionality.

By comparing the PR AUCs with and without retraining of hard negatives, we see that all descriptors benefit from retraining. Interestingly, compact GO+SI starts off worse than the much larger GO+SI, but it overtakes after retraining by adding more than twice the amount of hard negatives. This will be discussed in Section 7.2.

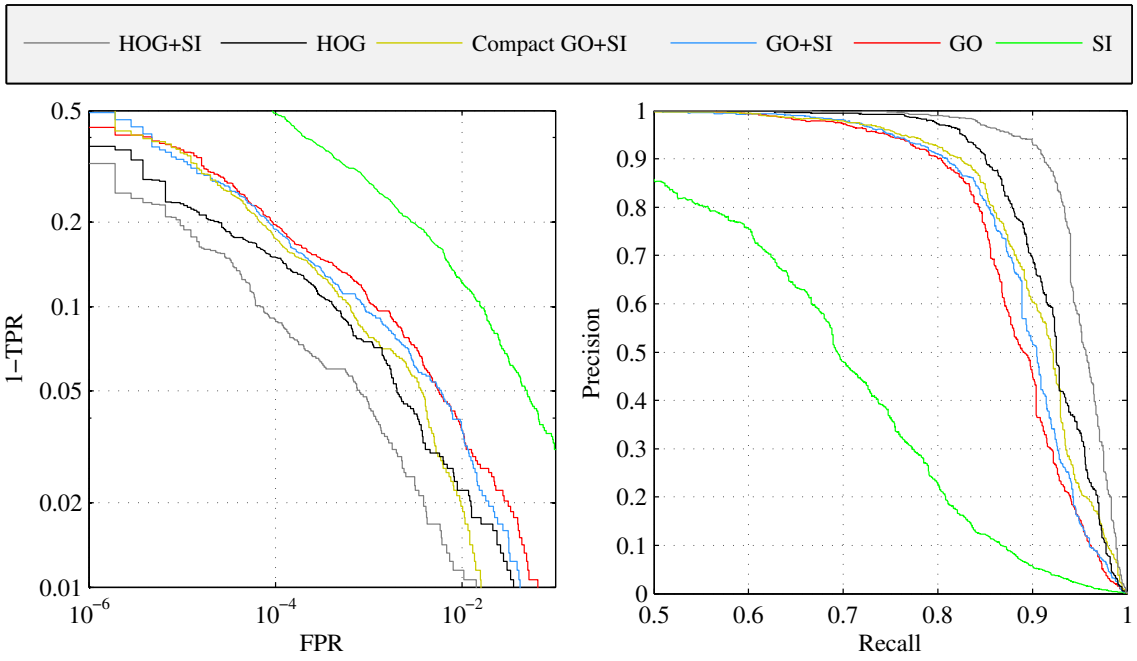
Figure 6.10 shows ROC- and PR-curves of the various descriptors. Both

Measure	HOG+SI	HOG	Compact GO+SI	GO+SI	GO	SI
Dimensions	8316	4743	4554	21346	11115	10231
Hard negatives	10703	24387	28528	12705	24547	75033
Initial PR AUC	0.922	0.903	0.839	0.877	0.833	0.608
PR AUC	0.951	0.919	0.902	0.891	0.881	0.680
ROC AUC	0.9990	0.9980	0.9983	0.9973	0.9971	0.9888
Recall at 10^{-4} FPR	0.910	0.850	0.824	0.811	0.804	0.511

Table 6.3: Summary of test results on the INRIA dataset. The descriptors are ordered from best to worst. For each descriptor we report the dimensionality, number of hard negatives added to the training data, test PR AUC without added hard negatives, and test PR AUC, ROC AUC, and recall at 10^{-4} FPR after the hard negatives are added. The latter measure is the one used by Dalal and Triggs [DT05].

curves show roughly the same ordering of descriptor performance as the PR AUCs. However the compact GO+SI, GO+SI, and GO curves cross each other multiple times, and hence their ordering depends on the chosen threshold t. The HOG results shown here are not quite as good as those presented by Dalal and Triggs [DT05]. This may be due to the VLFeat implementation of HOG, SVM differences, or our sliding window implementation.

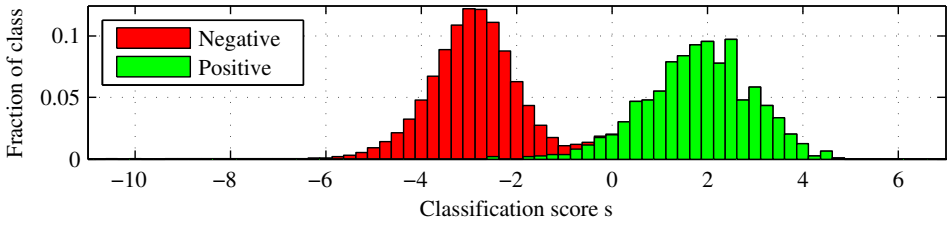
Figure 6.11 illustrates the distribution of classification scores for positive and negative images. Each class is normalized with respect to the amount of images in the class. The graphs give a more intuitive understanding of the performance of the descriptors. The amount of overlap between the two classes illustrates the separability of each classifier and the effect of choosing a threshold t.



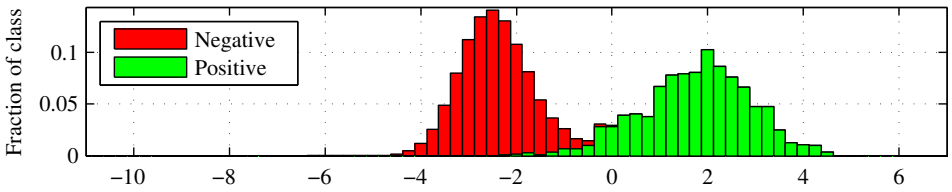
(a) Log-log ROC-curve

(b) PR-curve

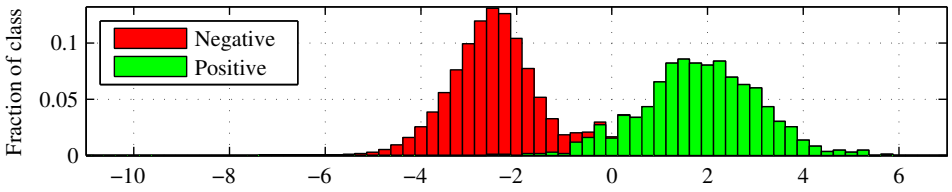
Figure 6.10: ROC- and PR-curve of the test results on the INRIA dataset. The ROC-curve is plotted in the same range as Dalal and Triggs [DT05], and the PR-curve is plotted starting from recall 0.5 for better visualization.



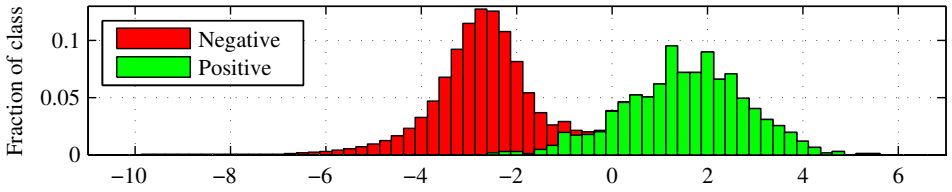
(a) HOG+SI



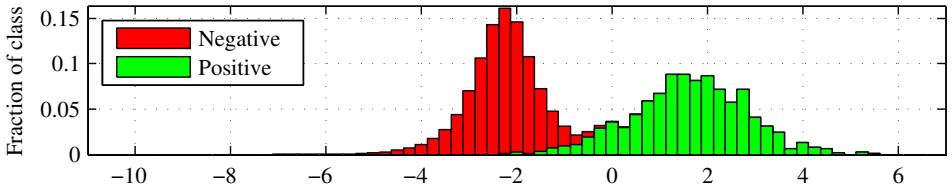
(b) HOG



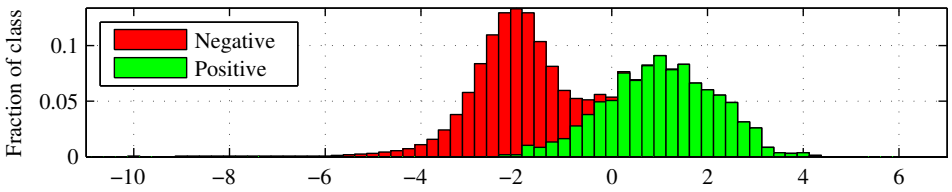
(c) Compact GO+SI



(d) GO+SI



(e) GO



(f) SI

Figure 6.11: Histograms of classification scores on INRIA test data for various descriptors. The negative and positive results are individually normalized due to the large skew in amount of images.

Chapter 7

Discussion

Recall that we have developed the two main descriptors: GO, which uses gradient information similarly to SIFT and HOG, and SI, which uses second order information in the form of shape index and curvedness. Furthermore we have combined the two descriptors into a GO+SI descriptor. The descriptors have a large number of parameters, which can be modified to fit the application. We have optimized the parameters for each of the two applications: image correspondence and pedestrian detection. In these applications we compare the final descriptor test results against the SIFT and HOG descriptors respectively.

In this chapter we first discuss our optimal descriptors and the results on each of the two applications. Then we discuss the differences between the applications, and finally we propose and discuss future work.

7.1 Image correspondence

Recall that the image correspondence evaluation shows that our descriptors are very close performance-wise, and marginally outperform SIFT. The difference is larger for light variations and statistically significant with 95% confidence in a few cases. However these statistic rely on the assumption that the PR AUCs are normally distributed for each position as described in Section 5.9. This assumption might not hold since the PR AUCs are bounded between 0 and 1, and hence the tails will be heavily clipped when the mean is close to either bound.

With the test results in mind, we now compare and reflect on the different design choices of SIFT and our descriptor. Recall that Table 5.4 shows our parameter study results, including corresponding parameters for SIFT.

Where SIFT uses vector normalization and clipping for illumination invariance and robustness, we use pixel-wise and vector normalization. The choice of using local illumination estimation and normalization is most likely the reason for the difference in performance under light variations.

Another difference between SIFT and our descriptor is that we do not support rotation estimation. The DTU dataset images have fixed orientation, and hence we have omitted this step from SIFT as mentioned in Section 5.5. The reasoning for this is to avoid a performance decrease due to wrong rotation estimation. The rotation estimation of SIFT could easily be adopted by our descriptor as an initial step for rotation invariance. Our cell layouts would then be rotated according to the estimated orientation.

The cell layouts of SIFT and our descriptors differ greatly. Our descriptors are based on polar grids such as GLOH and DAISY, whereas SIFT uses a square grid. Assuming that structure close to the interest point is more important than structure further away, we see two advantages of our grids. The corner cells of SIFT are further away than the other edge cells, while our cells in each ring have equal distance to the interest point. The SIFT grid contains more outer-cells than inner-cells, while our rings have have an equal amount of cells. This gives an even distribution of cells in both angular and radial direction. According to Cui et al. [Cui+09] polar grids are more robust against scale estimation errors, and hence this should apply to our descriptors as well. Furthermore we believe that our grids are be more robust against rotation estimation errors, since the cell areas would overlap more with the correctly rotated grid than for a squared grid. This will however need to be tested before coming to a conclusion.

We see that we have more cells and bins than SIFT resulting in a descriptor with larger dimensionality. This is related to the fact that our descriptor cells are spread across a larger area, and thus the amount of samples in each cell bin is closer to that of SIFT than the dimensionality suggests.

Our initial idea was to base our cell histograms on locally orderless images [Kv99]. This work is based on Gaussian kernels for bin and cell aperture functions rather than the triangle kernels of SIFT. Our parameter study has shown that the two kernels produce almost identical results, but the triangle kernels are less expensive to compute in practice. This is due to the complexity of the Gaussian kernels and their infinite support, which we limit to 3β, while the simple triangle kernels only have 2β support (see Section 3.3.1).

The center aperture function, which SIFT likewise uses, only has a minor influence on the performance of our descriptor. Our performance gain from this function is most likely not as high as for SIFT, since our cell layouts already favour the structure closer to the interest point.

From our test results we see that the addition of SI to GO only increases performance marginally. The addition however adds a substantial number of dimensions to the descriptor. Given the tiny performance increase, GO is the descriptor from our framework that we recommend.

7.2 Pedestrian detection

In the pedestrian detection application, our optimal parameters yield a very high dimensionality compared to HOG. We therefore introduced the compact GO+SI descriptor, which is constructed from picking slightly worse parameters than the optimal, in order to achieve a lower dimensionality than HOG. It turned out to produce slightly better results than the GO+SI descriptor, which has much higher dimensionality. The pedestrian detection evaluation shows that our descriptors perform worse than HOG, but with our compact GO+SI close behind. SI does not work well on its own, but adding it to both GO and HOG increases performance.

With the test results in mind, we now compare and reflect on the different design choices of HOG and our descriptor. Recall that Table 6.1 shows our parameter study results, including corresponding parameters for HOG.

A big difference between our descriptors and HOG is in the strategy of local normalization. An advantage of HOG's block normalization is that each cell contains four different normalization factors for different directions away from the cell. This information is different from the histogram bins, because it is based on relative magnitudes rather than orientations. An advantage of our pixel-wise normalization is that the normalization region is defined per pixel instead of per block. This allows for pixels inside a cell to be normalized individually, where HOG normalizes them equally. Interestingly, our optimal normalization region for GO defined by a Gaussian window with $\eta = 4$ is similar in size to the four overlapping 8×8 pixel block normalizations from HOG.

We both have a larger density and cell overlap compared to HOG. This results in a large number of cells that are slightly larger than HOG's. Assuming a fixed number of cells, increasing the overlap between cells is a trade-off between two factors. When specific gradient information is spread out to multiple cells, the robustness to small changes in pedestrian postures is increased. However, this also reduces the influence of the information to each cell. This causes the classification ability of easily recognisable pedestrians to decrease, but a larger range of harder pedestrians to score well. The same reasoning can be said for overlap between histogram bins.

The initial PR AUCs shown in Table 6.3 indicate that GO+SI is significantly better than compact GO+SI. Retraining on additional hard negatives removes this performance gap, and compact GO+SI takes a slight lead. Taking the non-optimal parameters into account this is surprising, and shows that including the retraining step in the parameter study may have improved performance. This would however have greatly increased the run time of the optimization. We can think of two possible reasons that explain this effect: Firstly, compact GO+SI has much lower dimensionality than GO+SI and therefore each added hard negative has a larger impact on the classifier for compact GO+SI than for GO+SI. Secondly, the amount of hard negatives

added for retraining of the compact GO+SI classifier is 2.2 times higher than for GO+SI.

The dimensionality and test results of GO+SI and compact GO+SI suggest that GO+SI contains redundant information. The critical difference between the two descriptors is the density of cells and bins. The density of GO+SI causes the dimensionality to supersede the number of pixels for a single descriptor in the scale-space images. Generally we want the descriptors to represent the essential parts of a region and thus be smaller than the raw data. This is the case for compact GO+SI and HOG.

To recap, HOG is still superior to compact GO+SI, but compact GO+SI is the descriptor from our framework that we recommend. HOG however gains performance from adding SI.

7.3 Differences between applications

When starting the project, we thought about creating a descriptor which generalized well to different applications. However as seen in Sections 5.8 and 6.7 we need both different design choices and different parameters for the image correspondence and pedestrian detection applications.

Firstly, we had to construct vastly different cell layouts depending on the use of an interest point detector or a sliding window. Secondly, both smoothing of scale-space images and normalization of the final descriptor add a minor improvement to image correspondence, while greatly decreasing performance for pedestrian detection. Finally, most of the optimal parameters for pedestrian detection differ significantly from image correspondence: For example the normalization scale and the overlap between cells is much larger.

Some of the differences can be explained by the difference in the nature of the problems. In image correspondence we try to create distinctive descriptors for each individual feature, whereas in pedestrian detection we try to generalize the class of pedestrians. For example colour information is not a good indicator in pedestrian detection, but it has proven useful for finding image correspondences.

In image correspondence the addition of shape index was ineffective, whereas it proved useful in pedestrian detection. We suspect that this is due to difference in descriptor comparison methods between the two applications. For image correspondence a simple Euclidean distance between two descriptors is used and weights all dimensions equally. This can affect performance negatively if some SI dimensions are less descriptive. For pedestrian detection an SVM is used for classifying descriptors. The SI dimensions that are less descriptive will be assigned a low weight and hence don't affect performance too negatively, whereas the more descriptive SI dimensions will have higher weights and boost performance.

7.4 Future work

Our pixel-wise normalization scheme, while significantly improving results over no normalization, has the downside of boosting noise in dark areas. An example of this can be seen in Figure 4.1. It may be possible to create a clever scheme to keep the increased robustness to local illumination changes while removing the noise from dark areas.

We decided to resize scale-space images respective to their scales, such that histograms are computed across the same pixel area regardless of detection scale. This means that potentially a lot of information is lost from downsampling the large scale images. It is possible to omit the resizing step and compute histograms at their original scales, but while experimenting with this we quickly ran into time and memory problems. The problem is that even medium size features at $\sigma = 8$ multiply the computation time and memory needed by $8^2 = 64$. Another problem is whether the bin scales should be compensated for changing the number of pixels in histograms. While the approach was infeasible for us to use, there may be ways to overcome these issues.

A possible issue with constructing our shape index histograms is the non-uniformity of the distribution of shape indices in natural images, which is reported in [LG09, Fig. 3]. This figure shows large peaks at what is equivalent to $S = \pm 1/2$ and very few shape indices around $S = \pm 1$, which is consistent with our own findings, e.g. Figure 4.12a. The result of this is that some of the histogram bins have a very little effect on our distance measure. One could investigate ways of adjusting the histograms to be more uniform, possibly by moving the bin positions and scales and/or weighting them differently.

As mentioned in Section 5.8, our scheme for optimizing parameters risks getting stuck in local optima. We defined our parameters to be as uncorrelated as possible, but it could still be the case that modifying several of the parameters together would give a better result. The initialization of the parameters could likewise give a suboptimal solution. A possible way to improve our parameter optimization, while still being able to evaluate it in a reasonable time frame, is a simulated annealing approach. This is a probabilistic method designed to find approximated global optima in a large search space, by trying nearby random parameter choices with gradually smaller jumps. Another potential improvement to our parameter study could be to optimize the combined GO+SI instead of optimizing GO and SI individually.

There are also many approaches used in the literature that could potentially improve our descriptor. PCA could be used to reduce the dimensionality, allowing for a higher number of cells and bins. Alternatives to the shape index and curvedness as mentioned in Section 4.1 could be thoroughly tested as well as higher than second order differential structure for

use as bin value and magnitude functions. We could extend our descriptor to represent structure from multiple scales like the galaxy descriptor Pedersen et al. [Ped+13]. This is however not necessarily a good idea in combination with omitting smoothing for pedestrian detection, since the information will largely be the same. Two-way matching, where we consider the distance ratio between best matches in both images, could improve image correspondence results. However, it is not important for comparing the performance of descriptors to each other. In order to improve upon the pedestrian detection results, we could use both higher level descriptors such as bag of visual words [Csu+04] and Fisher vectors [Sán+13], as well as improve upon the detection algorithm by e.g. introducing deformable part models [FMR08].

Chapter 8

Conclusion

In our work we have studied the literature of descriptors and developed our descriptor framework. Based on this framework we have proposed and optimized our own descriptors for the image correspondence and pedestrian detection problems. Our descriptors have been evaluated on the DTU Robot 3D and INRIA Person datasets and compared against the popular SIFT [Low04] and HOG [Fel+10] descriptors respectively.

Our descriptor framework is inspired by locally orderless images [Kv99], Parzen windows [Par62], SIFT based gradient orientation histograms, and the galaxy descriptor's [Ped+13] shape index histograms. The main aspects that separate our descriptors from SIFT and HOG are our choices of using shape index, spatial layout of histograms, kernels used for aggregation of image information, and method for handling local illumination. In our parameter studies we found that the nature of the two applications affect the optimal descriptor parameters significantly.

For image correspondence our descriptors perform marginally better than SIFT, and the difference is significant for our GO and GO+SI descriptors in some cases of light variation. The addition of shape index to gradient orientation histograms has not proven important, and hence for this application GO is the recommended descriptor from our framework.

For pedestrian detection the performance of our descriptors is lower than HOG. Using shape index on its own produces bad results, but adding it to gradient orientation histograms does improve performance. This is the case for both our descriptors as well as HOG. Our compact GO+SI descriptor, which has lower dimensionality and slightly worse performance than HOG, is the recommended descriptor from our framework for this application.

Appendix A

Histogram renormalization

This appendix chapter computes the equations needed to renormalize the bin aperture functions used in our histograms. We compute definite integrals from a to b of the box, triangle, and Gaussian kernel functions.

A.1 Box kernel

The box kernel with bin scale β is defined as:

$$Box(x;\beta) = \begin{cases} \frac{1}{2\beta}, & \text{if } |x| \leq \beta \\ 0, & \text{otherwise} \end{cases}$$

The kernel is only non-zero between $-\beta$ and β. Assuming $-\beta \leq a, b \leq \beta$,

$$\begin{aligned} \int_a^b Box(x;\beta)\,\mathrm{d}x &= \int_a^b \frac{1}{2\beta}\,\mathrm{d}x \\ &= \frac{1}{2\beta}(b-a) \end{aligned}$$

A.2 Triangle kernel

The triangle kernel with bin scale β is defined as:

$$Tri(x;\beta) = \begin{cases} \frac{1}{2\beta}\left(1 - \frac{|x|}{2\beta}\right), & \text{if } |x| \leq 2\beta \\ 0, & \text{otherwise} \end{cases}$$

The kernel is only non-zero between -2β and 2β. Assuming $-2\beta \leq a \leq 0 \leq b \leq 2\beta$,

$$\int_a^b Tri(x;\beta)\,\mathrm{d}x = \frac{1}{2\beta}\left(\int_a^0 \left(1 + \frac{x}{2\beta}\right)\,\mathrm{d}x + \int_0^b \left(1 - \frac{x}{2\beta}\right)\,\mathrm{d}x\right)$$

$$= \frac{1}{2\beta}\left(-a - \frac{a^2}{4\beta} + b - \frac{b^2}{4\beta}\right)$$
$$= \frac{1}{2\beta}(b-a) - \frac{1}{8\beta^2}(a^2+b^2)$$

A.3 Gaussian kernel

The Gaussian kernel with bin scale β is defined as:

$$G(x;\beta) = \frac{1}{\sqrt{2\pi}\beta}\exp\left(-\frac{x^2}{2\beta^2}\right)$$

The definite integral of $G(x;\beta)$ from a to b cannot be expressed by an elementary function. Instead we can use the non-elementary error function, defined as $\mathrm{erf}(b) = \frac{2}{\sqrt{\pi}}\int_0^b \exp(-x^2)\,\mathrm{d}x$, which can be computed by its Taylor series. First we show an intermediate result using the substitution $u = \sqrt{k}x$ with $\mathrm{d}u = \sqrt{k}\mathrm{d}x$:

$$\begin{aligned}\int_0^b \exp(-kx^2)\,\mathrm{d}x &= \frac{1}{\sqrt{k}}\int_0^{\sqrt{k}b}\exp(-u^2)\,\mathrm{d}u \\ &= \frac{1}{\sqrt{k}}\frac{\sqrt{\pi}}{2}\mathrm{erf}\left(\sqrt{k}b\right)\end{aligned}$$

We use this to compute $\int_a^b G(x;\beta)\,\mathrm{d}x$:

$$\begin{aligned}\int_a^b G(x;\beta)\,\mathrm{d}x &= \frac{1}{\sqrt{2\pi}\beta}\int_a^b \exp\left(-\frac{x^2}{2\beta^2}\right)\,\mathrm{d}x \\ &= \frac{1}{\sqrt{2\pi}\beta}\left(\int_0^b \exp\left(-\frac{x^2}{2\beta^2}\right)\,\mathrm{d}x - \int_0^a \exp\left(-\frac{x^2}{2\beta^2}\right)\,\mathrm{d}x\right) \\ &= \frac{1}{\sqrt{2\pi}\beta}\frac{\sqrt{\pi}}{2}\sqrt{2}\beta\left(\mathrm{erf}\left(\frac{1}{\sqrt{2}\beta}b\right) - \mathrm{erf}\left(\frac{1}{\sqrt{2}\beta}a\right)\right) \\ &= \frac{1}{2}\left(\mathrm{erf}\left(\frac{b}{\sqrt{2}\beta}\right) - \mathrm{erf}\left(\frac{a}{\sqrt{2}\beta}\right)\right)\end{aligned}$$

Appendix B

Image transformations

Recall the definitions of gradient orientation Θ, gradient magnitude M, shape index S, and curvedness C defined in Section 3.2. This appendix chapter computes the effect of certain image transformations on the mentioned functions.

B.1 Rotation

Let $\widetilde{L}$ be the blurred image L rotated around some point $\mathbf{x}$ by angle θ. That is, we rotate the original coordinate system axes $\mathbf{e}_x, \mathbf{e}_y$ by multiplying by a rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$
$$\mathbf{R}\mathbf{e}_x = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$
$$\mathbf{R}\mathbf{e}_y = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

Now the first and second order derivatives can be computed by directional derivatives along the coordinate system axes:

$$\begin{aligned}
L_i &= \nabla_{\mathbf{e}_i} L \\
\widetilde{L}_i &= \nabla_{\mathbf{R}\mathbf{e}_i} L = (\mathbf{R}\mathbf{e}_i)^T \nabla L \\
\widetilde{L}_x &= \cos\theta L_x + \sin\theta L_y \\
\widetilde{L}_y &= -\sin\theta L_x + \cos\theta L_y \\
L_{ij} &= \nabla_{\mathbf{e}_i}(\nabla_{\mathbf{e}_j} L) \\
\widetilde{L}_{ij} &= \nabla_{\mathbf{R}\mathbf{e}_i}(\nabla_{\mathbf{R}\mathbf{e}_j} L) = \nabla_{\mathbf{R}\mathbf{e}_i}\left((\mathbf{R}\mathbf{e}_j)^T \nabla L\right) = (\mathbf{R}\mathbf{e}_j)^T \nabla^2 L (\mathbf{R}\mathbf{e}_i)
\end{aligned}$$

$$
\begin{aligned}
\widetilde{L}_{xx} &= \cos^2\theta L_{xx} + \sin^2\theta L_{yy} + 2\cos\theta\sin\theta L_{xy} \\
&= \cos^2\theta L_{xx} + \sin^2\theta L_{yy} + \sin 2\theta L_{xy} \\
\widetilde{L}_{yy} &= \sin^2\theta L_{xx} + \cos^2\theta L_{yy} - 2\cos\theta\sin\theta L_{xy} \\
&= \sin^2\theta L_{xx} + \cos^2\theta L_{yy} - \sin 2\theta L_{xy} \\
\widetilde{L}_{xy} &= -\cos\theta\sin\theta L_{xx} + (\cos^2\theta - \sin^2\theta) L_{xy} + \cos\theta\sin\theta L_{yy} \\
&= -\frac{1}{2}\sin 2\theta (L_{xx} - L_{yy}) + \cos 2\theta L_{xy}
\end{aligned}
$$

B.1.1 Gradient orientation

First we show an intermediate result using a common tangent identity:

$$
\begin{aligned}
\tan\left(\tan^{-1}\left(\frac{L_y}{L_x}\right) - \theta\right) &= \frac{\tan\left(\tan^{-1}\left(\frac{L_y}{L_x}\right)\right) - \tan\theta}{1 + \tan\left(\tan^{-1}\left(\frac{L_y}{L_x}\right)\right)\tan\theta} \\
&= \frac{\frac{L_y}{L_x} - \frac{\sin\theta}{\cos\theta}}{1 + \frac{L_y}{L_x}\frac{\sin\theta}{\cos\theta}} \\
&= \frac{\left(\frac{-\sin\theta L_x + \cos\theta L_y}{L_x\cos\theta}\right)}{\left(\frac{\cos\theta L_x + \sin\theta L_y}{L_x\cos\theta}\right)} \\
&= \frac{-\sin\theta L_x + \cos\theta L_y}{\cos\theta L_x + \sin\theta L_y}
\end{aligned}
$$

We use this to compute the transformed gradient orientation $\widetilde{\Theta}$:

$$
\begin{aligned}
\widetilde{\Theta} &= \tan^{-1}\left(\frac{\widetilde{L}_y}{\widetilde{L}_x}\right) \\
&= \tan^{-1}\left(\frac{-\sin\theta L_x + \cos\theta L_y}{\cos\theta L_x + \sin\theta L_y}\right) \\
&= \tan^{-1}\left(\frac{L_y}{L_x}\right) - \theta = \Theta - \theta
\end{aligned}
$$

B.1.2 Gradient magnitude

We here compute the transformed gradient magnitude $\widetilde{M}$:

$$
\begin{aligned}
\widetilde{M} &= \sqrt{\widetilde{L}_x^2 + \widetilde{L}_y^2} \\
&= \sqrt{(\cos\theta L_x + \sin\theta L_y)^2 + (-\sin\theta L_x + \cos\theta L_y)^2} \\
&= \sqrt{L_x^2 + L_y^2} = M
\end{aligned}
$$

B.1.3 Shape index

Define $S_n = -L_{xx} - L_{yy}$ and $S_d = 4L_{xy}^2 + (L_{xx} - L_{yy})^2$ such that the shape index is defined as $S = \frac{2}{\pi} \tan^{-1}\left(S_n / \sqrt{S_d}\right)$.

We equivalently define the transformed nominator $\widetilde{S}_n$ and denominator $\widetilde{S}_d$ to show that $\widetilde{S}_n = S_n$ and $\widetilde{S}_d = S_d$:

$$\begin{aligned}
\widetilde{S}_n &= -\widetilde{L}_{xx} - \widetilde{L}_{yy} \\
&= -(\cos^2\theta + \sin^2\theta)L_{xx} - (\cos^2\theta + \sin^2\theta)L_{yy} \\
&= -L_{xx} - L_{yy} = S_n \\
4\widetilde{L}_{xy}^2 &= 4(-\frac{1}{2}\sin 2\theta(L_{xx} - L_{yy}) + \cos 2\theta L_{xy})^2 \\
&= (-\sin 2\theta(L_{xx} - L_{yy}) + 2\cos 2\theta L_{xy})^2 \\
(\widetilde{L}_{xx} - \widetilde{L}_{yy})^2 &= ((\cos^2\theta - \sin^2\theta)(L_{xx} - L_{yy}) + 2\sin 2\theta L_{xy})^2 \\
&= (\cos 2\theta(L_{xx} - L_{yy}) + 2\sin 2\theta L_{xy})^2 \\
\widetilde{S}_d &= 4\widetilde{L}_{xy}^2 + (\widetilde{L}_{xx} - \widetilde{L}_{yy})^2 \\
&= (\sin^2 2\theta - \cos^2 2\theta)(L_{xx} - L_{yy})^2 + 4(\cos^2 2\theta + \sin^2 2\theta)L_{xy}^2 \\
&= 4L_{xy}^2 + (L_{xx} - L_{yy})^2 = S_d
\end{aligned}$$

And thus $\widetilde{S} = S$.

B.1.4 Curvedness

Let $\widetilde{L}_{xx}$ and $\widetilde{L}_{yy}$ be the transformed L_{xx} and L_{yy}, and $\widetilde{C}$ be the transformed curvedness.

$$\begin{aligned}
\widetilde{L}_{xx}^2 + \widetilde{L}_{yy}^2 &= (\cos^4\theta + \sin^4\theta)(L_{xx}^2 + L_{yy}^2) + 2\sin^2 2\theta L_{xy}^2 \\
&\quad + \sin^2 2\theta L_{xx}L_{yy} + \sin 4\theta(L_{xx} - L_{yy})L_{xy} \\
2\widetilde{L}_{xy}^2 &= \frac{1}{2}\sin^2 2\theta(L_{xx} - L_{yy})^2 + 2\cos^2 2\theta L_{xy}^2 - \sin 4\theta(L_{xx} - L_{yy})L_{xy} \\
\widetilde{L}_{xx}^2 + \widetilde{L}_{yy}^2 + 2\widetilde{L}_{xy}^2 &= (\cos^4\theta + \sin^4\theta)(L_{xx}^2 + L_{yy}^2) + \sin^2 2\theta L_{xx}L_{yy} \\
&\quad + \frac{1}{2}\sin^2 2\theta(L_{xx} - L_{yy})^2 + 2L_{xy} \\
&= (\cos^4\theta + \sin^4\theta)(L_{xx}^2 + L_{yy}^2) + \frac{1}{2}\sin^2 2\theta(L_{xx}^2 + L_{yy}^2) + 2L_{xy} \\
&= (\cos^2\theta + \sin^2\theta)^2(L_{xx}^2 + L_{yy}^2) + 2L_{xy} \\
&= L_{xx}^2 + L_{yy}^2 + 2L_{xy}
\end{aligned}$$

and thus

$$\widetilde{C} = \frac{1}{2}\sqrt{\widetilde{L}_{xx}^2 + 2\widetilde{L}_{xy}^2 + \widetilde{L}_{yy}^2} = C$$

B.2 Illumination

Under the assumption of an affine illumination model $\widetilde{I} = aI + b$ where I is the underlying image, the four transformed measures $\widetilde{\Theta}$, $\widetilde{M}$, $\widetilde{S}$ and $\widetilde{C}$ are computed:

$$
\begin{aligned}
\widetilde{L}_{x^m y^n} &= aL_{x^m y^n}, \quad m+n>0 \\
\widetilde{\Theta} &= \tan^{-1}\left(\frac{aL_y}{aL_x}\right) = \tan^{-1}\left(\frac{L_y}{L_x}\right) = \Theta \\
\widetilde{M} &= \sqrt{(aL_x)^2 + (aL_y)^2} = a\sqrt{L_x^2 + L_y^2} = aM \\
\widetilde{S} &= \frac{2}{\pi}\tan^{-1}\left(\frac{-aL_{xx} - aL_{yy}}{\sqrt{4(aL_{xy})^2 + (aL_{xx} - aL_{yy})^2}}\right) \\
&= \frac{2}{\pi}\tan^{-1}\left(\frac{a(-L_{xx} - L_{yy})}{\sqrt{a^2\left(4L_{xy}^2 + (L_{xx} - L_{yy})^2\right)}}\right) \\
&= \frac{2}{\pi}\tan^{-1}\left(\frac{-L_{xx} - L_{yy}}{\sqrt{4L_{xy}^2 + (L_{xx} - L_{yy})^2}}\right) = S \\
\widetilde{C} &= \frac{1}{2}\sqrt{(aL_{xx})^2 + 2(aL_{xy})^2 + (aL_{yy})^2} \\
&= \frac{1}{2}a\sqrt{L_{xx}^2 + 2L_{xy}^2 + L_{yy}^2} = aC
\end{aligned}
$$

Appendix C

Grid layouts

This appendix chapter describes the construction of our log-polar grid layouts shown in Figure 4.2. They consist of a central cell as well as a constant number of Gaussian cells in each of a number of rings, where the cells' standard deviations (referred to as radii) are tangent to each other and thus non-overlapping. We define r_i to be the radius of a cell in ring i, and d_i to be its distance from the origin.

C.1 Log-polar grids

We consider two tangent cells from neighbouring rings in the log-polar grid, which will be sufficient to construct the whole grid:

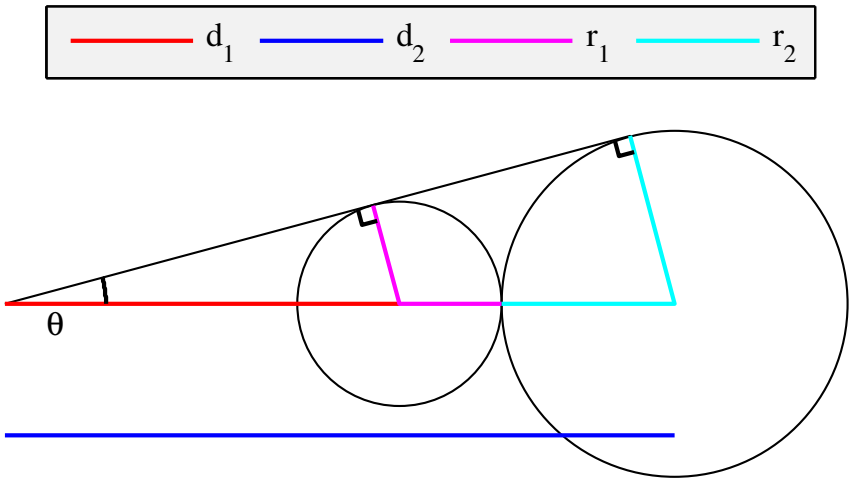


Figure C.1: Log-polar grid layout.

First we compute the relation between r_i and d_i. As we have a right-angled triangle, we can use the sine definition:

$$\sin\theta = \frac{r_i}{d_i} \tag{C.1}$$

Note that this linear dependency holds for any ring i. Thus, with the grid definition, the only thing left is to compute the relation between two neighbouring rings, which we can write as the constant k such that $r_{i+1} = kr_i$ and $d_{i+1} = kd_i$. We compute k:

$$\begin{aligned} d_2 = kd_1 = d_1 + r_1 + r_2 = d_1 + d_1\sin\theta + kd_1\sin\theta \quad &\Rightarrow \\ k - k\sin\theta = 1 + \sin\theta \quad &\Rightarrow \\ k = \frac{1+\sin\theta}{1-\sin\theta} \end{aligned}$$

Now the whole log-polar grid can be constructed. A circle is added to the middle as the central cell, and the whole grid is rescaled according to the desired radius.

C.2 Concentric log-polar grids

We again consider two tangent cells from neighbouring rings, this time offset by the angle θ, causing a different layout:

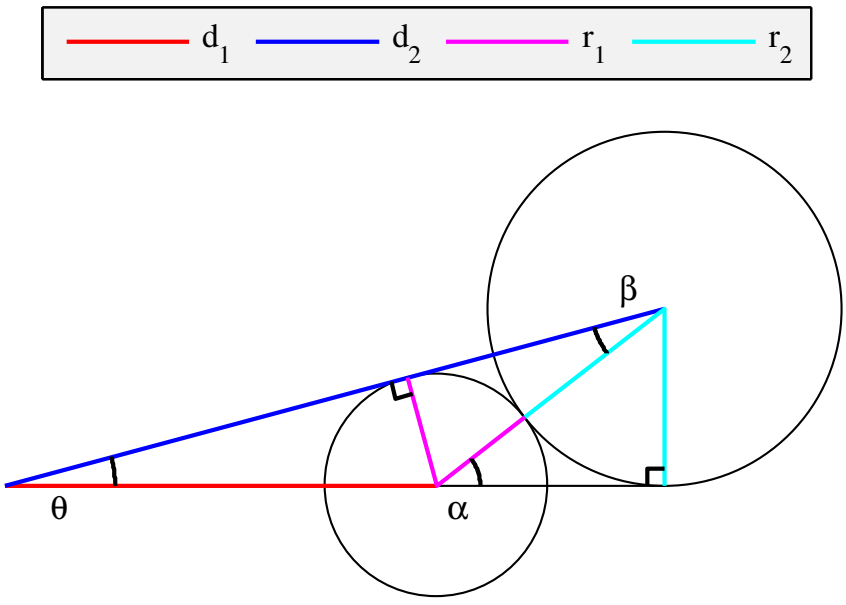


Figure C.2: Concentric log-polar grid layout.

Note that Equation (C.1) still holds for this grid. The constant k however is more complicated to compute. First note that the supplementary angle to α together with θ and β must add up to 180°, so $\alpha = \theta + \beta$. We use the sine definition with two of the right-angled triangles:

$$\begin{aligned} \sin(\theta + \beta) &= \frac{r_2}{r_1 + r_2} \\ \sin\beta &= \frac{r_1}{r_1 + r_2} \end{aligned} \tag{C.2}$$

Added together this gives

$$\sin(\theta + \beta) + \sin\beta = \frac{r_1 + r_2}{r_1 + r_2} = 1$$

which is sufficient to calculate β by symbolic computation, but the result is quite ugly. Then k can be computed from Equation (C.2):

$$\begin{aligned} \sin\beta &= \frac{1}{1+k} \quad \Rightarrow \\ k &= \frac{1}{\sin\beta} - 1 \end{aligned}$$

Now the whole concentric log-polar grid can be constructed alike the ordinary log-polar grid.

Appendix D

Gaussian derivatives

The n'th order derivative of the 1D Gaussian kernel G has the following relation to the Hermite polynomial:

$$G_{x^n}(x;\sigma) = \left(\frac{1}{\sqrt{2}\sigma}\right)^n \cdot \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{x^2}{2\sigma^2}\right) \cdot \text{hermite}\left(n, \frac{x^2}{\sqrt{2}\sigma}\right)$$

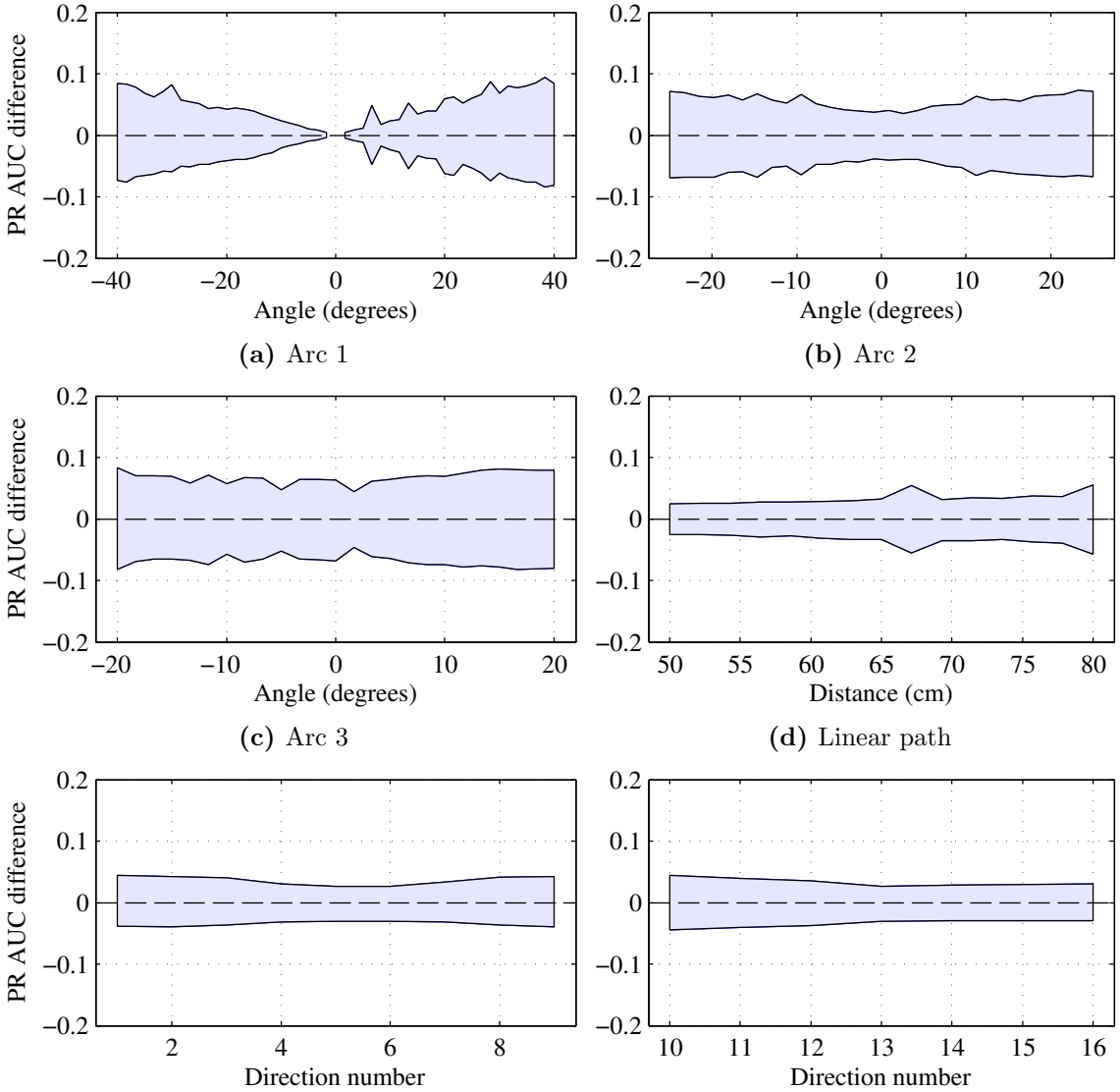
Thus an arbitrary 2D Gaussian derivative with equal variance in both dimensions can be written as:

$$G_{x^m y^n}(x, y; \sigma) = G_{y^n}(y;\sigma)^{\mathrm{T}} * G_{x^m}(x;\sigma)$$

Appendix E

Confidence intervals for image correspondence

In this appendix chapter we show the 95% confidence interval graphs for the difference between mean performance on the DTU dataset paths of two descriptors. Figures E.1 to E.5 shown the graphs for the combinations of the GO, SI, GO+SI and SIFT descriptors. The combination GO+SI and SIFT is left out since it is shown directly in the report in Figure 5.18.

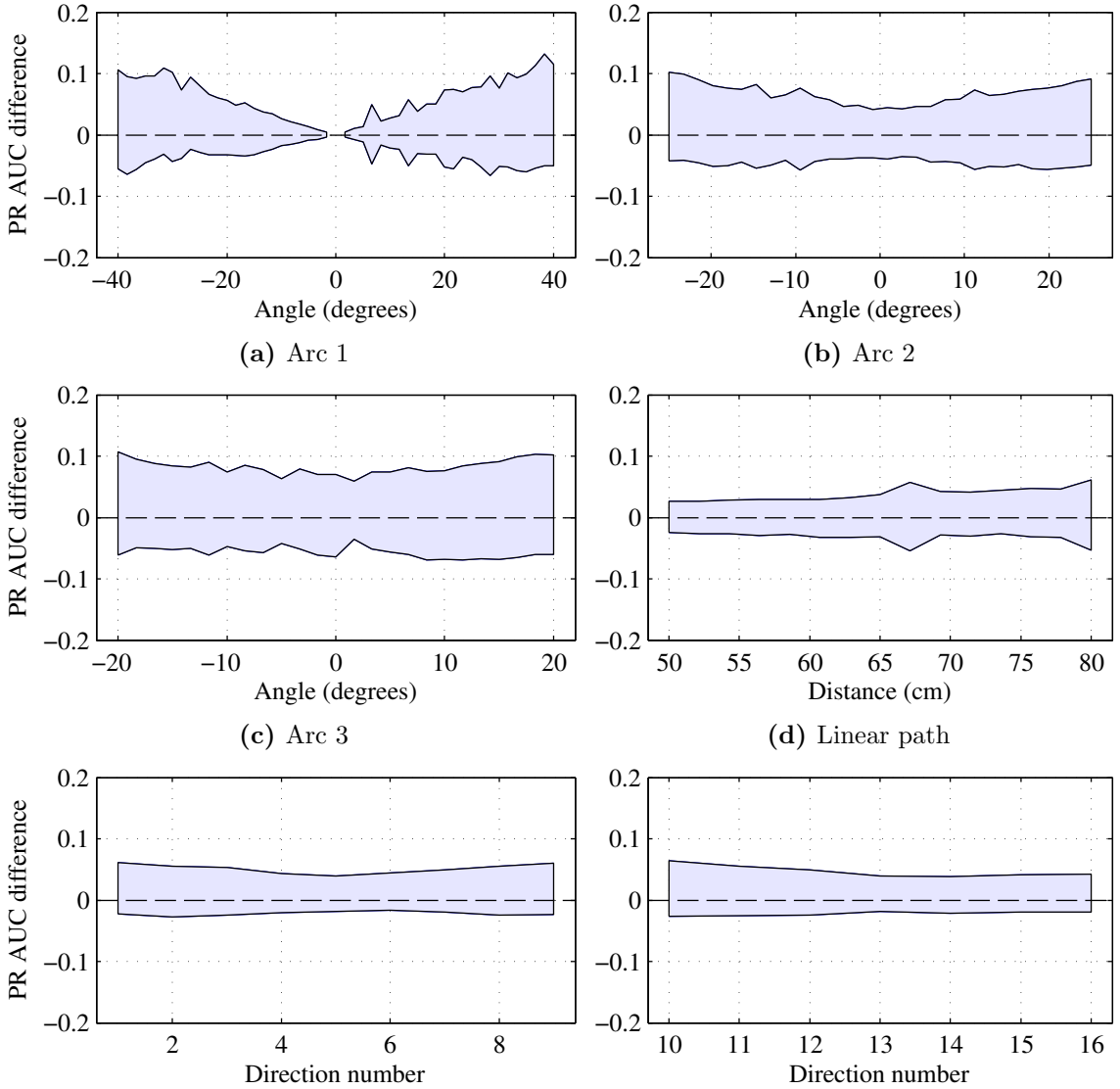


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure E.1: 95% confidence intervals on the PR AUC difference between our GO+SI and GO descriptors. A positive difference denotes that GO+SI outperforms GO and vise versa. If the confidence interval for a position contains zero, there is no significant difference in performance.

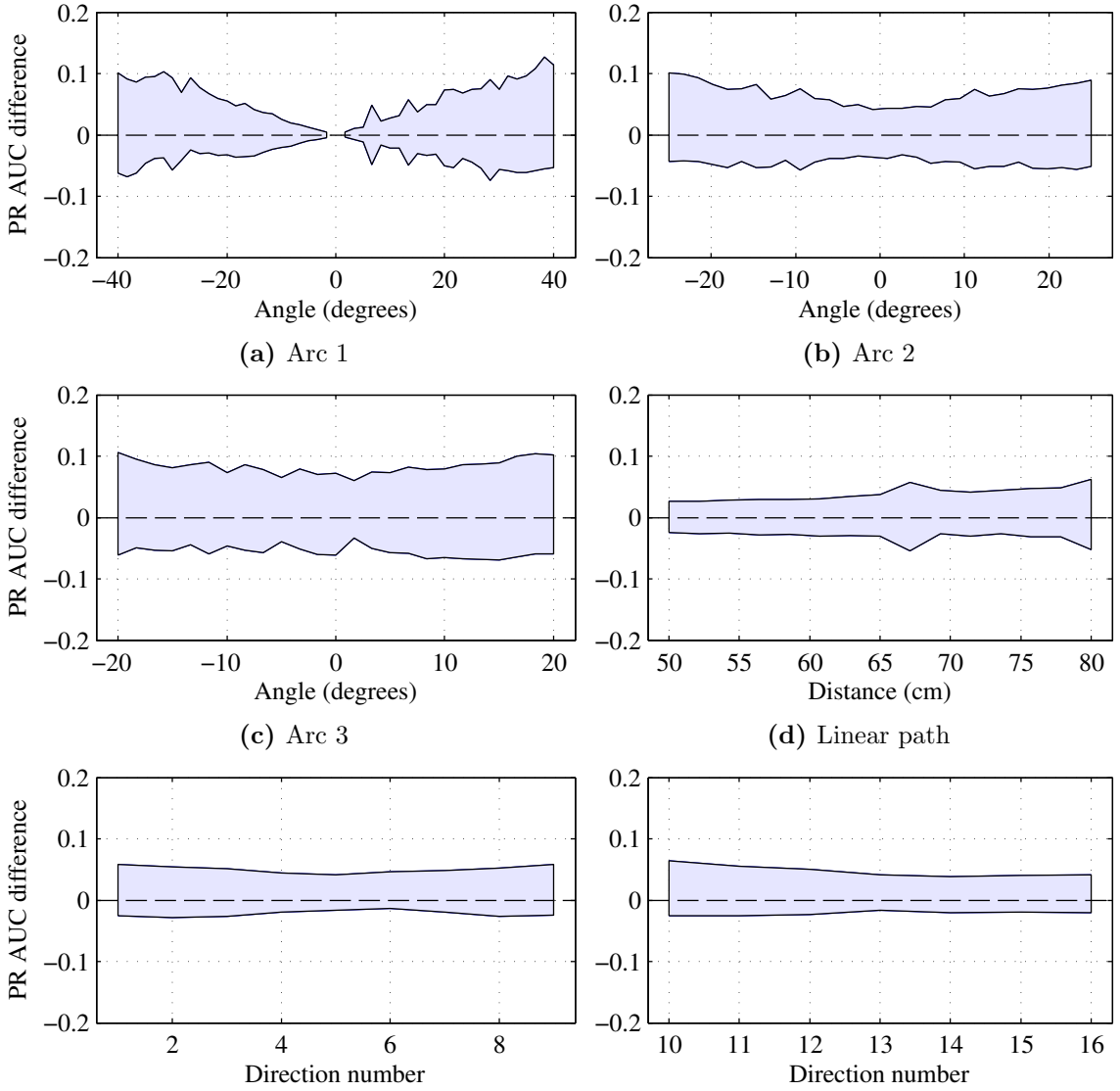


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure E.2: 95% confidence intervals on the PR AUC difference between our GO+SI and SI descriptors. A positive difference denotes that GO+SI outperforms SI and vise versa. If the confidence interval for a position contains zero, there is no significant difference in performance.

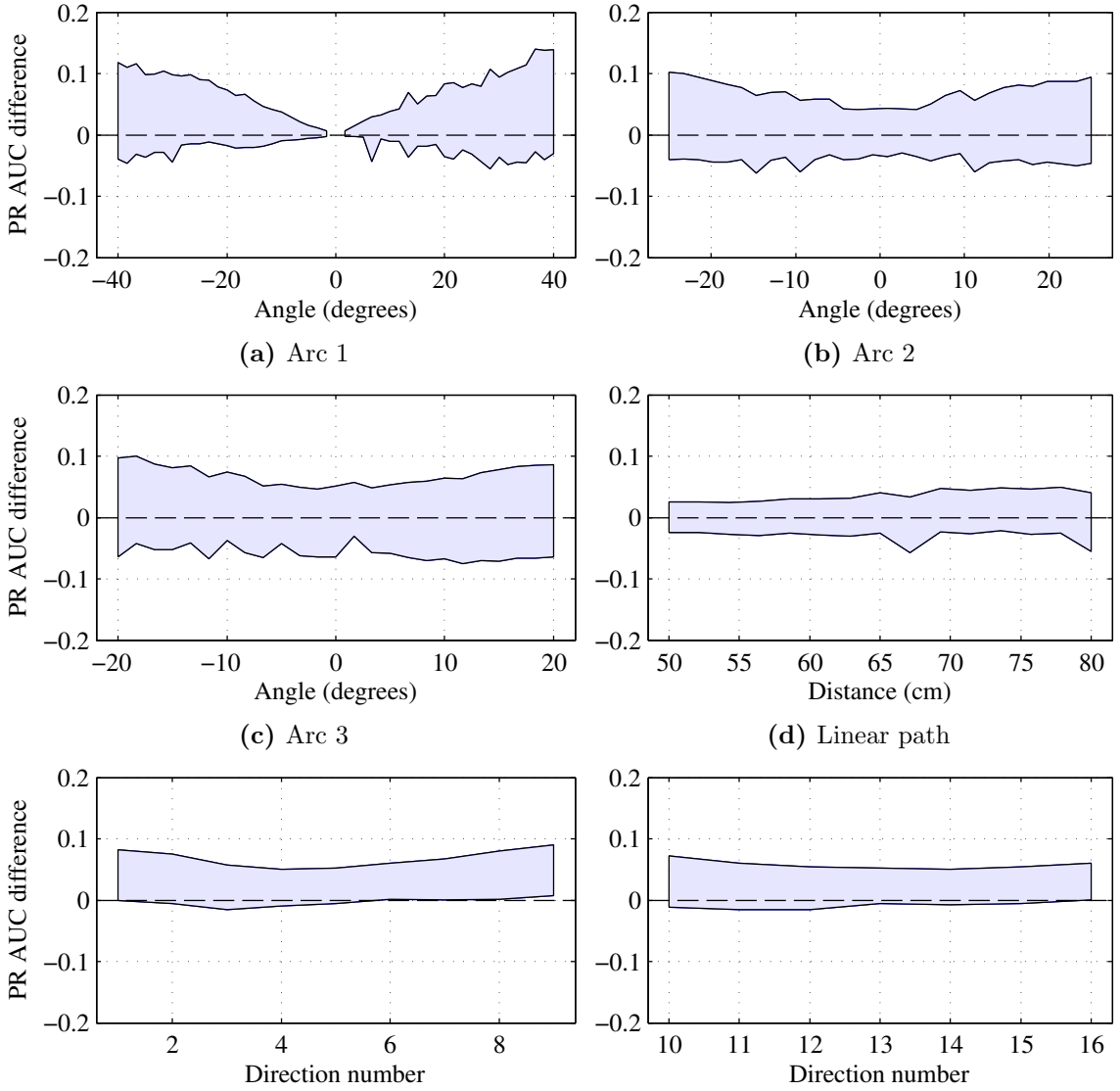


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure E.3: 95% confidence intervals on the PR AUC difference between our GO and SI descriptors. A positive difference denotes that GO outperforms SI and vice versa. If the confidence interval for a position contains zero, there is no significant difference in performance.

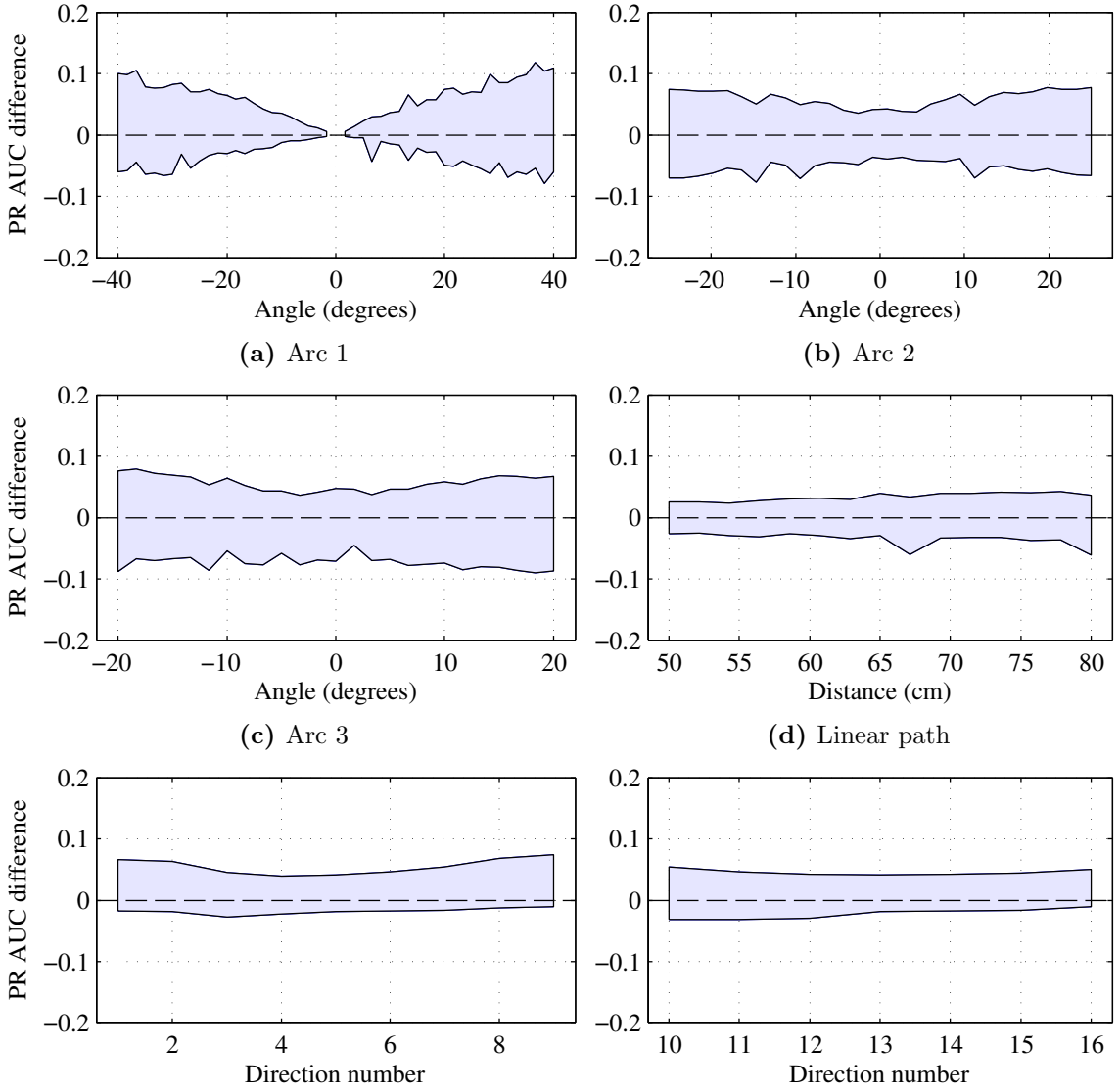


(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure E.4: 95% confidence intervals on the PR AUC difference between our GO descriptor and SIFT. A positive difference denotes that GO outperforms SIFT and vise versa. If the confidence interval for a position contains zero, there is no significant difference in performance.



(a) Arc 1 **(b)** Arc 2

(c) Arc 3 **(d)** Linear path

(e) Light path, x-axis **(f)** Light path, z-axis

Figure E.5: 95% confidence intervals on the PR AUC difference between our SI descriptor and SIFT. A positive difference denotes that SI outperforms SIFT and vise versa. If the confidence interval for a position contains zero, there is no significant difference in performance.

Bibliography

[ADP10a] H. Aanæs, A. L. Dahl, and V. Perfanov. *A Ground Truth Data Set for Two View Image Matching*. Tech. rep. DTU Informatics, Technical University of Denmark, 2010. URL: `http://roboimagedata.imm.dtu.dk/papers/technicalReport.pdf`.

[ADP10b] H. Aanæs, A.L. Dahl, and K.S. Pedersen. "On recall rate of interest point detectors". In: *3DPVT 2010: Fifth International Symposium on 3D Data Processing, Visualization and Transmission*. 2010.

[ADP12] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. "Interesting interest points". In: *International Journal of Computer Vision* 97.1 (2012), pp. 18–35.

[CG10] Michael Crosier and Lewis D Griffin. "Using basic image features for texture classification". In: *International Journal of Computer Vision* 88.3 (2010), pp. 447–460.

[Csu+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. "Visual categorization with bags of keypoints". In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. 2004, pp. 1–2.

[Cui+09] Yan Cui, Nils Hasler, Thorsten Thormählen, and Hans-Peter Seidel. "Scale Invariant Feature Transform with Irregular Orientation Histogram Binning". In: *Image Analysis and Recognition*. Ed. by Mohamed Kamel and Aurélio Campilho. Vol. 5627. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 258–267. ISBN: 978-3-642-02610-2. DOI: `10.1007/978-3-642-02611-9_26`. URL: `http://dx.doi.org/10.1007/978-3-642-02611-9_26`.

[DAP11] Anders Lindbjerg Dahl, Henrik Aanæs, and Kim Steenstrup Pedersen. "Finding the best feature detector-descriptor combination". In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*. IEEE. 2011, pp. 318–325.

[DG06] Jesse Davis and Mark Goadrich. "The Relationship Between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania: ACM, 2006, pp. 233–240. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143874. URL: http://doi.acm.org/10.1145/1143844.1143874.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[Fan+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. "LIBLINEAR: A Library for Large Linear Classification". In: *J. Mach. Learn. Res.* 9 (June 2008), pp. 1871–1874. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1390681.1442794.

[Fel+10] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part-Based Models". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (Sept. 2010), pp. 1627–1645. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.167.

[FMR08] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[Gri07] L. D. Griffin. "The Second Order Local-Image-Structure Solid". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* 29.8 (Aug. 2007), pp. 1355–1366.

[Gri97] L. D. Griffin. "Scale-imprecision space". In: *Image and Vision Computing* 15 (1997), pp. 369–398.

[GW08] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2008. ISBN: 013505267X.

[HDF12] J. Heinly, E. Dunn, and J.-M. Frahm. "Comparative Evaluation of Binary Features". In: *ECCV'12*. Springer LNCS 7573. 2012, pp. 759–773.

[KD92] Jan J Koenderink and Andrea J van Doorn. "Surface shape and curvature scales". In: *Image and vision computing* 10.8 (1992), pp. 557–564.

[KS04] Yan Ke and R. Sukthankar. "PCA-SIFT: a more distinctive representation for local image descriptors". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on.* Vol. 2. June 2004, pages. DOI: 10.1109/CVPR.2004.1315206.

[Kv99] J. J. Koenderink and A. J. van Doorn. "The Structure of Locally Orderless Images". In: *International Journal of Computer Vision* 31.2/3 (1999), pp. 159–168.

[Lar+12] Anders Boesen Lindbo Larsen, Sune Darkner, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. "Jet-based local image descriptors". In: *Computer Vision–ECCV 2012.* Springer, 2012, pp. 638–650.

[Lar12] Anders Boesen Lindbo Larsen. "An in-depth study of local image descriptors and their performance". MA thesis. Computer Science, University of Copenhagen, Feb. 2012.

[LG09] Martin Lillholm and Lewis D. Griffin. "Statistics and category systems for the shape index descriptor of local 2nd order natural image structure". In: *Image and Vision Computing* 27.6 (2009), pp. 771–781. ISSN: 0262-8856. DOI: http://dx.doi.org/10.1016/j.imavis.2008.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0262885608001753.

[Low04] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. "A performance evaluation of local descriptors". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.10 (2005), pp. 1615–1630.

[Par62] Emanuel Parzen. "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* (1962), pp. 1065–1076.

[Ped+13] Kim Steenstrup Pedersen, Kristoffer Stensbo-Smidt, Andrew Zirm, and Christian Igel. "Shape Index Descriptors Applied to Texture-Based Galaxy Analysis". In: *Computer Vision (ICCV), Conference on.* IEEE. 2013, pp. 2440–2447.

[PP00] Constantine Papageorgiou and Tomaso Poggio. "A Trainable System for Object Detection". In: *Int. J. Comput. Vision* 38.1 (June 2000), pp. 15–33. ISSN: 0920-5691. DOI: 10.1023/A:1008162616689. URL: http://dx.doi.org/10.1023/A:1008162616689.

[Sán+13] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. “Image classification with the Fisher vector: Theory and practice”. In: *International Journal of Computer Vision* 105.3 (Dec. 2013), pp. 222–245. DOI: 10.1007/s11263-013-0636-x.

[TLF08] Engin Tola, V. Lepetit, and P. Fua. “A fast local descriptor for dense matching”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. June 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587673.

[TW09] M. Toews and III Wells W. “SIFT-Rank: Ordinal description for invariant feature correspondence”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. June 2009, pp. 172–177. DOI: 10.1109/CVPR.2009.5206849.

[VF08] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. http://www.vlfeat.org/. 2008.

[VGS10] K. E A Van de Sande, T. Gevers, and C. G M Snoek. “Evaluating Color Descriptors for Object and Scene Recognition”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (Sept. 2010), pp. 1582–1596. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.154.

[VH00] Bram Van Ginneken and Bart M ter Haar Romeny. “Applications of Locally Orderless Images”. In: *Journal of Visual Communication and Image Representation* 11 (2000), pp. 196–208.

[WHB09] Simon Winder, Gang Hua, and Matthew Brown. “Picking the best daisy”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 178–185.