UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Low level methods for complex image analysis — edge and line segment detection and texture discrimination

Rui Filipe Cardoso Guerreiro

Supervisor: Doctor Pedro Manuel Quintas Aguiar

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Merit

2015

UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Low level methods for complex image analysis — edge and line segment detection and texture discrimination

Rui Filipe Cardoso Guerreiro

Supervisor: Doctor Pedro Manuel Quintas Aguiar

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Merit

Funding institutions
Fundação para a Ciência e a Tecnologia

2015

Resumo

Detecção de bordas, extracção de segmentos de recta e discriminação de múltiplas texturas são métodos de baixo nível usados frequentemente em análise de imagem. Contudo, os métodos actuais não são robustos ou eficientes a lidar com as imagens complexas de cenários realistas. Detectores de borda não conseguem simultaneamente lidar com ruído ou detalhes e ter boa localização. Esquemas de votação global baseados na Transformada de Hough são amplamente usados para extrair linhas, mas não lidam bem com imagens detalhadas. Em oposição, os chamados métodos locais carecem de robustez para extrair segmentos de recta em situações realistas desafiantes, por exemplo, quando os segmentos cruzam-se. Os métodos de discriminação de texturas são complexos ou pouco discriminantes. Nesta tese, abordamos as limitações críticas desses métodos. Detecção de bordas e extracção de segmentos de recta combinam informação contextual, que lida com o ruído e detalhes, com informação local, que tem boa localização, tendo em conta a conectividade. Os métodos resultam computacionalmente viáveis e precisos, com resultados sem intervalos. Discriminação de texturas usa aprendizagem supervisionada para aprender a extrair dados esparsos mas altamente discriminantes de texturas, resultando num método simples e preciso. Ilustramos com imagens sintéticas e reais o bom desempenho dos métodos propostos.

Palavras-chave: Processamento de imagem, detecção de bordas, extracção de segmentos de recta, discriminação de múltiplas texturas, conectividade, imagens complexas, testes duas-amostras, Transformada de Hough, momentos estatísticos, Algoritmos Genéticos

Abstract

Edge detection, line segment extraction and multiple texture discrimination are low-level methods that are widely used in image understanding. However, current methods are not robust or efficient in dealing with the complex images of realistic scenarios. Edge detectors cannot simultaneously cope with noise or image clutter and have good localization. Global voting schemes based on the Hough Transform have been widely used to extract lines but these methods do not deal well with cluttered images. In opposition, the so-called local methods lack robustness to extract line segments in challenging realistic situations, *e.g.*, when segments cross. Multiple texture discrimination methods are either very complex or not very discriminant. In this thesis, we address the critical limitations of these methods. Edge detection and line segment extraction methods combine contextual information, which can deal with noise and clutter, with local information, which has good localization, taking connectivity into account. The methods result computationally feasible and accurate, with gap-free results. Multiple texture discrimination uses supervised learning to learn how to extract sparse but highly discriminant data from textures, resulting in a simple and accurate method. We present experiments that illustrate, with synthetic and real images, the good performance of the proposed methods.

Keywords: Image processing, edge detection, line segment extraction, multiple texture discrimination, connectivity, complex images, two-sample tests, Hough Transform, statistical moments, Genetic Algorithms

Acknowledgments

I would like to express my special appreciation and thanks to my supervisor Professor Doctor Pedro Manuel Quintas Aguiar, who has been a great mentor to me. He has taught me to have a critical view about ideas, their value and, finally, on how to express them clearly. I believe this allowed me to focus my inquisitive spirit and made me grow as a researcher. I would also like to thank my committee members, Professor Doctor Mário Alexandre Figueiredo and Professor Doctor José Bioucas Dias for serving as my committee members and for their valuable comments and suggestions. I would also like to thank other professors at Instituto Superior Técnico and colleagues, for introducing me to new techniques, suggestions and friendly support.

I would like to thank Portuguese science institution Fundação para a Ciência e a Tecnologia (FCT) for partially supporting this work under Project PEst-OE/EEI/LA0009/2011, and Grants MODI-PTDC/EEA-ACR/72201/2006 and SFRH/BD/48602/2008.

Finally, a special thank you to my family. Words can not express how grateful I am for their sacrifices in my behalf and all their support. I would also like to thank many other people that have provided invaluable support over the years including Sónia Gomes, Doctor Spiros Kitsinelis, Sara Noll, Marco Pereira and Helen Yong.

Contents

Chapter 1
Introduction

1.1 Motivation

Despite powerful advances in computer vision such as deep learning [Bengio 2009], many real-world applications in areas such as automotive (*e.g.*, autonomous vehicles [Urmson 2007]) and surveillance [Thida 2013] rely on low-level methods such as, *e.g.*, edge detection, line segment extraction, and multiple texture discrimination. These applications require robust methods that deal with complex images.

The information contained in a complex image such as the one in Fig. 1.1 is so large and diverse that it is useful to describe it in a way that is: concise, drastically reducing the amount of data to be processed; explicit, preserving useful structural information about object boundaries; and reliable. This information is then used by image understanding methods to interpret the image. This may include figuring out which objects are in the image, their spatial relationships, what they are doing, etc. It may ultimately include making some decision for further action.

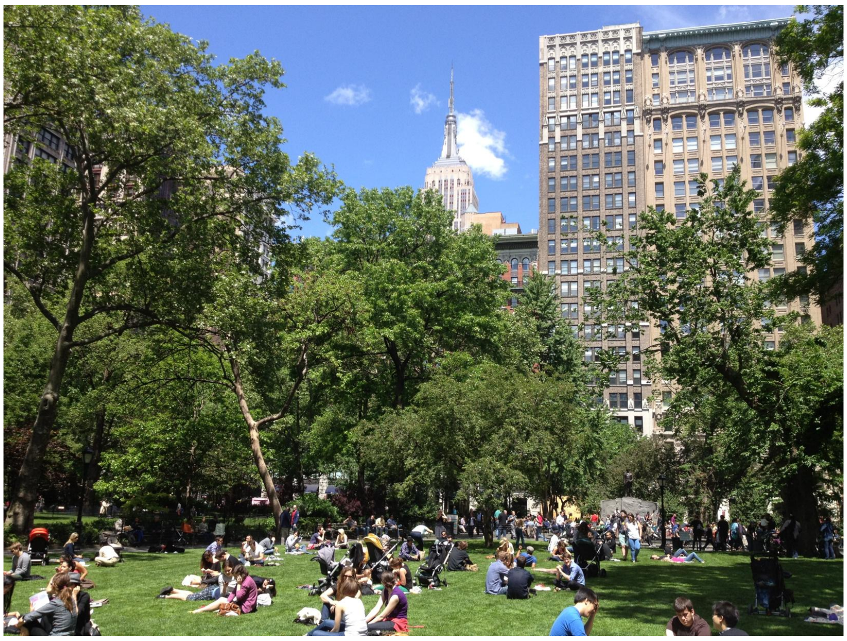


Figure 1.1: Complex image containing edges, line segments and textures.

To obtain a concise, explicit and reliable description of a complex image, image analysis methods extract low-level features such as edges, contours, line segments and textures. Experiments with the human visual system show that object boundaries are extremely important and often sufficient for object recognition [Attneave 1954]. It has been shown [Lindeberg 1996] that, under rather general assumptions about the image formation process, discontinuities in image brightness can be assumed to correspond to a discontinuity in either depth, surface orientation, reflectance or illumination. Given the key role of these low-level features, it is important that they are extracted accurately.

This thesis introduces novel low-level image analysis methods for edge detection, line segment extraction and multi-texture discrimination. Although such methods have been the focus of many researchers roughly since the start of the image processing discipline, it will be shown that the current state-of-the-art methods are not suitable for dealing with the complex images taken from unconstrained real life scenarios and that new solutions are needed. In particular:

• Edge detection — The Canny edge detector [Canny 1986] is a very popular method for edge detection but, by not having a suitable strategy for dealing with noise or clutter (*i.e.*, densely concentrated details on the image), it can miss the detection of many real edges and return false edges due to noise. This is illustrated in Fig. 1.2, where many edges of the challenging image are not captured.

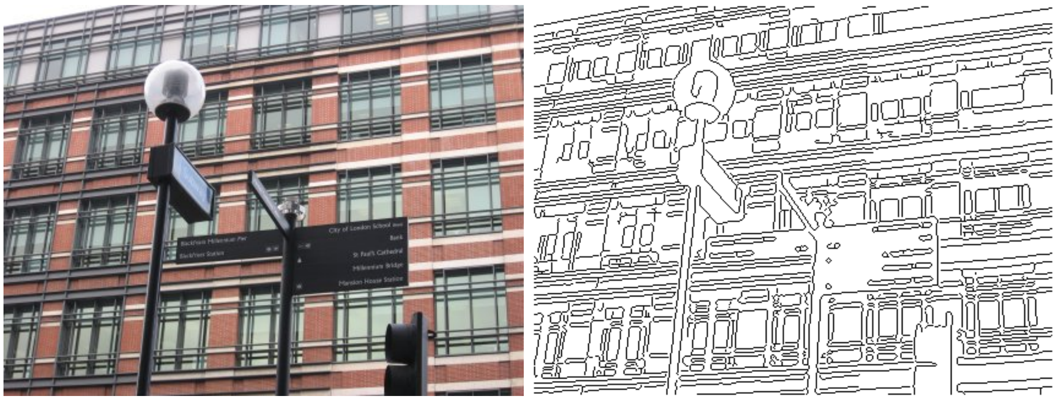


Figure 1.2: Complex image with dense clutter and the poor response of the Canny edge detector. Left: image; right: the detected edges (note the many missing edges)

• Line segment extraction — The Hough transform (HT) [Hough 1962, Duda 1972] is arguably the most popular method to detect lines in images. Its success comes from its global nature, since all points in a line contribute to its detection, which makes it a statistically robust estimator [Goldenshluger 2004]. However, when the image has many edge points, as is often the case for images taken from unconstrained real life scenarios, the HT no longer detects lines reliably. On the other hand, local methods for line segment extraction such as the popular Line Segment Detector (LSD) [von Gioi 2010] lack robustness to deal with challenging situations such as, *e.g.*, crossing line segments, originating many interrupted segments. The limitations of the HT and LSD methods, both considered state-of-the-art, in dealing with complex images are illustrated in Fig. 1.3.

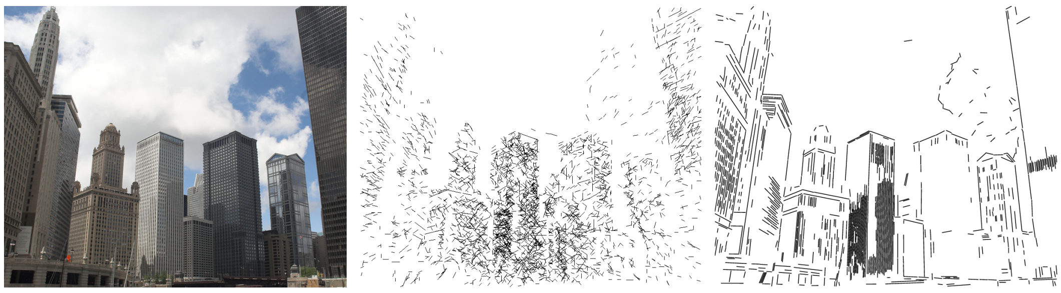


Figure 1.3: Limitation of the HT and LSD in detecting line segments in a complex image. Left to right: image and line segments detected by the HT and the LSD methods, respectively

• Multi-texture discrimination — The popular Gabor Filter Banks approaches [Jain 1991, Malik 2001] have high classification rates but, in practice, use filter banks made up of about ten to fifty relatively large two-dimensional filters, computed at every pixel. Another popular method is the Gray-Level Co-occurrence Matrices [Haralick 1973], which, according to the comparative study in [Partio 2007], has a low discrimination rate of about 66.7%. Local Binary Patterns [Ojala 2002] has high discrimination rate (about 90.7% [Partio 2007]) but is not simple to compute. This illustrates that current methods for multi-texture discrimination are either not very discriminant or are computationally too complex for several real-life applications.

1.2 Contributions

The main focus of this thesis is the development and testing of novel low-level image analysis methods for edge detection, line segment extraction and multi-texture discrimination of complex images. For each method, we review the related work, in general and in the context of complex images. The methods are then presented and, for each one, an experimental section with synthetic and real images shows their good performance in dealing with complex images. In detail:

• Edge detection — We propose two statistical edge detectors using elongated oriented footprints. These edge detectors combine what we denote as *contextual edges*, obtained with large footprints and that deal well with noise; and *local edges*, obtained with very small footprints and with good localization, to achieve an edge detector with both qualities, as idealized by Canny [Canny 1986]. The connectivity between edge points is handled explicitly inside the edge detector, originating edge maps that are connected, as desired by the methods that follow. The methods are:

1. Non-paired Total Variation (TV) parametric test — it computes the parametric parameters, sample mean and variance, of (assumed underlying Normal) pixel intensity values along oriented elongated footprints using a computationally simple running average approach. Then, the TV distance between the two distributions is computed, to test the existence of a contextual edge. The contextual edge transition type (*i.e.*, if the edge is a light-to-dark or dark-to-light transition) is used to select and mark local edges of the same type and that are connected, *i.e.*, close to each other;

2. Paired nonparametric sign test — it computes image derivatives and their sign, *i.e*, determines local light-to-dark and dark-to-light transition. Then, by adding the transition signs along elongated and oriented footprints, effectively counting the number of positive minus the number of negative transitions around each local edge pixel, it determines if there is a predominance of a positive or negative transition, as is expected around an edge. This is known as the sign test [Sheskin 2007]. Contextual and local edges are then combined using connectivity, as above.

• Line segment extraction — We propose two line segment extraction methods:

1. Connectivity-enforcing Hough Transform — Presented in [Guerreiro 2012, Guerreiro 2011], it incorporates connectivity into the Hough Transform (HT) voting process. Connectivity is enforced in the HT by imposing that edge points only vote for lines in which they are spatially connected to other points. As a consequence, the vast majority of spurious votes are eliminated and peaks in the accumulator array become prominent and truly correspondent to line segments of maximum length. Simultaneously, the method integrates into the voting process the usually separate step of determining the extremes of the line segments. Illustrative results of experiments that use synthetic and real images are presented to compare it with the standard HT [Duda 1972] and the state-of-the-art local method LSD [von Gioi 2010];

2. Combining contextual and local edges for line segment extraction in cluttered images — Presented in [Guerreiro 2013a], it uses the novel TV parametric test edge detector to obtain edge points that are connected, despite noise and/or clutter. This enables the use of a computationally simple region growing method to obtain connected areas of all lengths and widths, as in typical local methods. Each connected area is then fit into a rectangle and tested for straightness. Illustrative results using synthetic and real images to compare our method with other methods: the standard HT [Duda 1972], the state-of-the-art of local methods LSD [von Gioi 2010], and our HT-based method above.

• Multi-texture discrimination — Presented in [Guerreiro 2013b, Guerreiro 2010], we propose *optimal FIR filters* made up of a filter bank with adjustable coefficients obtained through supervised learning. Rather than the two-dimensional large footprint filters of typical filter banks, we use: a) one-dimensional filters, applied horizontally and vertically, to perform orientation-dependent discrimination; or b) ring-shaped filters, to perform rotationally-invariant discrimination of textures. These filters are simple to compute and we show that they suffice to extract the relevant textural features. In the local energy function, we compute the first four moments of each filter output and weight their contribution to the overall classification. This results computationally very simple and we show that it approximates typical local energy functions, including the computationally complex and exceptionally discriminant clustering and histogram matching approach of [Malik 2001]. We use a Maximum-Likelihood classifier with normalized Euclidean distance to obtain the class of a texture sample. To learn the filter bank coefficients, we define an objective function to minimize, which, due to the use of moments, results non-convex. Supervised learning is then performed by using a Genetic Algorithm, whose properties enable an apt solution. We conduct an experimental analysis of our method using the publicly available Brodatz [Brodatz 1966] and VisTex [vis 2002] albums. We conclude that our method outperforms state-of-the-art methods, such as Gabor Filter Banks, and Local Binary Patterns, both in terms of accuracy and computational simplicity.

1.3 Organization

The organization of the remaining of this thesis is as follows. Section 2 discusses the relevant prior art and section 3 introduces the two novel statistical edge detectors using elongated filters. The novel line segment extractors are introduced in section 4 and section 5. The novel multi-texture discrimination method is detailed in section 6. Section 7 provides a brief illustration that the combination of low level methods enable better image understanding of complex images. Finally, section 8 concludes the thesis and discusses ideas that can be part of future work.

Chapter 2 Related work

2.1 Edge detection in complex images

2.1.1 Motivation

A typical way to represent a complex image is by extracting their edges. It significantly reduces the amount of data to be processed, filtering out information that may be regarded as less relevant, while preserving the important structural properties of an image. This is supported by experimental studies with the human visual system, that show that object boundaries are extremely important and often sufficient for object recognition [Attneave 1954] and that, under rather general assumptions about the image formation process, discontinuities in image brightness can be assumed to correspond to a discontinuity in either depth, surface orientation, reflectance or illumination [Lindeberg 1996]. The strong link between edges in the image domain and physical properties of the world make edge detection one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques. Despite considerable work and progress made on this subject (a review on edge detection methods can be found in [Oskoei 2010]), edge detection is still a challenging research problem due to the lack of a robust and efficient general purpose algorithm, in particular for noisy or cluttered images.

According to Canny [Canny 1986], an edge detector should have the following three criteria: 1. good detection: low probability of failing to mark real edge points; 2. good localization: points marked as edges should be as close as possible to the center of true edges; 3. single response: only one response to a single edge point. Although several researchers in the past decades have tried to design of an edge detector that regards some or all of these properties in various scenarios, current solutions cannot deal with large amounts of noise or image clutter. Noisy images with low signal to noise ratio are common in a variety of domains in which pictures are captured under limited visibility. Examples include electron microscopy (EM) images taken under certain protocols (*e.g.*, cryo-EM), fingerprint images with low tissue contrast, photos acquired under poor lighting, etc.

Existing edge detection methods use various strategies to overcome noise. Methods that use isotropic smoothing (*e.g.*, [Canny 1986]) are limited, since aggressive smoothing tends to smear the edges. Anisotropic diffusion methods [Perona 1990] too face difficulties dealing with low signal-to-noise, as they are typically initialized by local image gradients, whose estimation in noisy images may be unreliable. Recent methods use a variety of filter banks of various lengths and orientations and in various resolutions to compute *significantly different* oriented means in an image and improve the detection of faint edges, *e.g.*, rectangular filters [Galun 2007], curvelets [Gebäck 2009], shearlets [Yi 2009], and beamlets [Xiaoming 2007]. Although these filters enable a better recovery of weak edges than standard edge detectors based on Sobel or Prewitt operators, they have typically poor localization; use multiple resolutions, thus eliminating smaller details; and most only discriminate opposing sets of pixels based on their mean, neglecting their median and/or standard deviations, which we show is quite useful in this scenario. Other methods compare histograms of intensities and textures in two adjacent half-disks [Martin 2004]. Finally, [Bovik 1986, Fesharaki 1994, Thune 1997, Williams 2006] considers the statistical problem of detecting the presence of edges but use small footprints. For this reason, which we detail in the sequel, the robust detection of edges in noisy and/or cluttered scenarios remains an open frontier.

2.1.2 Edge detection

To detect sharp changes in image brightness, edge detection methods typically work by computing the first derivatives of the image and then marking the pixels corresponding to local magnitude maxima as edges (alternatively, the second derivatives of the image are computed and the pixels where the resulting values cross zero are marked as edges). A one-dimensional derivative in the continuous space,

$$\frac{\partial f(x)}{\partial x} = \lim_{a_1 \to 0, a_2 \to 0} \frac{f(x+a_1) - f(x-a_2)}{a_1 + a_2},$$

can be approximated in the discrete scenario by

$$\frac{\partial f(x)}{\partial x} = \frac{1}{2}[f(x+1) - f(x-1)],$$

or, equivalently, by convolving the image with *central difference operator*

$$\boldsymbol{K}_{90} = \frac{1}{2}\begin{bmatrix}-1 & 0 & 1\end{bmatrix}.$$

The derivative in a two-dimensional space typically consists of a vector where each element corresponds to a one-dimensional derivative along direction θ,

$$(\boldsymbol{\nabla}_\theta \boldsymbol{I})_{(x,y)} = (\boldsymbol{I} * \boldsymbol{K}_\theta)_{(x,y)} \,. \tag{2.1}$$

The different ways to approximate ideal image derivatives lead to different results, depending on the image content. Typical derivative operators are:

- *Roberts operators*

$$\boldsymbol{K}_{45} = \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix},\ \boldsymbol{K}_{135} = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}, \tag{2.2}$$

- *Prewitt operators*

$$\boldsymbol{K}_0 = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \boldsymbol{K}_{90} = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \tag{2.3}$$

- *Sobel operators*

$$\boldsymbol{K}_0 = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \boldsymbol{K}_{90} = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \tag{2.4}$$

- *Central difference operators*

$$\boldsymbol{K}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \boldsymbol{K}_{45} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \boldsymbol{K}_{90} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{K}_{135} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}. \tag{2.5}$$

In the typical scenario in which the derivatives are computed along the horizontal and vertical directions, the magnitude and angle of the approximate derivatives are

$$\begin{aligned} \left|(\boldsymbol{\nabla I})_{(x,y)}\right| &= \sqrt{(\boldsymbol{\nabla}_0 \boldsymbol{I})^2_{(x,y)} + (\boldsymbol{\nabla}_{90} \boldsymbol{I})^2_{(x,y)}}, \\ \theta_{(x,y)} &= \arctan\left(\frac{(\boldsymbol{\nabla}_{90} \boldsymbol{I})_{(x,y)}}{(\boldsymbol{\nabla}_0 \boldsymbol{I})_{(x,y)}}\right), \end{aligned} \tag{2.6}$$

and a simple edge detection scheme can be obtained by thresholding this magnitude.

Despite having been proposed over a quarter of a century ago, the Canny edge detector [Canny 1986] is still considered a state-of-the-art method. It starts typically by applying a Prewitt or Sobel operators to an image, computing the edge magnitude (using, *e.g.*, equation (2.6) above), and thresholding the outcome. In an ideal scenario, the threshold should be large, to avoid detecting noise, but also small enough to capture fine details. To manage these often conflicting requirements, the Canny edge detector starts by using a relatively large threshold, to ensure that the detected edges are not erroneous ones. Then, by assuming that important edges form continuous contours, it attempts to find other more faint edges, *i.e.*, above a lower threshold, that are connected with the strong ones. This process can be repeated for multiple thresholds, from larger to smaller values. This processes is known as *thresholding with hysteresis*. Then, it uses an edge thinning step for suppressing edges corresponding to non-maximum derivative magnitude values [Canny 1986].

The Canny edge detector has many well-known issues. Firstly, by assuming that the faint edges within a contour must be connected with strong edges, faint contours with no strong edges are not detected. Secondly, when a strong edge is detected and when the image contains a great deal of noise or clutter, the strong edge enables the acceptance of erroneous edges connected to it, often forming complete erroneous contours. Thirdly, the edge thinning process marks a point as an edge if its amplitude is larger than that of its neighbors without checking that the differences between this point and its neighbors are higher than what is expected for random noise, making it susceptible to spurious and unstable boundaries wherever there is an insignificant change in intensity (*e.g.*, on smoothly shaded objects and on blurred boundaries) [Oskoei 2010].

2.1.3 Dealing with noise and image clutter

2.1.3.1 Low-pass filtering

Since derivatives can be seen as a high-pass filter, their output can contain a great deal of noise, as illustrated on Fig. 2.1 for a one-dimensional signal.

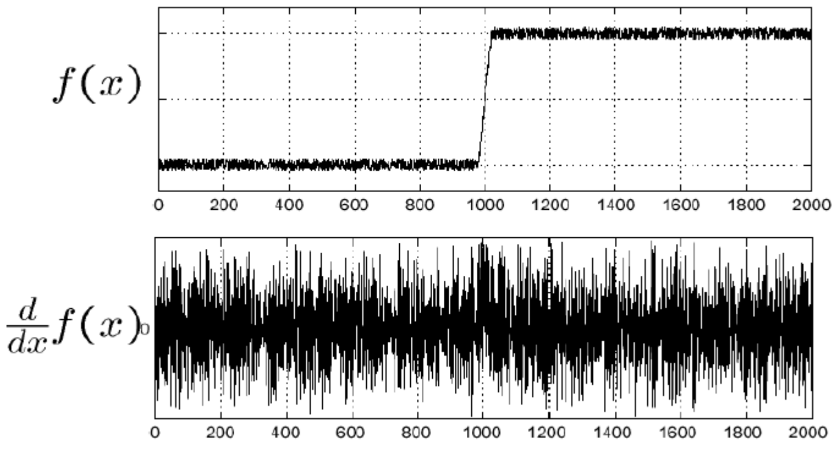


Figure 2.1: Noise harms edge detection (figure taken from [Oskoei 2010]). Top: a noisy one-dimensional signal with a smooth transition; bottom: its derivative, with an indiscernible transition

The presence of noise or clutter means that it is not always possible to obtain ideal edges from complex real life images. Therefore, edges extracted from non-trivial images are often hampered by fragmentation, meaning that the edge curves are not connected, as well as false edges not corresponding to interesting phenomena in the image — thus making the subsequent task of interpreting the image harder.

A common approach to deal with this problem is to apply a low-pass filter such as, *e.g.*, a Gaussian or box filter, to the image before applying the differential operator. This leads to the classical three stages of edge detection: noise reduction, differentiation and thresholding. An illustration of the data obtained in the classic edge detection stages is shown in Fig. 2.2.

In [Canny 1986], Canny proposed the use of an isotropic low-pass filter in the presence of noise before applying the methods described above. This technique can only be used for small amounts of noise or clutter because the aggressive isotropic smoothing step needed to deal with large amounts of noise or clutter tends to smear and, thus, destroy the edges.

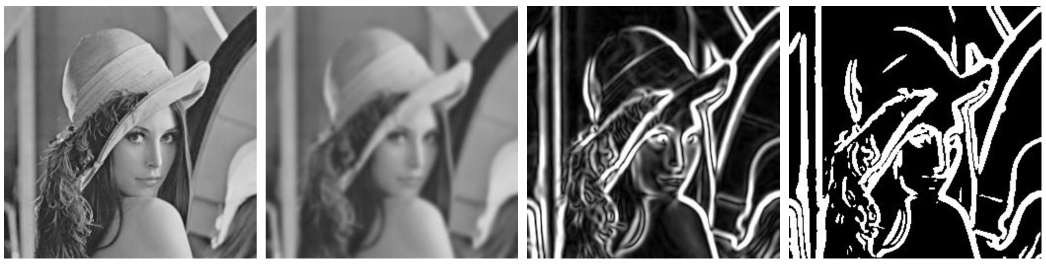


Figure 2.2: An original image (left) is blurred (second image), which is then differentiated (third image) and binarized using a fixed or variable threshold (right).

If anisotropic low-pass filtering is applied to the images, the orientation of the filter, *i.e.*, its alignment with regard to the edges on the image, is an extra factor to consider. Fig. 2.3 shows the effect of applying a long horizontal 1×15 average filter to an artificial noise image with a square on the center. Although the boundaries of the square are clearly perceptible to a human viewer on the original image, the vertical boundaries of the filtered image became blurred while the horizontal boundaries became almost noise-free (see zoomed detail of filtered image). This illustrates that low-pass filtering along the edges has the effect of preserving them while successfully dealing with noise, and that low-pass filtering across edges has a destructive effect.

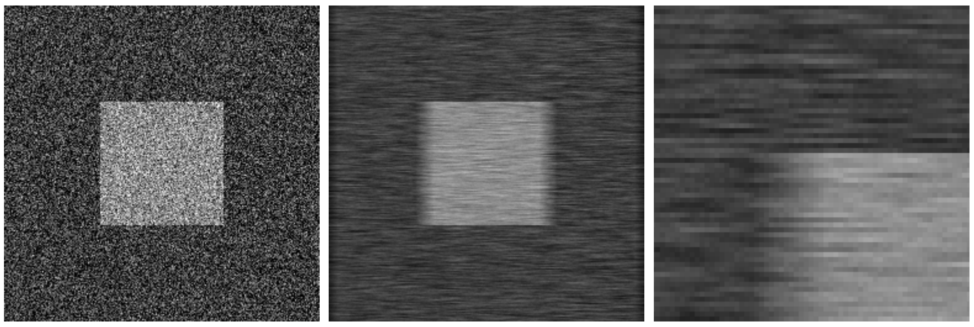


Figure 2.3: Blur along edge direction preserves edge, across destroys edge. Left: artificial noise image. Center and right: blur with a 1×15 average filter; detail.

2.1.3.2 Elongated filters

Noting that the Prewitt operators in (2.3) are equivalent to convolving the central difference operators in (2.5), a *pure derivative*, with a three pixels wide average filter along the edge direction, we see that operators such as Prewitt and Sobel contain an element of noise filtering. This makes these filters very popular for edge detection. However, if large amounts of noise or clutter are present, a three pixels wide low-pass filter is not sufficient to recover weaker edges.

To compute the relationship between filter size and noise attenuation, consider an observed one-dimensional signal $\hat{\boldsymbol{I}}$ given by the sum of a noise-free signal $\boldsymbol{I}$ and *i.i.d.* noise with Normal distribution, $\boldsymbol{n}(x,y) \sim \mathcal{N}\left(0,\sigma^2\right)$, *i.e.*, $\hat{\boldsymbol{I}} = \boldsymbol{I} + n$. The convolution of the $\hat{\boldsymbol{I}}$ with kernel $\boldsymbol{K}$ originates a signal with sample variance

$$\begin{aligned}\mathrm{Var}\left(\hat{\boldsymbol{I}} * \boldsymbol{K}\right) &= \mathrm{Var}\left(\boldsymbol{I} * \boldsymbol{K}\right) + \mathrm{Var}\left(n * \boldsymbol{K}\right) \\ &= \mathrm{Var}\left(\boldsymbol{I} * \boldsymbol{K}\right) + \sum_{i=0}^{M-1} \boldsymbol{K}\left(i\right)^2 \mathrm{Var}\left(n\left(x-i\right)\right) \\ &= \mathrm{Var}\left(\boldsymbol{I} * \boldsymbol{K}\right) + \sum_{i=0}^{M-1} \boldsymbol{K}\left(i\right)^2 \sigma^2.\end{aligned}$$

If all coefficients of kernel $\boldsymbol{K}$ are constant, $\boldsymbol{K} = 1/M$, we have

$$\mathrm{Var}\left(\hat{\boldsymbol{I}} * \boldsymbol{K}\right) = \mathrm{Var}\left(\boldsymbol{I} * \boldsymbol{K}\right) + \frac{\sigma^2}{M},$$

where the variance due to noise is σ^2/M, which quantifies the intuitive idea that larger low-pass filters provide increased noise reduction.

If, instead of convolving the central difference operators with 1×3 average filters along the edge direction to create the Prewitt operator, we convolve the central difference operators with a $1 \times M$ average filter, the resulting filter is better at dealing with noise but other issues arise: Firstly, since the resulting filter has a very sharp orientation, it has a narrow angular response. Freeman and Adelson [Freeman 1991] used the term steerable filter to describe a class of filters in which a filter of arbitrary orientation is synthesized as a linear combination of a set of basis filters and noted that many filters (defined as rotated copies of original basis filters) are needed to take all orientations into account. In fact, for a kernel $\boldsymbol{K}$ consisting of an elongated average filter of size $1 \times M$ to span the entire $180°$ angular range and to take every pixel in the perimeter of a window of radius $r = (M-1)/2$ into account, the number of directions that need to be analyzed is

$$N \geq \pi r, \tag{2.7}$$

if distributed uniformly. As illustration, an average filter of size $M = 15$ requires $N = 22$ directional filters. Secondly, filters with large footprints originate imprecise localization since every transition is effectively smeared by the large point spread function of the filters, requiring non-trivial mechanisms to deal with it.

Elongated filters have been exploited in recent edge detection methods to compute *significantly different* oriented means in an image by using a variety of filter banks of various lenghts and orientations and in various resolutions to improve the detection of faint edges, *e.g.*, rectangular filters [Galun 2007], curvelets [Gebäck 2009], shearlets [Yi 2009], and beamlets [Xiaoming 2007]. Although these filters enable a better recovery of weak edges than standard edge detectors based on Sobel or Prewitt operators, the particular implementation of the methods make use of multiple resolutions, which has a tendency to eliminate smaller details and is difficult to implement. Also, these methods discriminate opposing sets of pixels based only on their sample mean, neglecting their sample median or variance, which contains useful data for edge descrimination, as we show in chapter 3.

2.1.3.3 Statistical edge detection

The process of low-pass filtering an image and derivation can be seen roughly as computing the sample mean of two opposing sets of pixels and subtracting them. If the standard deviation of the noise or clutter in the image would be constant and equal to σ, the threshold for detecting an edge for a particular confidence interval could be defined simply as $C \propto \sigma/\sqrt{M}$. However, because the standard deviation of the data in an image is often not constant, it is useful to take a more comprehensive statistical analysis of the pixel set sample values to improve edge detection.

Such studies were initiated by Bovik et al. [Bovik 1986], which used two-sample statistical tests in the context of edge detection. Two-sample tests take two opposing sets of pixel values and considers that they are samples of two underlying distributions, $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$. The test then checks the null hypothesis $\mathcal{H}_0$ that the underlying probability distributions are in fact the same [Sheskin 2007]. An edge exists between the samples of $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$ if the distributions are deemed different.

If a parametric test is used (assuming that the pixel intensity values follow a Normal distribution), the parameters of each distribution can be estimated from the samples by computing $\hat{\mu}_i = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{x}_{ij}$ and $\hat{\sigma}_i^2 = \frac{1}{M-1}\sum_{j=1}^{M} (\boldsymbol{x}_{ij} - \hat{\mu}_i)^2$, with $i \in \{T, B\}$, $\boldsymbol{X}_T \sim \mathcal{N}(\mu_T, \sigma_T^2)$ and $\boldsymbol{X}_B \sim \mathcal{N}(\mu_B, \sigma_B^2)$. A typical test is the t-test, that assumes that the distributions are the same if their means coincide [Sheskin 2007]. Since the sample mean varies with the sample variance and the number of samples through formula $\sigma_{\hat{\mu}_i} = \hat{\sigma}_i/\sqrt{M}$, the t-test is given by

$$t = (\hat{\mu}_T - \hat{\mu}_B) \Big/ \sqrt{\frac{\hat{\sigma}_T^2 + \hat{\sigma}_B^2}{M}} \text{ [Sheskin 2007].} \qquad (2.8)$$

Once a t-value is determined, the test computes the probability that the test statistic would take a value at least as extreme as the one observed, denoted p-value, for $M - 1$ degrees of freedom. If the p-value is below the threshold chosen for statistical significance (usually 0.10, 0.05, or 0.01), the distributions are deemed different (two-tailed test) or larger than the other (one-tailed test) [Sheskin 2007].

If no assumption is made regarding the distributions of $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$, nonparametric tests can be used instead. These tests are more general and robust to outliers but typically computationally more intensive, since many require expensive sorting operations. Some tests compute empirical cumulative distributions (ecd) from samples $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$ and then compute the distance between them: the Kolmogorov–Smirnov test obtains the maximum distance between the ecds and the Fisz–Cramér–Von Mises test integrates the squared difference between the ecds. The Wilcoxon Mann–Whitney is a popular rank order test that mixes the samples of both distributions, sorts and ranks them. The difference between the distributions is assessed by adding the ranks of both distributions and comparing them [Sheskin 2007].

The success of two-sample statistical tests in edge detection is shown in [Lim 2002]. Reference [Fesharaki 1994] uses a t-test for detecting edges and a mixture of Normal distributions models noisy data in the edge detector of [Thune 1997]. Various statistical tests are used in the neural network approach [Williams 2006]. Despite the success of statistical edge detection, the small footprints in current methods limit their ability to handle the noise or clutter in complex images.

2.2 Line Segment Extraction

2.2.1 Motivation

In the path to extracting numerical or symbolic information from images, it is useful to decompose an image into components that belong to one or another simple family. For example, we might want to identify circles, line segments or other groups described by models. Line segments are fundamental low-level features for the analysis of many real-life images. Line segments are especially relevant because most man-made objects are made of flat surfaces, originating images with edge maps composed by line segments, and also because many shapes accept an economic description in terms of line segments. Thus, these segments provide important information about the geometric content of the imaged scene. This has been exploited, *e.g.*, for localizing vanishing points [Kosecka 2002] or to match line segments across distinct views [Schmid 1997]. Since more elaborated shapes are often described in an economic way in terms of line segments, their extraction is often a first step in many other problems, *e.g.*, rectangle detection [Micusík 2008], the inference of shape from lines [Slater 1996], map-to-image registration [Krüger 2001], 3D reconstruction [Liu 2011], or even image compression [Fränti 1998].

Although the problem of extracting line segments from images in an automatic way has been the focus of attention of several researchers in the past decades, current solutions make use of strong implicit assumptions that limit their applicability to simple and often non-realistic scenarios. Typical assumptions are that, *e.g.*, line segments are pronounced, thin, occur in small amounts, do not cross each other, are located away from noise or clutter. Some methods also assume that images have little data apart from line segments, such as textures or contours. Since these assumptions often fail in realistic scenarios, many line segments are often not detected, or are broken in two or more pieces. In more severe cases, extraction can fail altogether. For this reason, which we detail in the sequel, the robust detection of line segments in realistic scenarios remains an open frontier (see [Du 2010, von Gioi 2010, Borkar 2011, Ji 2011] for examples of recent advances).

2.2.2 The Hough Transform

The Hough transform (HT) [Hough 1962, Duda 1972] is arguably the most popular method to detect lines in images. The classical Hough technique is useful if little is known about the location of particular curves we wish to detect and their shape can be described parametrically (*e.g.*, a two-dimensional parameter of a straight line). Its input is an edge map such as the Canny edge detector, and it extracts the parameters that correspond to the largest number of edge points. In a situation where nothing is known about the location of the curves, each edge point contributes by restricting the possible locations of the curves to those that include it. The HT then works roughly by collecting all the evidence put forward by each edge point and obtaining the parameters that explain the greatest amount of evidence.

For example, consider the point (x', y') in Fig. 2.4 and the equation for a line $y = mx + c$. What are the lines that can pass through (x', y')? All the lines with (m, c) satisfying $y' = mx' + c$. Regarding (x', y') as fixed, the last equation is that of a line in parameter space (m, c). Repeating this reasoning, a second point (x'', y'') will also have an associated line in parameter space and both will intersect at point (m', c'), which corresponds to the line AB. All points on the line AB will yield lines in parameter space which intersect at the point (m', c').

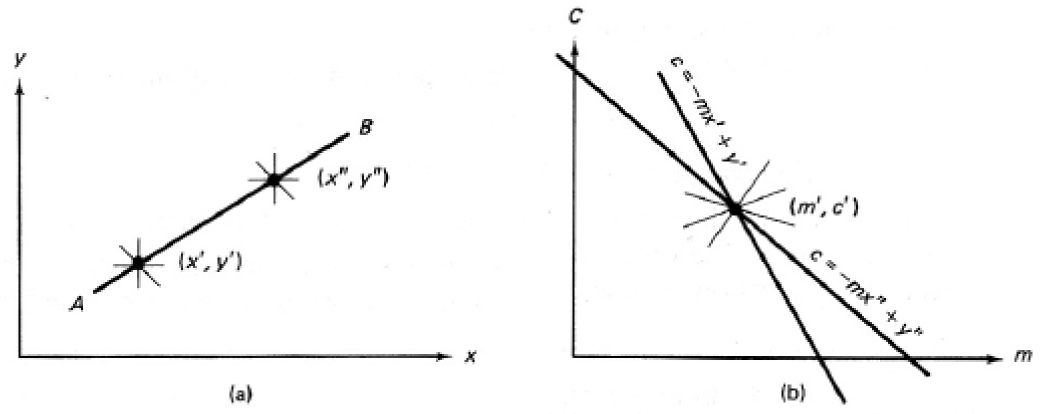


Figure 2.4: A line (a) in image space; (b) in parameter space (figure taken from [Ballard 1982]).

Since m may be infinite in the slope-intercept equation $y = mx + c$, a better parameterization of the line is given by its polar equivalent [Duda 1972], $x \cos\theta + y \sin\theta = r$, which produces a sinusoidal curve in (r, θ) space for fixed (x, y).

More concretely, the HT works in two steps. In the first step, the Hough transform starts by determining the pencil of all the lines that go through each edge point, which correspond to a set of two-dimensional parameters. Each of these parameters is used to access a coordinate in a two-dimensional space typically denoted as *Hough space* or *accumulator array* and add a vote. By processing all edge points, the votes for each location in the Hough space are accumulated and the locations with larger number of votes correspond to the most likely parameterizations of the lines in the image. The second step then consists of identifying the two-dimensional coordinates with the largest amount of votes. These steps are illustrated on Fig. 2.5.

The success of the HT comes from its global nature, since all points in a line contribute to its detection. As indicated in [Goldenshluger 2004], which pursued a statistical study of the HT, it is a statistically robust estimator for finding lines.

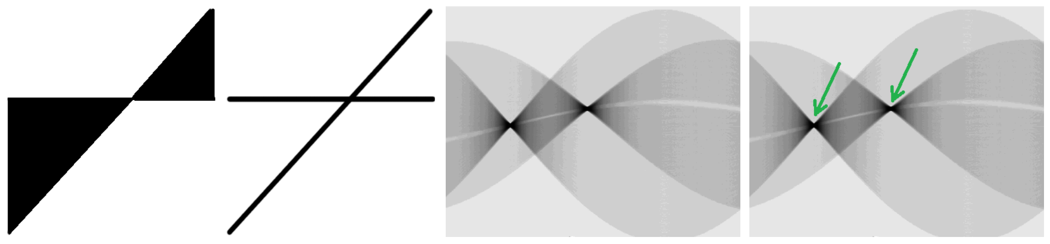


Figure 2.5: Steps of the HT method. From left to right: an original image, the respective edges, the Hough space with vote accumulations, the detected vote peaks.

It was shown that, although the convergence rates of the HT are seen to be slower than those found in some standard regression methods, the HT estimator is shown to be more robust as measured by its breakdown point.

There are, however, several practical problems with the HT:

• *Selecting the accumulator array resolution* – As mentioned above, the votes originated by all edge points in the image are stored in the accumulator array. The selection of the appropriate resolution for this array is difficult. If the resolution is too coarse, many quite different lines may be mapped to a single bucket, *i.e.*, a coordinate in the accumulator array, which can lead to an erroneously large amount of votes and the extraction of erroneous lines. If the resolution is too fine, small errors in the location of edge points (caused by, *e.g.*, edge points being located in the discrete pixel grid) may cause the votes to be dispersed by neighboring buckets, leading to line misdetections as no bucket concentrates a large amount of votes. Finer accumulator array resolutions also require a lot of storage memory and lead to increased computational costs. A proportionally higher amount of votes needs to be computed and stored for each edge point, such as the number of values in the accumulator array that need to be scanned through in search of local maxima.

Many efforts have been made to alleviate these problems. They include efficient accumulator methods, based on the observation that it is necessary to have high accumulator resolution only in places where a high density of votes accumulate. Examples include hierarchical schemes [Li 1986], multiple accumulator resolutions [Illingworth 1987a] and a probabilistic formulation [Stephens 1991]. The edge gradient can also be used [Illingworth 1987b, Zhou 2006] to reduce the number of votes to be processed, with the drawback of increasing the reliance on good edge information [Illingworth 1987b]. Some approaches reduce the computational cost by sub-sampling the edge map (randomized HT) [Kälviäinen 1995].

• *Low signal-to-noise ratio in Hough space* – Since all votes originated by an edge point are *wrong* except for the vote which corresponds to the actual segment, these wrong votes have no positive contribution to the detection of local peaks in the accumulator array. Instead, they constitute noise. The amount of noise in the Hough space becomes very significant for complex images with many lines and edge points [Kim 1998]. This makes it difficult to identify the most likely parameterization of actual lines in the Hough space. This is particularly critical for short segments, since they originate small peaks that are hard to identify [Guil 1995]. The limitations of the HT in dealing with complex images have been pointed out by several authors, *e.g.*, [Guil 1995, Guru 2004, Lee 2006, von Gioi 2010].

Accidental alignments of unrelated edge points can also originate false peak detections [Kim 1998]. Even in images with fewer edge points, it is usually possible to find many quite good phantom lines in a large set of reasonably uniformly distributed edge points. For example, regions of texture can generate peaks in the voting array that are larger than those associated with the lines sought [Guil 1995].

Fig. 2.6 shows an example where Hough space contamination due to vote accumulations lead to complete extraction breakdown. The large number of edge points in the real image, many of them forming very short segments or due to texture or noise, originates large numbers of votes that do not correspond to real segments. As a consequence, the accumulator array does not exhibit prominent peaks and its processing originates a poor result, where even longer lines are not extracted.

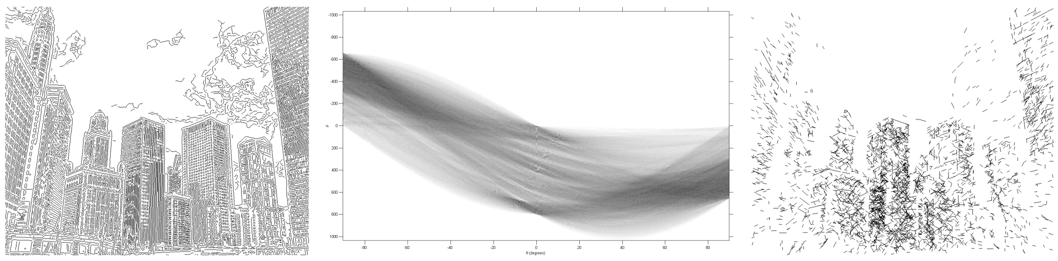


Figure 2.6: Limitation of the HT in detecting line segments in a complex image. Left: the input to the HT, the edge map obtained by applying the Canny edge detector to the real image shown in the top left of Fig. 5.4); middle: the HT accumulator array (note the absence of discernible peaks); right: detected line segments.

A common approach to deal with excessive wrong votes is to use the edge direction to reduce the accumulation of spurious votes, with the drawback of increasing the reliance on good edge information, as indicated before and in [Illingworth 1987b]. Other approaches sub-sample the total number of edge points that are processed and/or the votes that each edge point originates (randomized HT) [Kälviäinen 1995], which has the effect of reducing the number of wrong votes but also of the number of correct votes, leading to approximately the same signal-to-noise ratio when identifying local maxima in the accumulator array. Reference [Guil 1995] proposes a strategy that is based on processing a line a time: after detecting the line corresponding to the largest peak in the accumulator array, the votes of the edge points that belong to this line are removed. However, in many practical situations, when facing highly textured images and/or in the presence of noise, these approaches are still unable to provide accurate results. Although Duda and Hart [Duda 1972] argued as early as 1972 that the noise in the Hough space can be reduced by taking connectivity between collinear points into account, this topic received little attention in the past (exceptions are [Yuen 1991, Kim 1998]).

• *Reliance on local edge detection* – The HT requires an edge detector (such as [Canny 1986]) to generate its input edge map. As illustrated in Section 2.1, edge detection is by itself a hard problem, recognized as ill-posed in general [Bertero 1988]. To keep edge localization error to the minimum, edge detectors typically make use of small noise reduction filters, which makes the resulting edges very prone to noise. This creates a delicate balance between detecting spurious edge points in noisy or textured areas and missing faint edges. As a consequence, the threshold for accepting an edge point is typically set to a high value in common edge detectors, resulting in partial or complete segment misdetections. Although the global nature of the HT makes it a robust estimator in theory [Goldenshluger 2004], its reliance on local noise prone edge detectors counteracts its global nature.

• *Only extracts thin lines* – Another issue of the HT is that it cannot deal properly with wide line segments [Yang 1997], *i.e.*, segments made up of blurred edges, since the highest number of votes corresponds to the diagonal of the segment rather than the segment itself. Some methods assume constant line thickness and others address the extraction of various widths [Song 2005, Jang 2001, Yang 1997].

• *Not suitable for extracting line segments* – The above description and analysis of the HT refer to the extraction of lines. For the extraction of line segments, HT-based methods typically work by using the standard HT to obtain the most likely lines in the image and then a post-processing that obtains the start and end points of segments. This post-processing can make use of a gap-and-length method [Duprat 2005], the shape of the spread of votes in the Hough space [Kamat-Sadekar 1998, Ji 2011] or extra accumulators [Teutsch 2011]. A gap-and-length method works by taking the edge points corresponding to an identified vote peak and obtaining segments where consecutive edge points are separated by at most a certain *gap* and are larger than a minimum *length*.

The way the HT is adapted to extract line segments, *i.e.*, by first taking the output of the standard HT and then obtaining the start and end points of the segments, originates issues of its own. While initially each point in the Hough space supported the existence of a line, now the votes can refer to multiple collinear line segments of various lengths and different start and end points. The votes of individual line segments cannot be distinguished because only the total number of collinear edge points is stored in a Hough space. This means that a local maximum in the Hough space does not imply a maximum likelihood that line segments actually exist in the image with such parameterization — this adaptation of the HT is not a statistically robust line segment estimator.

Fig. 2.7 illustrates how the accumulation of votes does not guarantee good segment detections (as it guarantees good line detections). If this edge image would be used as input to the HT, it would lead to the detection of the lines marked in red as the three most voted lines. However, there are no connected edge points in either of these three lines. This means that the largest peaks in the Hough space do not correspond to the largest line segment in the image. In fact, the only true line segment in this illustration, the one marked in green, would correspond to the fourth most voted peak or more.

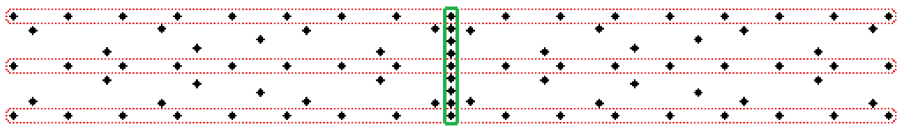


Figure 2.7: Illustration of an edge image and how the three most voted lines, marked in red, actually contain no line segment. The only line segment, marked in green, is the fourth largest peak or more.

In sum, current HT-based methods are not able to tackle the fundamental issues of the HT in extracting line segments in complex images. HT methods cannot deal with the large amount of edge points in complex images and rely too heavily on noise-prone edge detection. Furthermore, the poor adaptation of the HT to deal with line segments eliminates the attractive robustness feature.

2.2.3 Other robust methods: RANSAC, full search

The ability to recognize *the big picture* and overcome local imperfections due to noise and clutter makes global methods attractive. However, few papers apart from those dealing with the HT have approached the problem of developing global methods to extract, simultaneously, the line parameters and its extremes. A fruitful example is the method in [Desolneux 2006], which searches among all possible line segment candidates, using a Helmholtz principle for validating. Naturally, the good results come at a high computational cost. Another global approach is the widely known random sample consensus (RANSAC) [Fischler 1981]: two edge points from the edge map, randomly sampled, define a candidate line; then, the consensus of the line is evaluated by counting the number of other edge points that fit that line segment, given an error tolerance; for segments with a high consensus, the parameters could be refined by using an iterative expectation-maximization (EM) method [Hartley 2004]. However, since a consensus criteria able to cope with complex images is not trivial and the success of RANSAC hinges on the usage of a very large number of samples, these approaches result computationally too complex for many realistic applications.

2.2.4 Local methods

For the reasons above, the majority of methods for line segment extraction rely on local decisions, rather than global ones (see [Nevatia 1980, Burns 1986, Guru 2004, Nguyen 2007, von Gioi 2010] for examples). These local methods outperform the HT by taking connectivity into account, and result computationally simple, but lack robustness to deal with challenging situations, *e.g.*, when line segments cross. Their local nature makes long line segments particularly difficult to extract in many realistic scenarios, because, due to noise and clutter, these segments are interrupted.

The majority of local methods use three steps: first, obtaining a region of connected edge points; then, roughly estimating the line segment direction; and finally, refining and extending the segment by including new edge points that approximately fit the line. The data that is present at each step is illustrated in Fig. 2.8.

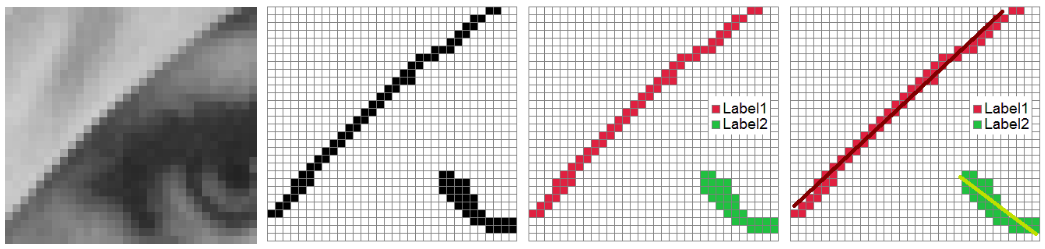


Figure 2.8: Typical data that is present at each step of local methods. From left to right: an original image, detected edges, labeled edges and fitted line segments.

The first step consists of chaining edge points [Etemadi 1992]. Methods such as the one in [Faugeras 1992] even skip the subsequent line fitting and refinement steps by chaining connected edge points into curves and then cutting them into line segments, using a straightness criterion. Texture, low-contrast regions, crossing segments, and noise make difficult the extraction of large connected regions belonging to a single segment. The second step consists of fitting a line to the chain of edge points using, *e.g.*, total least-squares (TLS) [Nguyen 2007]. Naturally, the reliability of the regression depends on the length of the underlying point chain. Some methods bypass the chaining of edge points: [Wilson 1979] uses the so-called local HT (LHT) [Xiao 2003], roughly estimating the segment direction from the peaks of local orientation histograms, computed at each edge point; [Guru 2004, Arras 1997] directly fit a line to all edge points inside a sliding window, which only provides reasonable estimates for simple scenes, with very small clutter. The final step usually involves alternating between two stages until convergence [Nguyen 2007]: inclusion of new edge points that are close to the candidate line, according to a distance measure; and re-estimation of the line segment parameters from the new set of edge points. As is typical with this type of methods, a poor initial estimate of the line segment may compromise the final result. Furthermore, the process may terminate too early when attempting to extract a long line segment, due to the cluttered nature of the edge maps of real images. Two popular local methods for line segment detection are [Burns 1986] and the LSD (Line Segment Detector) of [von Gioi 2010]. The method in [Burns 1986] coarsely quantizes the local orientation angles, chains adjacent pixels with identical orientation labels, and fits a line segment to the grouped pixels. LSD [von Gioi 2010] extends this idea by using continuous angles and eliminates false line segment detections with the Helmholtz principle of [Desolneux 2006].

2.3 Multi-texture Discrimination

2.3.1 Motivation

Texture is an important and ubiquitous property of regions of various types of images. Image texture analysis impacts then many fields. In remote sensing, the textures of multispectral images, taken from satellites or aircraft, provide information about the sensed scene. In biomedicine, magnetic resonance images, ultrasound images of the human body, and microscopic images of cell cultures or tissue samples, often contain regions of different textures, whose analysis is required. The different textures present in natural images (outdoor scenes, object surfaces, etc) has motivated the computer vision community to address the texture analysis problem.

Although texture analysis has been the focus of attention of several researchers in the past decades, current solutions are either not very discriminant or are computationally complex, thus not suited for applications where both characteristics are needed. Optimized filters have been developed to deal with this problem but are most effective when the number of textures to discriminate is small — their computational complexity increases dramatically in scenarios with multiple textures. For this reason, simple and robust discrimination of textures remains an open frontier (see, *e.g.*, [Mirmehdi 2008] for examples of recent advances).

2.3.2 Overview of texture discrimination methods

The most popular texture discrimination methods use Gray-Level Co-occurrence Matrices (GLCM), Gabor Filter Banks (GFB), or Local Binary Patterns (LBP). GLCM [Haralick 1973] create a $2^8 \times 2^8$ matrix (for 8-bit precision) for each texture patch, where each entry counts the number of co-occurring image intensity pairs of neighboring pixels. From this matrix, a vector of so-called Haralick features (Energy, Contrast, Dissimilarity, ...) is computed and used as the texture descriptor (the distances between texture patches is usually measured by the simple Euclidean norm of the vector difference). GFB compute, for every pixel, the output of about ten to fifty filters. These large dimensional vectors may either make up local energy estimates [Jain 1991] or be clustered and represented by histograms [Malik 2001] (Euclidean or histogram distance metrics are then used for classification). When using LBP [Ojala 2002], the intensity values of each set of P pixels in a texture patch are thresholded and stored into a binary pattern, which addresses a 2^P look-up table. Each entry then indicates if the pattern has few 0–1 or 1–0 transitions and their type. A histogram accumulates this information for the entire patch and histogram distance metrics are used for classification. The discrimination success of GLCM, GFB, and LBP is measured in the comparative study of [Partio 2007] as being 66.7%, 92.2%, and 90.7%, respectively. Furthermore, the computational complexity of GLCM and GFB restrict their application in several real-life scenarios.

Usually, methods for texture discrimination are organized into five major categories [Tüceryan 1993]: statistical, geometrical, structural, model-based, and signal processing. Early work in texture discrimination utilized statistical methods to extract features from the image. Examples of these methods are GLCM, LBP, and those using autocorrelation [Toyoda 2005] or power spectrum [Chetverikov 2000]. Geometrical methods seek to extract texture elements and infer their placement rule. They include Voronoi tessellation features [Tüceryan 1990] and structural methods [Yalniz 2010]. Model-based methods are based on the construction of an image model that can be used not only to describe texture, but also to synthesize it. These include random fields, such as the Gauss Markov, or the Gibbs distribution texture models [Wilson 2003], and fractals [Xia 2006]. Signal processing methods use spatial filters, *e.g.*, GFB and Laws filters [Laws 1980]. Detailed reviews of methods for texture analysis can be found in [Tüceryan 1993] and [Mirmehdi 2008].

Studies of the Human Visual System [Tüceryan 1993] support signal processing methods, by showing that it analyzes the retinal image, decomposing textured areas into their frequency and orientation components. Earlier attempts at defining spatial domain discrimination methods concentrated on measuring the edge density per unit area, usually by using simple edge masks such as the Roberts or the Laplacian operators [Husoy 1993]. Fine textures tend to have a higher density of edges per unit area than coarser ones. In a pioneering work by Laws [Laws 1980], a bank of 25 separable bandpass filters was applied. This joint spatial–frequency representation captures image characteristics at several scales, providing meaningful information for texture analysis. Subsequent works have focused on using different filter bank families and processing filter outputs in distinct ways.

Fig. 2.9 shows the typical block diagram of a signal processing method for texture discrimination. Usually, the filtering step consists of a bank of linear filters. These filters should exhibit appropriate frequency and orientation selectivity, so that the energy of their outputs is approximately constant within the same texture region but different for distinct textures. Fig. 2.10 displays the normalized orientation-frequency responses of common filter banks: Laws, GFB, and ring and wedge filters [Coggins 1985]. Since each filter bank is better suited to textures of particular characteristics, it is not trivial for the algorithm designer to choose the appropriate one. As a consequence, a frequent solution is to resort to a large number of filters (in practice, GFB requires about ten to fifty filters, computed at every pixel). The local energy function is usually divided into a nonlinearity and a smoothing function, as represented in Fig. 2.9. Several nonlinearities have been used, *e.g.*, magnitude [Laws 1980], square (power), and sigmoid (tanh) [Jain 1991]. A low-pass Gaussian filter is a common smoothing function. The most common classifier is the nearest neighbor, using pre-computed class centroids, which corresponds to the Maximum Likelihood classifier when the class conditionals are Gaussian with identical covariance. In [Malik 2001], a computationally more complex method that has shown to perform well uses K-means to cluster GFB filter outputs. Then, the resulting histograms are compared using metrics such as χ^2 or the Earth Movers Distance.

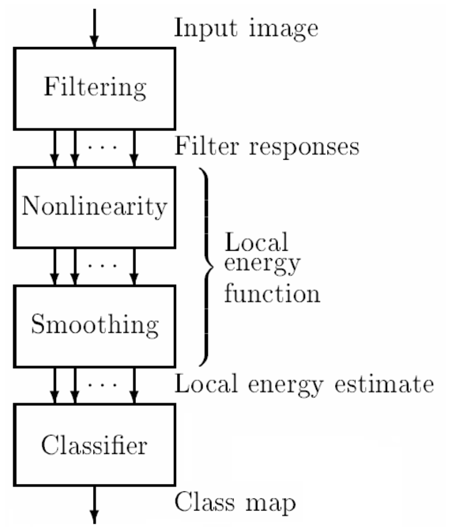


Figure 2.9: Block diagram of signal processing method for texture discrimination

2.3.3 Optimized filters for texture discrimination

The difficulties in selecting few but highly discriminant filter banks motivated research into the automatic optimization of filters, tuned to deal with particular sets of textures. We provide an overview of such methods in the sequel.

2.3.3.1 Eigenfilters

Reference [Ade 1983] computes a matrix for each texture, obtained from its autocorrelation. Then, the most significant eigenvectors are used as image filter banks, their output is squared and smoothed with a Gaussian filter. This method requires a large number of filters, typically 5 to 9 per texture (so that the corresponding eigenvalues sum up to at least 99% of the total). Despite requiring a large number of filters, the study in [Randen 1997] reports only a medium discrimination ability, suggesting that an optimal texture representation does not necessarily correspond to an optimal image discrimination.

2.3.3.2 Optimal FIR filters

A two-texture design approach using FIR filters is proposed in [Mahalanobis 1994]. This method achieves optimal discrimination by taking the maximum ratio between the mean of squared filter outputs as criterion. This work was later extended to use a criterion proposed by Unser [Unser 1986] and the Fisher criterion, which takes feature variances into account. Experimental results show that these criteria enable highly discriminant FIR filters [Randen 1997].

To extend the design criteria above to multiple textures, a filter for all possible pairing of textures is computed, which results in, *e.g.*, 45 filters for 10 textures or 190 filters for 20 textures. Reference [Randen 1997] groups textures and uses a smaller set of filters to recursively select the most likely class. Although fewer filters are used than in the complete pairwise comparison scenario, the computational complexity is still high and the discrimination ability is reported as average.

2.3.3.3 Prediction error filtering

Reference [Randen 1997] uses a linear predictor for each texture class, learned with least squares optimization. After training, a texture is assigned to the class with the smallest linear prediction error. Despite the potentially high complexity in a multi-texture scenario (the number of filters is equal to the number of textures), reference [Randen 1997] reports modest results for these filters, which may possibly be explained, again, due to the emphasis on optimal representation rather than discrimination.

2.3.3.4 Optimal representation Gabor Filter Banks

Tuning the central frequencies of a narrow-band GFB to the spectral peaks of the textures is proposed in [Bovik 1991]. They introduce a semi-automatic (claimed to be easily extended to fully automatic) procedure for determining the spectral peaks. Despite the high dimensionality of the feature vector (two filters per texture are used), the study in [Randen 1997] reports that classification accuracy of [Bovik 1991] is similar to significantly worse than the best optimal filtering approaches due to representation not implying good discrimination.

A GFB design scheme yielding filters optimized for texture separation in a two-class scenario is proposed in [Dunn 1995]. The optimal center frequency is determined by evaluating a large range of center frequencies, using the Fourier transform, and selecting the best candidate. This approach is extended for an arbitrary number of textures in [Weldon 1996], with a manually defined filter number. The study in [Randen 1997] concludes that the two-texture method originates good results (except when dealing with similar textures) but that a large number of filters is needed to obtain good discrimination in the multi-texture scenario.

2.3.3.5 Neural networks

A feed forward neural network, trained through back-propagation, is proposed in [Jain 1996]. In this approach, image intensities are weighted and summed in the input nodes of the network, resembling a filter bank. The nonlinearities in the network also resemble the nonlinearity in the local energy function. Comparative tests reported in [Randen 1997] show significant training times and an average discrimination, except for very simple texture pairs. Reference [Tivive 2007] uses a more complex network, with 15 layers of neurons. Although computational complexity is significant, they report error rates on the Brodatz database of about 16-25%, which decrease with complex pre and post processing smoothing stages.

To conclude, results show that simple optimization schemes are available when the texture classes to discriminate are few. However, even for a moderate number of texture classes, *e.g.*, 10, the number of filters that are needed for accurate discrimination increases dramatically, approaching that of standard filter banks, with medium discrimination ability.

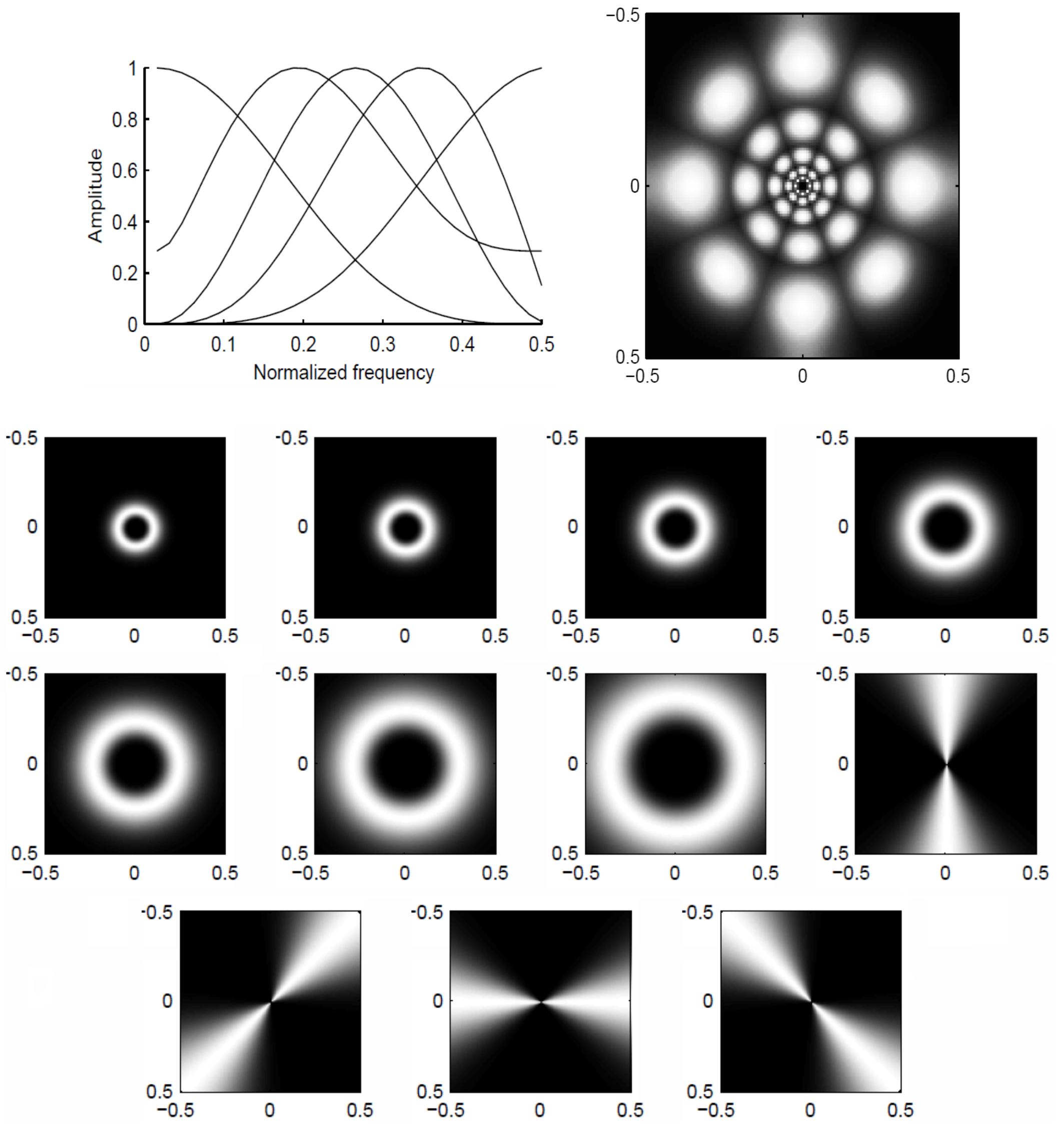


Figure 2.10: Normalized frequency responses of the Laws (top left), Gabor (top right), ring and wedge filter banks (bottom)

CHAPTER 3

Statistical edge detection using elongated oriented footprints

3.1 Proposed approach

The novel and key aspect of our approach is the combination of what we denote *contextual* and *local edges*, taking connectivity into account. Contextual edges are obtained with filters of large footprint, which are able to deal with noise in identifying image transitions. The footprint of a contextual filter is illustrated on the left of Fig. 3.1, and its application on an image to detect a contextual edge is illustrated on the right of Fig. 3.1.

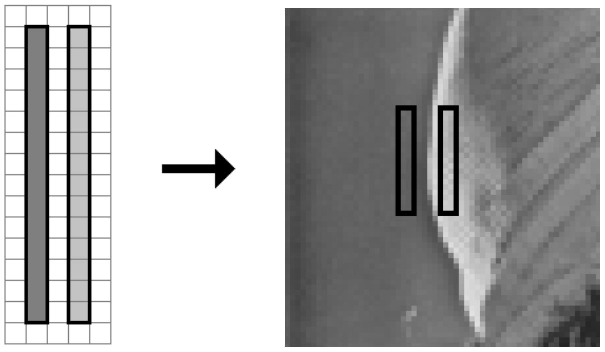


Figure 3.1: Footprint of a contextual filter (left) and its use in an image (right).

Although such large footprint filters are robust to noise, edge localization is imprecise since every transition is smeared by their large point spread function. On the other hand, local edges are thresholded image derivatives obtained with filters of very small footprint, *e.g.*, Sobel, Prewitt, Roberts and central difference operators. Since these filters are small, edges are located precisely but noise may originate erroneous detections. Our proposal combines contextual and local edges obtained at the same pixels by taking the sign of the contextual edge and using it to identify so-called valid local edges with the same sign. The edge sign indicates if it is a dark–to–bright transition or the opposite.

In complex images, valid local edges are disconnected from each other, due to noise and image clutter. To obtain connected edge maps, we handle connectivity explicitly in the combination process by checking if the valid local edges, along a given direction, are located at a distance not greater than a maximum distance threshold from each other. If so, the pixels corresponding to valid local edges and those in between are marked as edge points. The resulting edge detector has the robustness of contextual edges in dealing with noise and the localization of local edges, as idealized by Canny [Canny 1986].

Typical contextual edge detectors handle noise by applying a low-pass filter of large footprint such as, *e.g.*, Gabor or steerable filters [Freeman 1991], followed by a derivation step and binarization with a fixed threshold. A fixed threshold is often deemed sufficient because noise is assumed to be white and of constant amplitude on simple images. However, since complex images often exhibit high-frequency variations due to, *e.g.*, textures and clutter from interfering image data, a more comprehensive statistical analysis of the pixel intensity values is beneficial to overcome the use of a simple threshold. Our contextual edge detector collects pixel intensity values from elongated oriented footprints and uses a two-sample statistical test to determine if a contextual edge exists, by thresholding the confidence interval for the null hypothesis that both distributions are actually the same.

We propose two separate approaches for edge detection:

1. Non-paired Total Variation (TV) parametric test — A computationally simple running average approach is used to compute the sample mean and variance of pixel intensity values along elongated and oriented footprints. The parametric values at opposing positions are used to compute the TV distance between the distributions (analogous to computing a derivative) and determine if a contextual edge exists and its sign. The contextual edges are then used to locate valid local edges, *i.e.*, local edges with the same sign. Valid local edges whose distance is not greater than a maximum distance threshold are marked as edge points, plus the gap between them.

2. Paired nonparametric sign test — A computationally simple running average approach adds the signs of local edges along elongated and oriented footprints, counting the number of positive minus the number of negative transitions that occur around each pixel. If there is a predominance of positive or negative transitions, then a contextual edge exists — this implements the sign test. As in the case above, the contextual edges are used to locate valid local edges and those that are sufficiently close are marked as edge points, plus the gap between them.

Section 3.2 details the non-paired TV parametric test and section 3.3, the paired nonparametric sign test. We show experimental results on section 3.5.

3.2 Non-paired Total Variation parametric test

3.2.1 Computing Normal parameters: running average

Each direction $n \in \{1, \dots, N\}$ (with N given by (2.7)) corresponds to angle $\theta_n = 180°(n-1)/N$, which lies in the horizontal (H) or vertical (V) half of the semicircle,

$H(\theta_n)$, and in quadrant $Q(\theta_n)$. This is illustrated in Fig. 3.2 and defined as

$$H(\theta_n) = \begin{cases} \text{V} & \theta_n \in [45°, 135°[\\ \text{H} & \textit{otherwise} \end{cases} \tag{3.1}$$

$$Q(\theta_n) = \begin{cases} 0 & \theta_n \in [0°, 22.5°[\cup[157.5°, 180°[\\ 45 & \theta_n \in [22.5°, 67.5°[\\ 90 & \theta_n \in [67.5°, 112.5°[\\ 135 & \theta_n \in [112.5°, 157.5°[\end{cases}$$

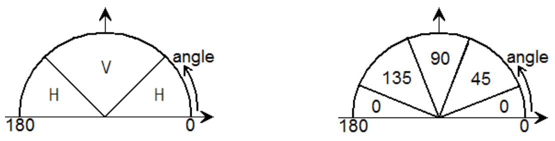


Figure 3.2: Left: half of a semicircle, $H(\theta_n)$; right: the respective quadrants, $Q(\theta_n)$.

For each direction θ_n, we divide the image into lines, as illustrated in Fig. 3.3. We compute the sample average and variance, $\hat{\boldsymbol{\mu}}(\boldsymbol{p}_m)$ and $\hat{\boldsymbol{\sigma}}^2(\boldsymbol{p}_m)$, of M consecutive pixels in the line, pixels $\boldsymbol{p}_m$ to $\boldsymbol{p}_{m+M-1}$, and store the results in pixel $\boldsymbol{p}_m$. Since the set of M consecutive pixels needed for the next point, $\boldsymbol{p}_{m+1}$, is the same as the set for point $\boldsymbol{p}_m$ except the points at the start and end of the sets, we use a recursion that simplifies calculations and is used often in practice: the **running average**.

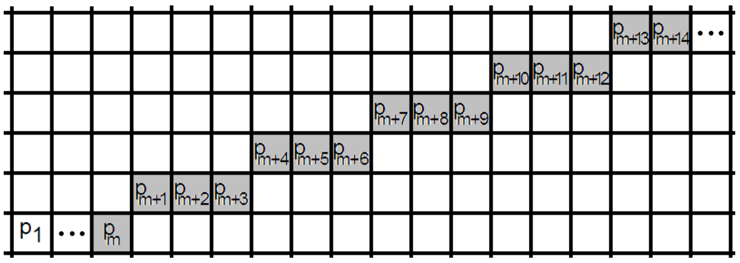


Figure 3.3: Illustration of $M = 15$ pixels starting at pixel $\boldsymbol{p}_m$ along direction θ_n.

For image $\boldsymbol{I} \in \mathbb{R}^{S_x \times S_y}$, let $\boldsymbol{A}$ address each line along direction θ_n,

$$\boldsymbol{A}(x, y, \theta_n) = \begin{cases} (x, y + [x \tan\theta_n] + \gamma S_y) & \text{if } H(\theta_n) = H \\ (x + [y \cot\theta_n] + \gamma S_x, y) & \text{if } H(\theta_n) = V \end{cases}, \tag{3.2}$$

where $[\cdot]$ is the rounding operation and $\gamma \in \mathbb{Z}$ is chosen so that $\boldsymbol{A}(x, y, \theta_n)$ lies inside the image limits. Methods such as Digital Differential Analyzer [Watt 2000] or [Wu 1991] introduce less aliasing, by using neighboring pixels and weighting their contribution approximately with their distance to the ideal point, but are more complex. If $H(\theta_n) = H$, the y-th line is made up of the pixels addressed by $\boldsymbol{A}(x, y, \theta_n)$, with $x \in \{1, \ldots, S_x\}$ and in increasing order. Analogously, if $H(\theta_n) = V$, the x-th line is addressed by $\boldsymbol{A}(x, y, \theta_n)$, with increasing $y \in \{1, \ldots, S_y\}$. To compute the average and variance in a recursive way, we define accumulators Φ_x and Φ_{x^2},

$$\begin{aligned} \Phi_x &\leftarrow \Phi_x - \boldsymbol{I}(\boldsymbol{p}_{m-1}) + \boldsymbol{I}(\boldsymbol{p}_{m+M-1}) \\ \Phi_{x^2} &\leftarrow \Phi_{x^2} - \boldsymbol{I}^2(\boldsymbol{p}_{m-1}) + \boldsymbol{I}^2(\boldsymbol{p}_{m+M-1}), \end{aligned} \tag{3.3}$$

where $\boldsymbol{p}_m$ is the current pixel; $\boldsymbol{p}_{m-1}$ is the pixel that is leaving the set of M pixels, and $\boldsymbol{p}_{m+M-1}$ is the pixel that is entering the set,

$$\begin{aligned} &\boldsymbol{p}_m = \boldsymbol{A}(x, y, \theta_n), \\ &\boldsymbol{p}_{m-1} = \boldsymbol{A}(x-1, y, \theta_n), \quad \boldsymbol{p}_{m+M-1} = \boldsymbol{A}(x+M-1, y, \theta_n) \quad \text{if } H(\theta_n) = H \; . \\ &\boldsymbol{p}_{m-1} = \boldsymbol{A}(x, y-1, \theta_n), \quad \boldsymbol{p}_{m+M-1} = \boldsymbol{A}(x, y+M-1, \theta_n) \quad \text{if } H(\theta_n) = V \end{aligned} \tag{3.4}$$

With accumulators Φ_x and Φ_{x^2} updated at pixel $\boldsymbol{p}_m$, the parametric parameters of the Normal distribution for direction θ_n are then, for point $\boldsymbol{p}_m$,

$$\begin{aligned} \hat{\boldsymbol{\mu}}(\boldsymbol{p}_m) &= \frac{1}{M}\Phi_x \\ \hat{\boldsymbol{\sigma}}^2(\boldsymbol{p}_m) &= \frac{1}{M-1}\Phi_{x^2} - \frac{1}{M(M-1)}\Phi_x^2. \end{aligned} \tag{3.5}$$

3.2.2 Total Variation distance

In the context of edge detection, two-sample tests consider opposing sets of pixels in the image as samples of two underlying distributions, $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$ (where T refers to the top area and B to the bottom area) and tests the null hypothesis $\mathcal{H}_0$ that the underlying probability distributions are the same [Sheskin 2007]. An edge exists if the distributions are deemed different. Unlike other statistical tests, the pixel samples are taken from elongated and oriented footprints, as illustrated in Fig. 3.4.

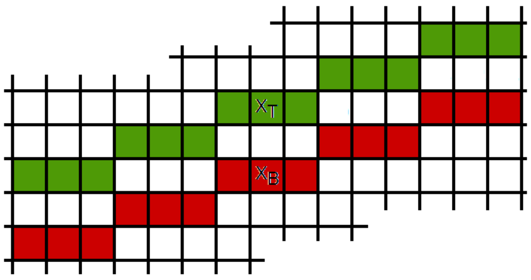


Figure 3.4: Elongated oriented footprints with samples $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$.

Although the t-test is a very popular parametric test, it assumes that coinciding mean values imply identical distributions (see equation (2.8)). Since a human viewer sees an edge when pixel distributions have the same sample mean but different sample variances, this test is not suitable for contextual edge detection. To enable this feature, we use the Total Variation (TV) distance [DasGupta 2010],

$$\delta(\boldsymbol{X}_T, \boldsymbol{X}_B) = \frac{1}{2}\int_{-\infty}^{\infty} \left| f(\xi; \hat{\mu}_T, \hat{\sigma}_T^2) - f(\xi; \hat{\mu}_B, \hat{\sigma}_B^2) \right| d\xi \in [0, 1], \tag{3.6}$$

where $f(\cdot; \mu, \sigma^2)$ represents the probability density function (pdf) of the Normal distribution. The TV distance is the integral of the (linear) distance between the pdfs of $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$. Unlike the Kullback-Leibler or Hellinger divergences, the TV indicates by how much the two pdf differ and in a linear way.

Fig. 3.5 shows the result of the TV distance when applied to illustrative distributions (the TV distance is the integral of the gray and black areas). On the left, both distributions are well separated and the TV distance is 1. On the middle, both distributions mostly overlap, so the TV distance is low. The right image of Fig. 3.5 illustrates a scenario in which two distributions with the same average but distinct standard deviations are deemed different by the TV distance.

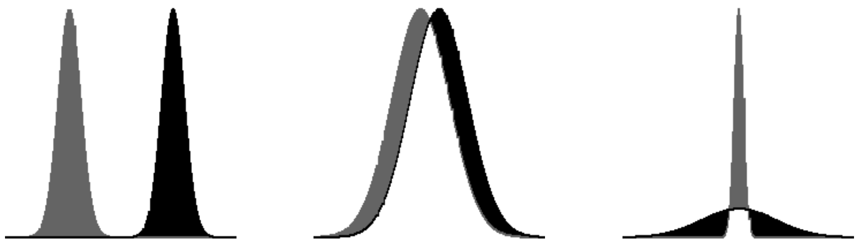


Figure 3.5: Application of the TV distance. Separate distributions (left, TV = 1); mostly overlapping distributions (middle, TV = 0.25); distributions with the same average but distinct standard deviation (right, TV = 0.75).

3.2.3 Computing the Total Variation distance

We now need a simple way to compute the TV distance (3.6). Let $\xi_i \in \{-\infty, \hat{\xi}_1, \hat{\xi}_2, \infty\}$ be the points where the pdf of distributions $\boldsymbol{X}_T$ and $\boldsymbol{X}_B$ cross, as illustrated in Fig. 3.6. These points help in dealing with the magnitude operator, using the cumulative density function of the Normal distribution, $F(\cdot; \mu, \sigma^2)$,

$$\begin{aligned}
\delta(\boldsymbol{X}_T, \boldsymbol{X}_B) &= \frac{1}{2}\sum_{i=1}^{3}\int_{\xi_i}^{\xi_{i+1}} \left| f(\xi; \hat{\mu}_T, \hat{\sigma}_T^2) - f(\xi; \hat{\mu}_B, \hat{\sigma}_B^2) \right| d\xi \\
&= \frac{1}{2}\sum_{i=1}^{3}\left| \int_{\xi_i}^{\xi_{i+1}} f(\xi; \hat{\mu}_T, \hat{\sigma}_T^2) - f(\xi; \hat{\mu}_B, \hat{\sigma}_B^2) d\xi \right| \\
&= \frac{1}{2}\sum_{i=1}^{3}\left| F(\xi_{i+1}; \hat{\mu}_T, \hat{\sigma}_T^2) - F(\xi_i; \hat{\mu}_T, \hat{\sigma}_T^2) - F(\xi_{i+1}; \hat{\mu}_B, \hat{\sigma}_B^2) + F(\xi_i; \hat{\mu}_B, \hat{\sigma}_B^2) \right| .
\end{aligned} \tag{3.7}$$

To determine $\hat{\xi}_1$ and $\hat{\xi}_2$, we make $f(\hat{\xi}; \hat{\mu}_T, \hat{\sigma}_T^2) = f(\hat{\xi}; \hat{\mu}_B, \hat{\sigma}_B^2)$, which results in $\hat{\xi} = \left(-b \pm \sqrt{b^2 - 4ac}\right)/2a$, where

$$\begin{aligned}
a &= 1/\left(2\hat{\sigma}_T^2\right) - 1/\left(2\hat{\sigma}_B^2\right)\\
b &= -\hat{\mu}_T/\left(\hat{\sigma}_T^2\right) + \hat{\mu}_B/\left(\hat{\sigma}_B^2\right)\\
c &= \hat{\mu}_T/\left(2\hat{\sigma}_T^2\right) - \hat{\mu}_B/\left(2\hat{\sigma}_B^2\right) - \ln\left(\hat{\sigma}_B/\hat{\sigma}_T\right).
\end{aligned}$$

If all parameters are equal, $\delta(\boldsymbol{X}_T, \boldsymbol{X}_B) = 0$. For equal sample variances, $\hat{\xi}_1 = \hat{\xi}_2 = \left(\hat{\mu}_T - \hat{\mu}_B\right)/2$. For different sample variances, two crossing points are obtained.

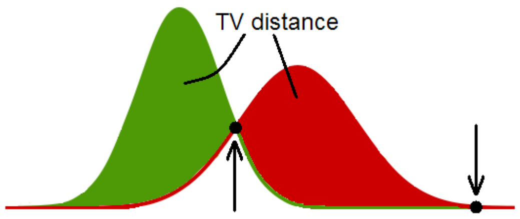


Figure 3.6: Pdf intersection points and area that is integrated in the TV distance.

We further simplify (3.7) by noting that only the way $\hat{\mu}_T$ and $\hat{\mu}_B$ relate to each other is needed, and only the ratio between $\hat{\sigma}_T$ and $\hat{\sigma}_B$. Let variables $\hat{\mu}'_B$ and $\hat{\sigma}'_B$ be

$$\hat{\mu}'_B = \frac{|\hat{\mu}_B - \hat{\mu}_T|}{\min(\hat{\sigma}_T, \hat{\sigma}_B)} \quad \hat{\sigma}'_B = \frac{\max(\hat{\sigma}_T, \hat{\sigma}_B)}{\min(\hat{\sigma}_T, \hat{\sigma}_B)} . \tag{3.8}$$

Assuming that $\hat{\mu}_T \leq \hat{\mu}_B$ and $\hat{\sigma}_T \leq \hat{\sigma}_B$, equations (3.8) become

$$\hat{\mu}_B = \hat{\mu}_T + \hat{\mu}'_B \hat{\sigma}_T \quad \hat{\sigma}_B = \hat{\sigma}_T \hat{\sigma}'_B. \tag{3.9}$$

Replacing (3.9) in (3.6) and substituting variable $\zeta = (\xi - \mu_T)/\sigma_T$, we obtain

$$\begin{aligned}
\delta(\boldsymbol{X}_T, \boldsymbol{X}_B) &= \frac{1}{2}\int_{-\infty}^{\infty} \left| f(\xi; \hat{\mu}_T, \hat{\sigma}_T^2) - f(\xi; \hat{\mu}_T + \hat{\mu}'_B \hat{\sigma}_T, \hat{\sigma}_T^2 \hat{\sigma}_B'^2) \right| d\xi\\
&= \frac{1}{2}\int_{-\infty}^{\infty} \left| f(\zeta; 0, 1)/\hat{\sigma}_T - f(\zeta; \hat{\mu}'_B, \hat{\sigma}_B'^2)/\hat{\sigma}_T \right| \hat{\sigma}_T d\zeta\\
&= \frac{1}{2}\int_{-\infty}^{\infty} \left| f(\zeta; 0, 1) - f(\zeta; \hat{\mu}'_B, \hat{\sigma}_B'^2) \right| d\zeta,
\end{aligned} \tag{3.10}$$

which shows that the TV distance depends only on two parameters. A two-dimensional look-up table $\boldsymbol{\Delta}(\hat{\mu}'_B, \hat{\sigma}'_B)$ is then filled in at the start of the program execution for various $(\hat{\mu}'_B, \hat{\sigma}'_B)$, with $\hat{\mu}_B \geq 0$ and $\hat{\sigma}'_B \geq 1$, using (3.7).

During (on-line) program execution, we obtain the coordinates above and below $\boldsymbol{p}$, $\boldsymbol{p}_T = \boldsymbol{A}(x_p - [\sin\theta_n], y_p - [\cos\theta_n], \theta_n)$ and $\boldsymbol{p}_B = \boldsymbol{A}(x_p + [\sin\theta_n], y_p + [\cos\theta_n], \theta_n)$. These coordinates coincide with the non-zero positions of the central difference operator for angle θ_n, $\boldsymbol{K}_{Q(\theta_n)}$, illustrating that the TV distance can be seen as a derivative operation that takes the variance into account. With these coordinates, we extract the parameters of the two distributions, $\hat{\mu}_T = \hat{\boldsymbol{\mu}}(\boldsymbol{p}_T)$, $\hat{\sigma}^2_T = \hat{\boldsymbol{\sigma}}^2(\boldsymbol{p}_T)$, $\hat{\mu}_B = \hat{\boldsymbol{\mu}}(\boldsymbol{p}_B)$, $\hat{\sigma}^2_B = \hat{\boldsymbol{\sigma}}^2(\boldsymbol{p}_B)$. We normalize them using equation (3.8) and access the two-dimensional look-up table containing the TV distances, $\boldsymbol{\Delta}(\mu'_B, \sigma'_B)$. The contextual edge value is computed by including the type of transition, $\delta(x, y, n) = \boldsymbol{\Delta}(\mu'_B, \sigma'_B)\mathrm{sgn}(\hat{\mu}_T - \hat{\mu}_B)$. A contextual edge exists if $|\delta(x, y, n)| \geq C$.

3.3 Paired nonparametric sign test

Despite the multiple sources of inaccuracies for edge detection such as noise and image clutter, there should be a predominance of either positive or negative intensity variations (corresponding to light-to-dark and dark-to-light transitions, respectively) along the edges. This predominance is what we exploit to compute contextual edges.

We start by computing the local directional content of an image $\boldsymbol{I}$ along directions $\{0°, 45°, 90°, 135°\}$, through the convolution with the central difference operators of equation (2.5). Then, an approximation of the sign is computed using

$$\boldsymbol{E}_\theta(x, y) = \begin{cases} 1 & \text{if } \boldsymbol{\nabla}_\theta \boldsymbol{I}(x, y) \geq T \\ -1 & \text{if } \boldsymbol{\nabla}_\theta \boldsymbol{I}(x, y) \leq -T \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.11}$$

By using $T > 0$, we avoid using the actual sign function, since experimental tests have shown that it originates many unwanted edges such as compression grid artifacts or small gradients in the image. Then, we propose two alternative but equivalent ways to collect evidence that there is a positive, negative or no transition at each pixel: using the running average approach, similar to that described in section 3.2.1, and; accumulating local responses in a histogram.

1. Running average approach — This approach is very similar to the one defined in section 3.2.1, with two differences. Firstly, only the mean value is computed and not the variance. The mean effectively *counts* the positive minus the negative transitions. Secondly, we do not compute the mean values of image intensities but instead of the four local edge maps, $\boldsymbol{E}_{Q(\theta_n)}(x, y)$, that are selected through angle θ_n and function $Q(\cdot)$, defined in equation (3.1). The method computes the mean value $\delta(x, y, n)$ of the directional map $\boldsymbol{E}_{Q(\theta_n)}$, for each pixel and direction θ_n. A contextual edge exists if $|\delta(x, y, n)| \geq C$.

Fig. 3.7 illustrates the results of applying the sign test to a set of $M = 15$ pairs of pixels. On the left of this figure, all the pixels on the left are larger than the respective pixels on the right. This means that all individual pixel comparisons return one, so the overall result is the addition of the fifteen ones. Because the overall result is that 100% if the pixels are showing a positive transition, there is a clear positive edge at this location. In the middle of this figure, the number of positive transitions is the same as the number of negative ones, therefore the addition of all contributions is zero and there is no contextual edge. On the right of this figure, there is a $6/15 = 40\%$ predominance of negative transitions, which may be classified as a negative contextual edge, depending on the threshold C.

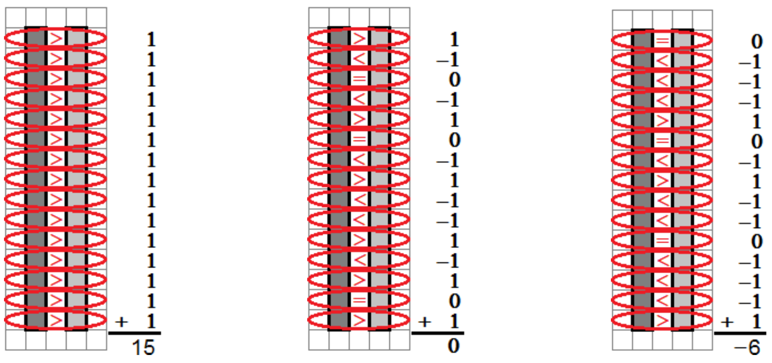


Figure 3.7: Illustrations of the sign test. If the pixel on the left is larger than the respective one on the right, the comparison contributes 1; if smaller, it contributes -1; if equal, contributes 0. The sign test aggregates all contributions and determines if there is a predominance of positive or negative results, or no predominance.

2. Accumulating local responses in a histogram — We compute local orientation histograms in pixels where a local edge exists, *i.e.*, where $|\boldsymbol{E}_{Q(\theta)}(x, y)| = 1$ for some θ. We use neighboring local edge points whose direction is coherent with its position. To clarify, when building a histogram for edge point (x_0, y_0), the directions of the segments passing through (x_0, y_0) and each neighboring edge point (x, y) are

$$\theta^{(x_0,y_0)}(x, y) = \arctan\left(\frac{y - y_0}{x - x_0}\right) \in [0°, 180°)\,. \tag{3.12}$$

Then, for each neighbor (x, y), we use the directional edge map $\boldsymbol{E}_{Q(\theta)}(x, y)$, with θ computed above, to update the histogram count and $Q(\theta)$ given by equation (3.1) and illustrated on the right of Fig. 3.2. For example, if $(x_0, y_0) = (0, 0)$ and $(x, y) = (0, 3)$, we have $\theta_{(0,0)}(0, 3) = 90°$, a vertical segment, and the directional edge point that contributes to the histogram is $\boldsymbol{E}_{90}(0, 3)$, since $\boldsymbol{K}_{90}$ is the kernel that best responds to the horizontal transitions that define vertical edges.

We define the histogram for point (x_0, y_0) as $\delta(x_0, y_0, \cdot)$, with N bins (given by (2.7)). Each edge point contributes to the two bins most corresponding to angle $\theta^{(x_0,y_0)}(x, y)$ with weights implementing linear interpolation. The signs in the directional edge maps are taken into account through positive or negative contributions to the histogram bins, effectively filtering out conflicting contributions due to noise, textures and image clutter. The neighborhood that is used for the local histograms has diameter M. A contextual edge is found in a particular pixel and direction by thresholding the magnitude of the histogram bins, (*e.g.*, a threshold of 50% of M), *i.e.*, if $|\delta(x, y, n)| \geq C$. Alg. 1 synthesizes the procedure for each point $\boldsymbol{p}_0 = (x_0, y_0)$.

Algorithm 1 Accumulating local responses in a histogram

```
1: input: $\boldsymbol{p}_0 = (x_0, y_0)$, edge maps $\boldsymbol{E}_\theta$, circular window radius $r = (M-1)/2$, bin
   number $N$
2: % initialize orientation histogram
3: $\delta(\cdot,\cdot,\cdot) = 0$
4: % for every point inside the circular ball of radius $r$ centered in $\boldsymbol{p}_0$
5: for $\boldsymbol{p} = (x, y) \in B(r, \boldsymbol{p}_0)$ do
6:     % compute the angle $\boldsymbol{p}$ makes with $\boldsymbol{p}_0$ (in degrees)
7:     $\theta_{(x_0,y_0)}(x, y) = \arctan(y - y_0/x - x_0)$
8:     % compute fractional bin to update
9:     $n = \theta_{(x_0,y_0)}(x, y)\, N/180°$
10:    % update integer bins using bilinear interpolation
11:    $\delta(x_0, y_0, \lfloor n \rfloor) = \delta(x_0, y_0, \lfloor n \rfloor) + (\lceil n \rceil - n)\, \boldsymbol{E}_{Q(\theta_{(x_0,y_0)}(x,y))}$
12:    $\delta(x_0, y_0, \lceil n \rceil) = \delta(x_0, y_0, \lceil n \rceil) + (n - \lfloor n \rfloor)\, \boldsymbol{E}_{Q(\theta_{(x_0,y_0)}(x,y))}$
13: output: $\delta$
```

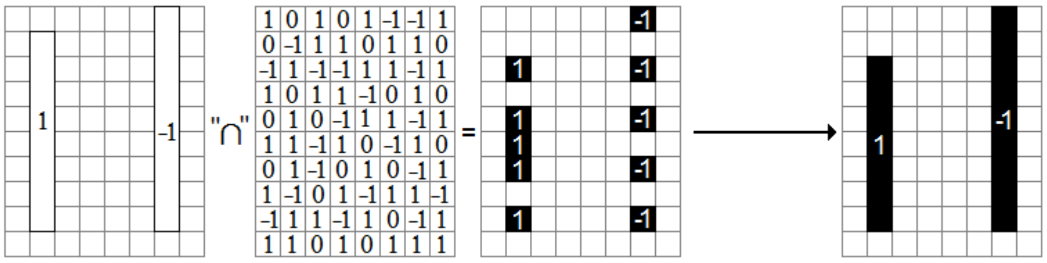
3.4 Combining contextual and local edges using connectivity

The method starts by finding contextual and local edges for every direction θ_n. Local edges are computed as in chapter 3.3, by computing image derivatives through the convolution of $\boldsymbol{I}$ with central difference operators and then obtaining their sign, using equation (3.11) above. Local orientation has been exploited before and captured by using several types of kernels, see, *e.g.*, [Burns 1986, Illingworth 1987b, Dahyot 2009, von Gioi 2010]. Although kernels with a large support would smooth the noise, the use of simple central difference operators enables more precise edge localization and angular responses, by minimizing the influence of surrounding pixels. Finally, the edge maps of both the local and contextual edges are combined, for each pixel and orientation: if local edge $\boldsymbol{E}_{Q(\theta_n)}(x, y)$ has the same sign as contextual edge $\delta(x, y, n)$ then we store the sign of the edges on the *combined edge map*,

$$\boldsymbol{C}_{\theta_n}(x, y) = \begin{cases} 1 & \text{if } \boldsymbol{\nabla}_{Q(\theta_n)}\boldsymbol{I}(x, y) \geq T \wedge \delta(x, y, n) \geq C \\ -1 & \text{if } \boldsymbol{\nabla}_{Q(\theta_n)}\boldsymbol{I}(x, y) \leq -T \wedge \delta(x, y, n) \leq -C \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.13}$$

If a valid edge is found at pixel $\boldsymbol{A}(x, y, \theta_n)$, *i.e.*, $|\boldsymbol{C}_{\theta_n}(\boldsymbol{A}(x, y, \theta_n))| = 1$, the next d pixels along direction θ_n (where d is the maximum distance threshold) are scanned until another valid edge with the same sign is found. If a valid edge was found, all these pixels are marked as edge points. If not, the valid edges are deemed disconnected and the pixels are not marked as edges.

Fig. 3.8 illustrates how the sign of a contextual edge is used to select coinciding local edges with the same sign, valid local edges. The arrow indicates the optional step of edge linking that can occur if the valid local edges are close to each other.



1	0	1	0	1	-1	-1	1
0	-1	1	1	0	1	1	0
-1	1	-1	-1	1	1	-1	1
1	0	1	1	-1	0	1	0
0	1	0	-1	1	1	-1	1
1	1	-1	1	0	-1	1	0
0	1	-1	0	1	0	-1	1
1	-1	0	1	-1	1	1	-1
-1	1	1	-1	1	0	-1	1
1	1	0	1	0	1	1	1

Figure 3.8: Using contextual edges to select valid local edges. From left to right: result of the contextual edge detector; result of the local edge detector; valid local edges; linked valid local edges, to form connected segments.

3.5 Experiments

The values for the contextual footprint size, M; contextual threshold, C; and local threshold, T, depend on the image content. Larger footprints enable better handling of noise and clutter but have the drawback that the edges on the image need to be aligned along curves with smaller curvatures for detections to occur; and, due to the narrower angular response of longer footprints, more filters are needed to span the entire range of angles, which leads to higher computational cost. In the remainder of this thesis, we use $M = 15$, which we found experimentally that provides a good compromise between noise filtering and computational complexity for the images that are used in this thesis, $C = 0.7$ and $T = 10$. The fact that all the experiments in this thesis use the same parameters (except where noted), despite the diversity of the input images, anticipates the robustness of the methods that we propose. By not requiring constant and careful tuning of the internal parameters to obtain good results for particular images, the proposed methods are suitable for real-life applications where user interaction is minimal.

In the absence of an established database for benchmarking the performance of edge detection methods, we single out demonstrative results of both our methods, contrasting them with the ones obtained with the standard Canny edge detector [Canny 1986] (in the implementation of MATLAB©) and the statistical edge detector of Bovik et al. [Bovik 1986]. The Canny edge detector is considered a state-of-the-art in edge detection method and the statistical edge detector of Bovik et al. [Bovik 1986] is the benchmark of statistical edge detectors. We first describe experiments with synthetic images to illustrate extreme cases that help to characterize the general behavior of our methods. Then, we present results obtained with several real world images that demonstrate their performance in practice.

3.5.1 Synthetic images

We use a synthetic image made up of intersecting line segments of multiple lengths and widths, shown on the top left of Fig. 3.9. We conclude that the Canny edge detector, with results shown on the top center of Fig. 3.9, succeeds in extracting clutter-free thin edges but is not able to extract thick edges. The statistical edge detector [Bovik 1986], is mostly able to extract the edges in the image, except for points in the center of the edge, where the distributions of the opposing sets of pixels are deemed equal. The bottom images of Fig. 3.9 show the results of our proposed edge detectors, which include most edges in the original image, of all lengths and widths. Because light-to-dark transitions are considered different from dark-to-light ones, the thin line segments in the image result in edges making up a pair of twin line segments next to the original segment. The thick line segments in the image are extracted, with only a few gaps where the transition type changes.

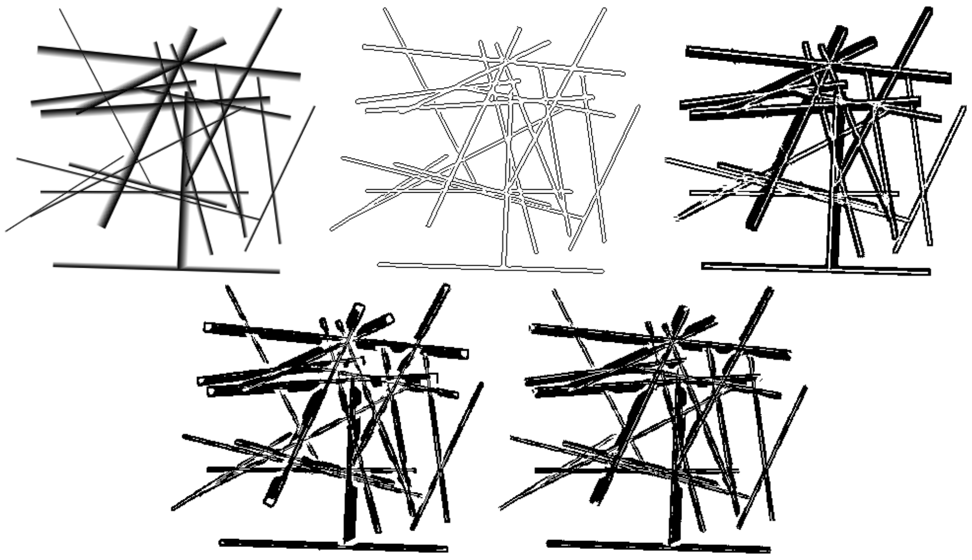


Figure 3.9: Image with crossing lines. Top left to right: image, Canny and statistical edge detectors. Bottom left to right: TV and sign test edge detectors.

We now illustrate the behavior of the algorithms when dealing with images whose line segments are characterized by being frontiers of differently textured regions. We use the synthetic images in the left column of Fig. 3.10, which were generated by adding variations of noise to a piecewise constant map. The central image is a special scenario in which the textured and textureless regions have the same mean value and their standard deviation is the only discriminating feature. The second column of Fig. 3.10 displays the results obtained by applying the Canny edge detector to the images on the left. Due to the high amplitude of noise and the reliance on local noise-prone derivative operators (*e.g.*, Sobel), many spurious edges are found. The third column of Fig. 3.10 displays the results of the statistical edge detector, which show that it does not originate many erroneous edges and is able to capture the actual transitions in the image. The fourth column of Fig. 3.10 displays the results of our TV parametric edge detector, which show that it also extracts few erroneous edges and captures the actual transitions in the image. In particular, it is the only method that can extract the boundaries in middle original image. Finally, the fifth column of Fig. 3.10 displays the results of our nonparametric sign test edge detector, whose performance is similar to the TV parametric method, except that it cannot deal with boundaries with the same mean value.

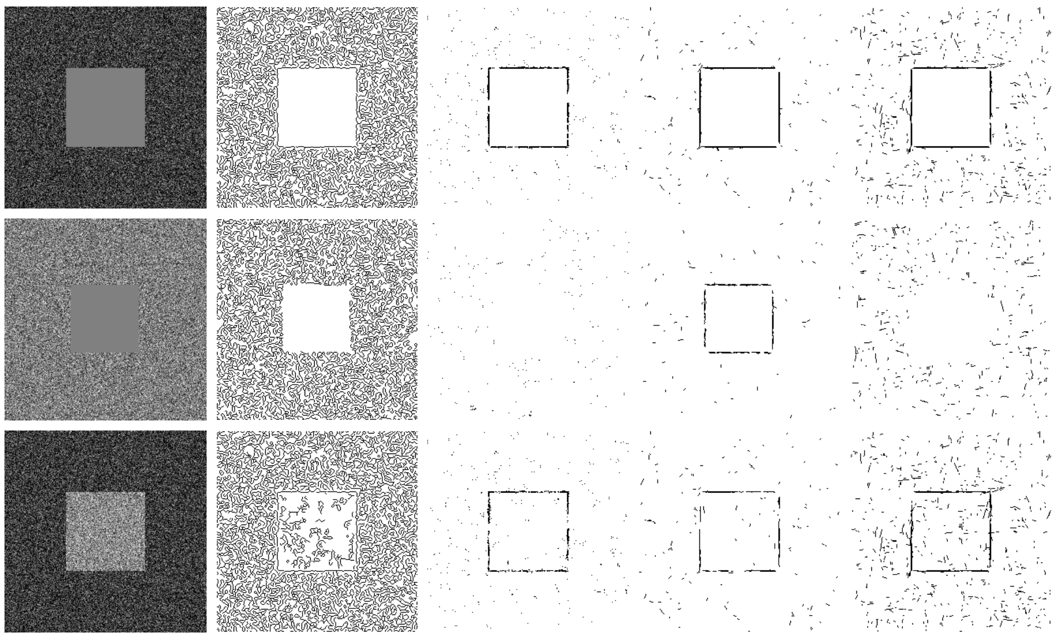


Figure 3.10: Textured images. From left to right: original image, result of the Canny [Canny 1986] and statistical edge detector [Bovik 1986], proposed TV and sign test edge detection methods.

3.5.2 Real images

We start by showing results obtained with the complex image in Section 1.1 to clarify the limitations of current edge detectors. This image is challenging due to its dense packing of line segments of multiple lengths. In the top right image of Fig. 3.11, we display the result of Canny edge detection [Canny 1986]. Although stronger edges are detected, the underlying crossing line segments and contours are not. The statistical edge detection [Bovik 1986], on the left of row two, is not able to capture details in the cluttered areas of the image, due to the shape of the footprints in its filter. The next two images of Fig. 3.11 display the results of our proposed TV and sign test edge detectors, where most edges are extracted successfully. To illustrate the good performance of our methods, the last three images of Fig. 3.11 show the edges extracted at individual angles, $\{16°, 90°, 172°\}$, using the TV test. These images illustrate that our methods extract entire connected sets of edges at each direction, thus preserving the underlying crossing line segments and contours.

Finally, Fig. 3.12, first shown in Section 1.1, contains many crossing line segments. The Canny edge detector fails to detect many of the edges due to the cluttered nature of the image, as mentioned in Section 1.1. The results of the statistical edge detector, shown on the left of row two of Fig. 3.12, show that many edges that are parallel and close to each other are not obtained, as in Fig. 3.11. On the last two images of Fig. 3.12, we display the results of our proposed TV and sign test edge detectors. The images show that most edges are extracted successfully.

3.6 Conclusion

We introduced two novel methods for edge detection, which combine contextual and local edges taking connectivity into account. Contextual edges are obtained with filters of large footprint, which are able to deal with noise; and local edges, thresholded image derivatives obtained with filters of small footprint, which have good localization. Furthermore, by taking connectivity into account the resulting edge maps result gap-free. We confirm experimentally that our edge detectors combine robustness to noise and good localization, as idealized by Canny [Canny 1986].

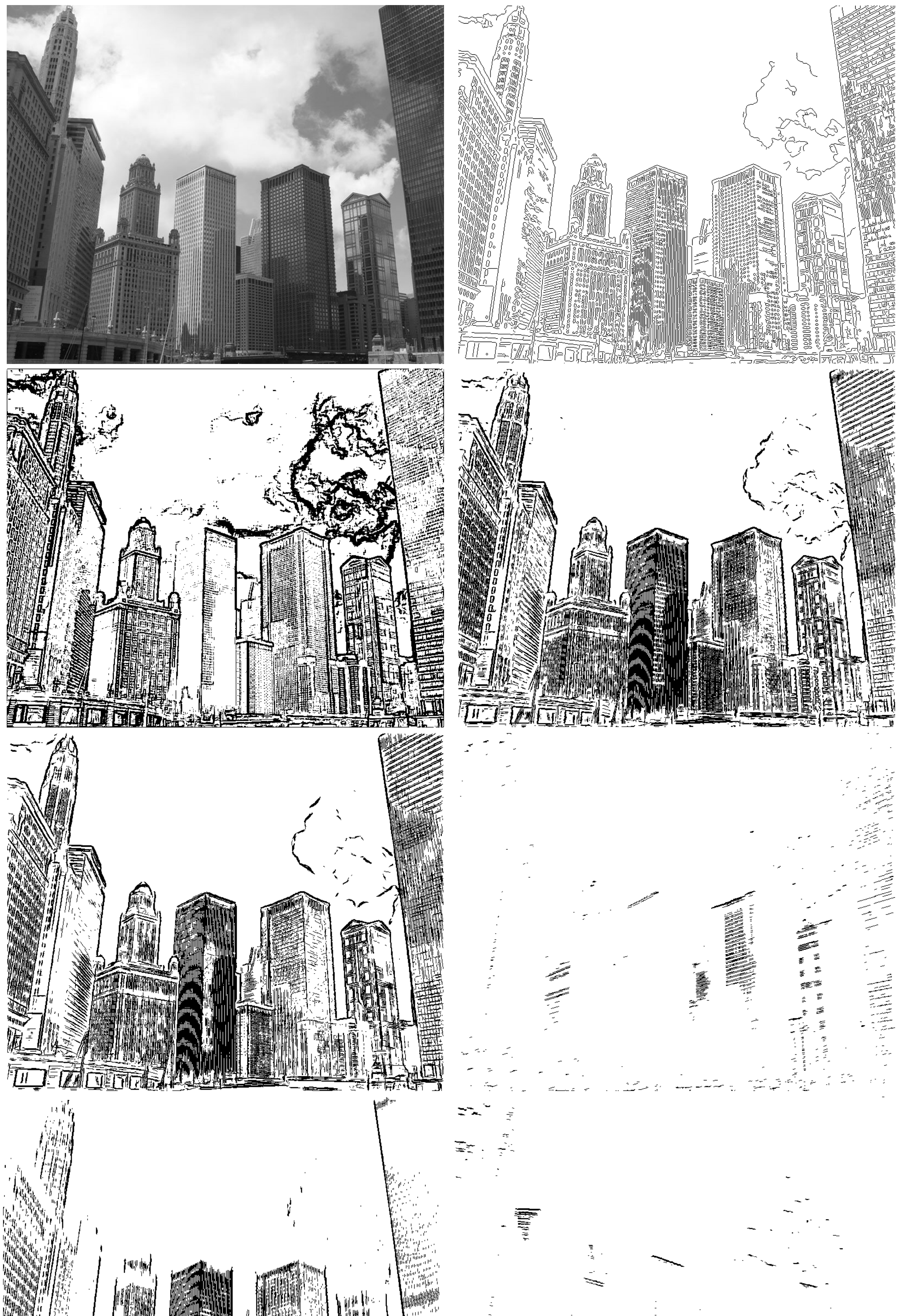


Figure 3.11: Left to right, top to bottom: image, Canny and statistical edge detection, TV and sign test, and edges along angles 16°, 90°, and 172° with TV test.

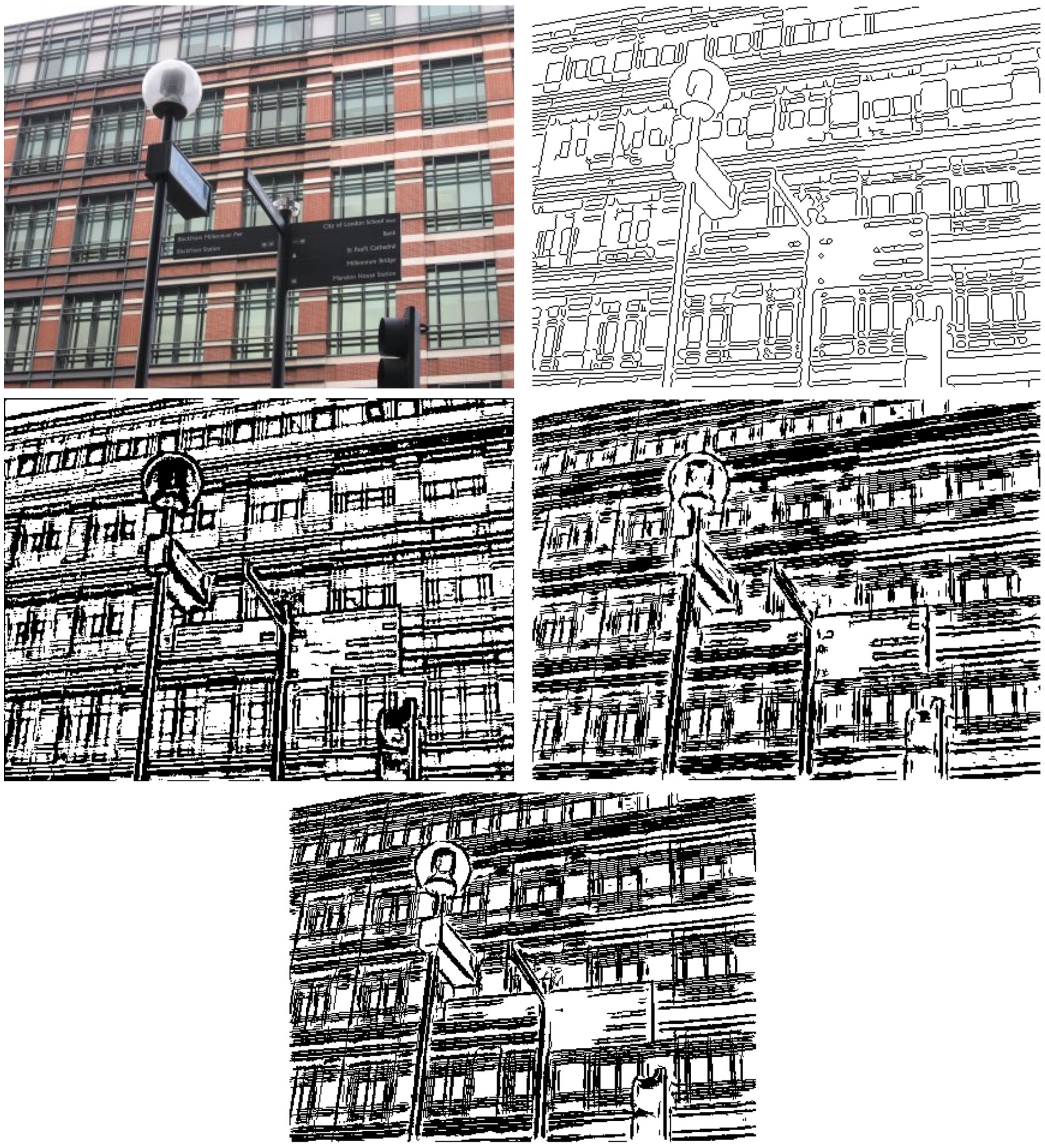


Figure 3.12: Left to right, top to bottom: image, Canny and statistical edge detection, TV and sign test.

Chapter 4
Connectivity-enforcing Hough Transform

4.1 Proposed approach

The key ingredient to the method presented in this chapter, denominated **STRAIGHT** (Segment exTRAction by connectivity-enforcInG Hough Transform) and presented in [Guerreiro 2012, Guerreiro 2011], is the incorporation of connectivity into the Hough Transform (HT) voting process. Connectivity is enforced in the HT by imposing that edge points only vote for lines in which they are spatially connected to other points. As a consequence, the vast majority of spurious votes are eliminated and peaks in the accumulator array become prominent and truly correspondent to line segments of maximum length. Simultaneously, the method integrates into the voting process the usually separate step of determining the extremes of the line segments. The block diagram of Fig. 4.1 summarizes the approach, which is outlined in the sequel.

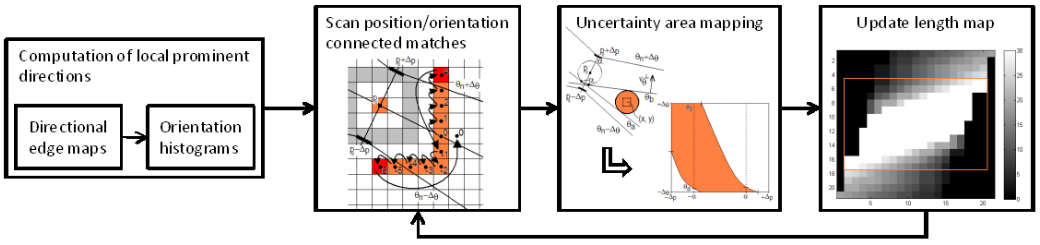

Computation of local prominent directions
Directional edge maps
Orientation histograms
Scan position/orientation connected matches
Uncertainty area mapping
Update length map

Figure 4.1: Block diagram of connectivity-enforcing Hough Transform.

STRAIGHT starts by computing the prominent directions at each edge point, which will guide the search for the orientations of line segments. An image line segment is characterized by a rectilinear alignment of dark-to-light (or opposite) transitions. The prominent direction detector (the paired nonparametric sign test detailed on section 3.3) computes, for each edge point, the set of directions according to which there is a predominance of those intensity transitions. As depicted in the first macro-block of Fig. 4.1, this is accomplished by taking into account directionally coherent edge points: in a first step, signed directional edge maps are computed; then, transition evidence is collected. This latter step can occur in two equivalent ways. In the first, a running average approach adds positive minus negative transitions along each orientation at each edge point, and large magnitude entries indicate prominent directions. In the second, orientation histograms are built at each edge point, by considering the neighboring edge points whose relative positions agree with the angle of the directional edge map. The histogram accumulates the signed values of the intensity transitions, thus the prominent directions at each edge point are detected by finding large magnitude entries in the corresponding histogram.

After computing the local prominent directions, STRAIGHT extracts line segments using the knowledge that the edge points forming each of them must be connected. This is done by computing new Local Hough Transform (LHT)-like maps (which we will call *length maps*) for each edge point, this time taking into account all other edge points whose position and directional content agree with potential line segments. Position matters because only points that respect connectivity are considered; directional content matters because only edge points with prominent direction that agrees with the candidate segment are considered. In practice, as illustrated in Fig. 4.1 for each prominent direction of each edge point, STRAIGHT progressively considers edge points further away until the connectivity criterion is violated. After exploring all candidate directions, the ones that collected more distant edge points correspond to the orientations of the longest connected line segments going through the starting point. Note that allowing a set of prominent directions, rather than a single one, enables dealing with crossing segments.

This proposed implementation of STRAIGHT incorporates the explicit mapping of uncertainty balls around the edge points into the Hough space (penultimate block of Fig. 4.1), increasing robustness and accuracy, and uses a hierarchical coarse-to-fine strategy to explore candidate directions, leading to a computationally tractable algorithm. Illustrative results of experiments that use synthetic and real images are presented to compare STRAIGHT with the standard HT [Duda 1972] and the state-of-the-art local method LSD [von Gioi 2010].

The organization of the remaining of this chapter is as follows. Section 4.2 describes the computation of the directional initialization. Section 4.3 details the extraction of connected line segments, by enforcing connectivity. The hierarchical implementation is described in Section 4.4. Experimental results are reported in Section 4.5 and Section 4.6 concludes the chapter.

4.2 Directional initialization

The directional initialization of STRAIGHT makes use of the paired nonparametric sign test that is detailed in section 3.3 but without enforcing connectivity, since this is done inside this method. To be used in STRAIGHT, the output is formated into array $\boldsymbol{\Theta}(\boldsymbol{p}_0)$, that contains, for each point, a list of directions for which a strong edge was found and its sign,

$$\boldsymbol{\Theta}(x_0, y_0) = \left\{ \left(\theta_0, \boldsymbol{\nabla}_{Q(\theta_0)} \boldsymbol{I}(x_0, y_0)\right), \left(\theta_1, \boldsymbol{\nabla}_{Q(\theta_1)} \boldsymbol{I}(x_0, y_0)\right), \ldots, \left(\theta_B, \boldsymbol{\nabla}_{Q(\theta_B)} \boldsymbol{I}(x_0, y_0)\right) \right\}, \tag{4.1}$$

where $0 \leq B \leq N$ is the number of histogram bins whose count is above the threshold. We denote by θ_n the central angle of the bin n and by $\boldsymbol{\nabla}_{Q(\theta_n)} \boldsymbol{I}(x_0, y_0)$

the image gradient, with orientation $Q(\theta_n)$ that best matches θ_n, according to (3.1). To account for the histogram discretization, *i.e.*, the nonzero width of the bins, we consider in the sequel as possible directions of line segments all the orientations $\theta \in [\theta_n - \Delta_\theta, \theta_n + \Delta_\theta]$, with $\Delta_\theta = 180/N/2 = 90/N$.

4.3 Extracting Connected Line Segments

We now describe the core of STRAIGHT, *i.e.*, the way we incorporate connectivity into the line segment extraction. We start by making explicit the parameter search problem that underlies the extraction of each of the segments, introducing the *length map*, which plays a similar role of the HT accumulator array. Then, we describe how edge points are sequentially mapped into the length map by taking into account both the edge point connectivity and the uncertainty due to discretization. Finally, we describe the procedure to extract line segments from the length map.

4.3.1 Line segment extraction as a parameter search problem

To accurately detect a line segment whose candidate location $\boldsymbol{p}_0 = (x_0, y_0)$ and orientation θ_n was roughly computed as described in the previous section, it is necessary to refine the estimates of both parameters simultaneously. This is done by accumulating information from other edge points lying close to the candidate line and having a candidate direction angle similar to θ_n. Our methods estimate the pair $(\hat{p}, \hat{\theta})$ that corresponds to the longest linear alignment of candidate matches.

We consider a line segment search range centered in the initial rough estimate $(\boldsymbol{p}_0, \theta_n)$, *i.e.*, the line segments that span the area depicted in Fig. 4.5. The line position is specified in terms of its distance δ_p to the edge point $\boldsymbol{p}_0$. The line orientation is represented by δ_θ, which represents the deviation with respect to the prominent direction angle θ_n in $\boldsymbol{\Theta}(\boldsymbol{p}_0)$. Thus, as illustrated in Fig. 4.2, any point $\boldsymbol{p} = (x, y)$ belonging to the line $(\delta_p, \delta_\theta)$ obeys

$$\langle \boldsymbol{p}, \boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta} \rangle = \langle \boldsymbol{p}_0, \boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta} \rangle + \delta_p \,, \tag{4.2}$$

where $\langle \cdot, \cdot \rangle$ is the inner product and $\boldsymbol{v}^{\perp}_{\theta} = (\sin(\theta), -\cos(\theta))$ is a unit vector with directional orthogonal to θ. The candidate line segment is allowed to deviate at most a predefined Δ_p from $\boldsymbol{p}_0$, *i.e.*, $\delta_p \in [-\Delta_p, \Delta_p]$, (in our experiments, we use $\Delta_p = 1$) and Δ_θ from θ_n, *i.e.*, $\delta_\theta \in [-\Delta_\theta, \Delta_\theta]$ (Δ_θ was defined in the previous section).

When the estimate $(\delta_p, \delta_\theta)$, coincides with the true value of the parameters defining a line segment in the image, there are several other edge points along the line $(\delta_p, \delta_\theta)$ for which the orientation $\theta = \theta_n + \delta_\theta$ is also prominent (with matching signs of the image gradient), according to the information collected in $\{\boldsymbol{\Theta}(\cdot)\}$. We call a *candidate match* to each of these edge points. Besides, due to the connectivity of the edge points forming a line, the gap between candidate matches must be smaller than a predefined *maximum distance threshold* d. Naturally, the closer the estimated line is to the actual one, more distant edge points of the true line segment

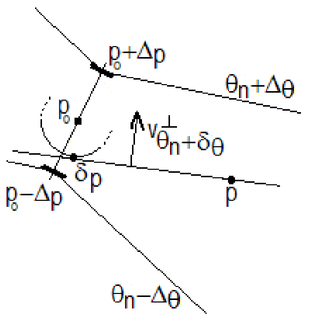


Figure 4.2: Edge point $\boldsymbol{p}_0$, prominent direction θ_n, line segment specified by parameters $(\delta_p, \delta_\theta)$, and range limits Δ_p and Δ_θ.

are captured. This is the key point of our approach, which formalizes the extraction of a line segment as the search for the parameters $(\delta_p, \delta_\theta)$ that *maximize the length* of the segment that can be extracted in the neighborhood of $\boldsymbol{p}_0$.

To extract the maximum length segments that pass close to each edge point, we borrow inspiration in the LHT, where local accumulator arrays are used in contrast to the single accumulator array of the HT, which cannot discriminate between distinct segments falling in the same line. In our case, we define local 2D *length maps* $\boldsymbol{L} : [-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta] \mapsto \mathbb{N}_0$. For each edge point, $\boldsymbol{L}(\delta_p, \delta_\theta)$ will contain the integer length of the line segment $(\delta_p, \delta_\theta)$. This map can be regarded as an extension of the accumulator array of a LHT in order to take line segment connectivity into account. In this scenario, the extraction of a line segment passing close to $\boldsymbol{p}_0$ consists of filling $\boldsymbol{L}(\cdot, \cdot)$, obtaining the parameters $(\delta_p, \delta_\theta)$ for which $\boldsymbol{L}(\delta_p, \delta_\theta)$ is maximum, and computing its start and end points.

To fill $\boldsymbol{L}(\cdot, \cdot)$ through the direct implementation of an exhaustive scanning of the space $[-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ would be computationally unbearable. In fact, the discretization of this space must be very fine to yield accurate results and, for each location, many edge points have to be processed (possibly, hundreds or thousands). Besides, this approach would use redundant computations, since each edge point would be visited several times because it may belong to several candidate line segments. For this reason, as usually done to fill the HT accumulator array, we also adopt a pixel-centered approach, where the edge points are used to fill the length map $\boldsymbol{L}(\cdot, \cdot)$ in an efficient way.

4.3.2 Incorporating uncertainty in the length map – the update region

We now show how each individual edge point is processed in our pixel-centered approach. Due to the pixel grid discretization, we model each edge point by an uncertainty ball, rather than a pointwise feature. We use the uncertainty ball radius $R = 1$, the maximum expected error in the location of each pixel. Because the uncertainty ball has a non-infinitesimal area, there is a set of parameters $\{(\delta_p, \delta_\theta)\}$, whose corresponding line segments cross it. This is illustrated in the left side of Fig. 4.3, where the lines formed by all orientations between θ_a and θ_b cross the uncertainty ball centered at (x, y), for position deviation $\delta_p = -\alpha$ from $\boldsymbol{p}_0$. We call *update region* of the length map domain to the set of positions and orientations $\{(\delta_p, \delta_\theta)\}$, whose corresponding line segments cross the uncertainty ball centered at each edge pixel. The update region for the pixel $\boldsymbol{p} = (x, y)$, in the length map of $\boldsymbol{p}_0$, is shown on the right side of Fig. 4.3.

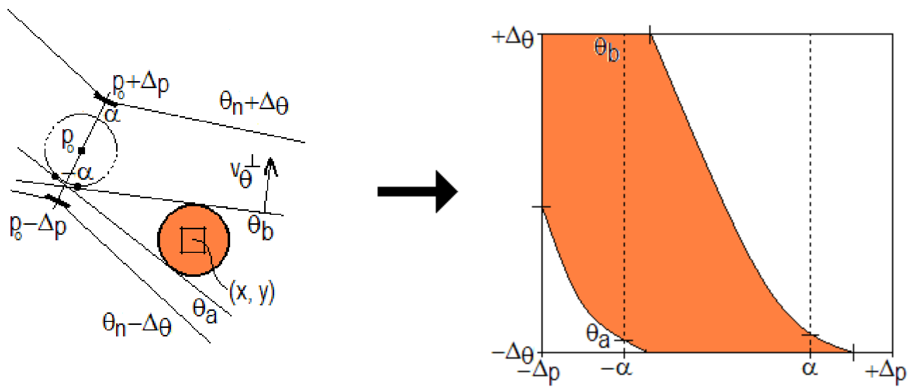


Figure 4.3: The uncertainty ball of an edge pixel (left) and the corresponding update region in the length map domain (right).

To find the analytic expressions for the bounds of update region, we use the line equation (4.2), now seen as a condition for line segments rather than points. When computing the length map for the edge point $\boldsymbol{p}_0$, we see from (4.2) that any segment $(\delta_p, \delta_\theta)$ that crosses the uncertainty ball of pixel $\boldsymbol{p} = (x, y)$ must verify

$$\langle \boldsymbol{p}, \boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta} \rangle = \langle \boldsymbol{p}_0, \boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta} \rangle + (\delta_p - r) \ , \tag{4.3}$$

where $-R \leq r \leq R$ is the distance between the center of the ball and the segment, along vector $\boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta}$ (depicted in Fig. 4.3). From (4.3), the line segment position δ_p is easily expressed in terms of its orientation δ_θ and r:

$$\delta_p = \langle \boldsymbol{p} - \boldsymbol{p}_0, \boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta} \rangle + r \, . \tag{4.4}$$

The update region, which we denote by $\boldsymbol{U}$, is thus the collection of intervals specified by all possible orientations δ_θ and corresponding positions δ_p given by (4.4), with $|r| \leq R$:

$$\boldsymbol{U} = \left\{ (\delta_p, \delta_\theta) : \delta_\theta \in [-\Delta_\theta, \Delta_\theta] \, , \delta_p \in \left[\delta_p^-(\delta_\theta), \delta_p^+(\delta_\theta) \right] \cap [-\Delta_p, \Delta_p] \right\} \ , \tag{4.5}$$

where the limits $\delta_p^-(\delta_\theta)$ and $\delta_p^+(\delta_\theta)$ are made explicit from (4.4) by expanding $\boldsymbol{v}^{\perp}_{\theta_n+\delta_\theta}$:

$$\delta_p^-(\delta_\theta) = (x - x_0) \sin(\theta_n + \delta_\theta) - (y - y_0) \cos(\theta_n + \delta_\theta) - R \, , \tag{4.6}$$

$$\delta_p^+(\delta_\theta) = (x - x_0)\sin(\theta_n + \delta_\theta) - (y - y_0)\cos(\theta_n + \delta_\theta) + R\,. \tag{4.7}$$

If the update region of a given pixel is non-empty, we say that the pixel is within the *search range*. In the illustration of Fig. 4.3, the search range for $\boldsymbol{p}_0$ is the one limited by the lines labelled with $\theta_n - \Delta_\theta$ and $\theta_n + \Delta_\theta$.

As geometrically evident, the range of angles of lines that cross an uncertainty ball decreases as the distance between $\boldsymbol{p}_0$ and $\boldsymbol{p}$ increases (an approximate expression for this range is $2\arcsin(R/\|\boldsymbol{p} - \boldsymbol{p}_0\|)$, obtained by noting that $\overline{\boldsymbol{p}_0, \boldsymbol{p}}$ can be approximated by the hypotenuse of a right-angled triangle of which R is the small cathetus). As a consequence, to enable the extraction of long line segments, *i.e.*, containing edge points $\boldsymbol{p}$ far from $\boldsymbol{p}_0$, the length map must be densely discretized to sample all relevant values of δ_θ. We propose an efficient way to deal with this need through the hierarchical coarse-to-fine procedure described in the Section 4.4.

We observe that expressions (4.6) and (4.7) are similar to the parameterizing equation of the HT [Duda 1972], $\rho = x\cos(\theta) + y\sin(\theta)$, with $\boldsymbol{p}_0 = (x_0, y_0)$ as the origin of the coordinate system and θ shifted by 90°. Thus, the boundaries of the update region resemble the sinusoidal shape of the bundles of votes of each edge point in the HT accumulator array (see Fig. 4.3 where, in fact, only a segment of that shape is seen, due to the length map limits). In what respects to the resolution of the accumulator array, when using the HT, the contradictory requirements of accuracy (high resolution) and coping with discretization error (low resolution, so that votes of the same line fall within the same bin) makes difficult, if not impossible, to achieve a good compromise. Strategies that uniformly blur the accumulation array (*e.g.*, by using multiple resolutions [Li 1986, Illingworth 1987a], or kernels of various sizes [Dahyot 2009]) do not change the scenario, since they still neglect the distinct influence of the discretization error of edge points located at different positions. In opposition, in our case, the resolution of the length map can be chosen arbitrarily large, since we model the actual discretization error of each individual edge point by using the correspondent (position-dependent) update region, as described above.

4.3.3 Sequential mapping of edge points to the length map

After describing how each pixel maps to a corresponding update region in the length map, we now show how to fill this map in a sequential way by processing all image edge points. This is illustrated in a simplified way in Fig. 4.4. Assuming that the starting point is the one in red, the pixels in the vicinity are scanned in search of an edge point. When an edge point is found, the set of line parameters that go through both the uncertainty ball of the original red pixel and the uncertainty ball of the edge point is computed, *i.e.*, the update region. Then, in a simplified way, the length map is updated in these parameter locations with the distance between the original red edge point and the current edge point. For the first edge point in the figure, with the value two. This step is repeated for all subsequent edge points while they are close to each other. Because this process occurred from the red point towards the right, the same procedure needs to occur between the red point and the edge points in the left, resulting in another length map. In the end, both length maps are added together and the parameter that corresponds to the longest line segment, *i.e.*, the parameter with the largest value in the length map, is obtained.

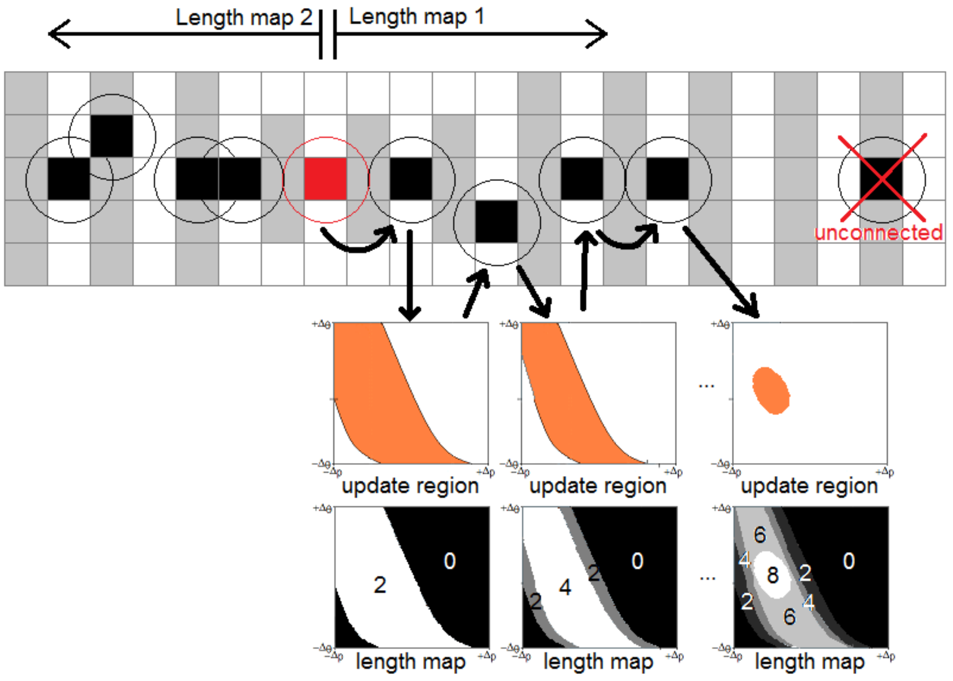


Figure 4.4: Illustration of how the length map is filled in (see details in the text).

In detail, let us consider, as before, the case the of the edge point $\boldsymbol{p}_0$, with prominent direction θ_n. When filling the corresponding length map $\boldsymbol{L}$, we consider the image divided in two half-planes by the line orthogonal to θ_n passing through $\boldsymbol{p}_0$. In each half-plane, starting from $\boldsymbol{p}_0$ we circularly scan the image, with progressively larger radius, mapping to the corresponding half-plane length map the candidate matches that fall within the search range and do not violate the connectivity requirement.

Due to the graceful adaptation to the discrete pixel grid, we use the so-called Manhattan distance to define the *equidistant curves* as the set of pixels $\boldsymbol{p}$ located at fixed distance e from $\boldsymbol{p}_0$ (*i.e.*, such that $\|\boldsymbol{p} - \boldsymbol{p}_0\|_\infty = e$). Fig. 4.5 illustrates the scenario, with the central edge point $\boldsymbol{p}_0$, the equidistant curves, labeled by the distance values, the search range and the half-planes.

For each half-plane, we scan each equidistant curve, starting with the one closer to $\boldsymbol{p}_0$ (*i.e.*, the curve with label $e = 1$ in Fig. 4.5), looking for candidate matches. Fig. 4.6 illustrates the scanning pattern for each equidistant curve. It starts in the center of the search range, *i.e.*, the pixel labeled with 0 in Fig. 4.6, and processes each pixel within the curve until reaching the limit of the search range (*i.e.*, the pixels labeled with positive values in Fig. 4.6, up to label 4, shown in red). Then, the pixels in the other direction are scanned, until the complete equidistant curve within the search range was processed (*i.e.*, the pixels labeled with negative values

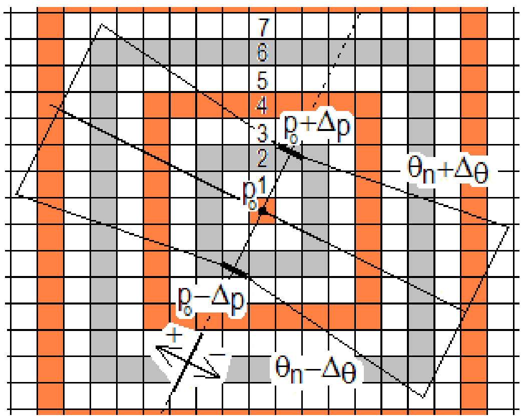


Figure 4.5: Central edge point $\boldsymbol{p}_0$, search range, and equidistant curves, each labeled by its integer distance to $\boldsymbol{p}_0$.

in Fig. 4.6, down to label −6, shown in red). Then, the following equidistant curve is processed.

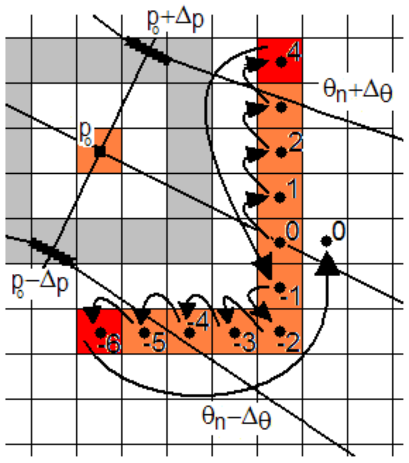


Figure 4.6: Illustration of the scanning pattern for a single equidistant curve in one of the half planes.

To account for the usage of the Manhattan distance, the discrete pixel $[\boldsymbol{p}]$ at the center of the search range for the equidistant curve e is given by

$$[p] = \text{round}\left(\boldsymbol{p}_0 \pm e\frac{\boldsymbol{v}_{\theta_n}}{\|\boldsymbol{v}_{\theta_n}\|_\infty}\right) ,$$

where $\boldsymbol{v}_\theta = (\cos(\theta), \sin(\theta))$ is a unit vector with angle θ, and the signal $\pm$ depends on the half-plane being considered.

When a candidate match is found in the equidistant curve e, its update region is computed, according to (4.5), (4.6), and (4.7), and the corresponding entries of the length map are updated. This updating consists simply in setting those entries to the value of the equidistant curve number, e. This indicates that there are valid line segments of (at least) size e with the parameters corresponding to those entries. To capture the connected nature of the line segments, we first prune the update region, eliminating the locations where the difference between the current value of the length map, $\boldsymbol{L}$, and the equidistance value e is larger than the maximum distance threshold d, *i.e.*,

$$\boldsymbol{U} \leftarrow \boldsymbol{U} \backslash \{[e - \boldsymbol{L}(\delta_p, \delta_\theta)] \geq d, \delta_p \in [-\Delta_p, \Delta_p], \delta_\theta \in [-\Delta_\theta, \Delta_\theta]\} \,. \tag{4.8}$$

Then, we update the length map according to

$$\boldsymbol{L} \leftarrow e\boldsymbol{U} + \boldsymbol{L} \odot \overline{\boldsymbol{U}} \,,$$

where $\boldsymbol{U}$ is seen as a binary mask and $\odot$ denotes the Hadamard, or elementwise, product. When the distance between e and all the values in the length map, $\boldsymbol{L}(\cdot,\cdot)$, is larger than d, *i.e.*, when $\boldsymbol{U} = \emptyset$, there are not updatable entries in the length map and the scanning stops for the corresponding half-plane. Naturally, parameter d controls the definition of connectivity. If $d = 1$, no line interruptions are allowed, thus the results are expected to show many broken lines; if d is very large, connectivity is relaxed and the results approach the ones of a standard HT. In our experiments, we used $d = 3$ pixels.

Alg. 2 synthesizes the procedure just described to compute each half-plane length map. The final length map for each edge point $\boldsymbol{p}_0$ and prominent direction θ_n is obtained by adding the two half-plane length maps.

Algorithm 2 Filling one half-plane length map $\boldsymbol{L}$ for edge point $\boldsymbol{p}_0$ and prominent direction θ_n.

```
1: input: $\boldsymbol{p}_0 = (x_0, y_0)$, $\theta_n$, prominent directions $\{\boldsymbol{\Theta}(\cdot)\}$, maximum distance $d$,
   uncertainty radius $R$
2: $\boldsymbol{L}(\cdot,\cdot) = 0$, $e = 1$
3: repeat
4:   for $[\boldsymbol{p}] = (x, y)$ in the equidistant curve $e$ (as illustrated in Fig. 4.6) do
5:     if $[\boldsymbol{p}]$ is a candidate match (according to $\boldsymbol{\Theta}(\boldsymbol{p})$) then
6:       $\boldsymbol{U} = \emptyset$
7:       for $\delta_\theta \in [-\Delta_\theta, \Delta_\theta]$ do
8:         $\delta_p^- = (x - x_0)\sin(\theta_n + \delta_\theta) - (y - y_0)\cos(\theta_n + \delta_\theta) - R$
9:         $\delta_p^+ = (x - x_0)\sin(\theta_n + \delta_\theta) - (y - y_0)\cos(\theta_n + \delta_\theta) + R$
10:        $\boldsymbol{U} \leftarrow \boldsymbol{U} \cup \left\{(\delta_p, \delta_\theta) : \delta_p \in \left[\delta_p^-, \delta_p^+\right] \cap [-\Delta_p, \Delta_p]\right\}$
11:      $\boldsymbol{U} \leftarrow \boldsymbol{U} \backslash \{e - \boldsymbol{L}(\cdot,\cdot) \geq d\}$ (requirement of line segment connectivity)
12:      $\boldsymbol{L} \leftarrow e\boldsymbol{U} + \boldsymbol{L} \odot \overline{\boldsymbol{U}}$
13:  $e \leftarrow e + 1$
14: until $e - \max\{\boldsymbol{L}(\cdot,\cdot)\} \geq d$
15: output: $\boldsymbol{L}$
```

4.3.4 Extracting line segments

As when detecting lines from the peaks of the HT accumulator array, we detect line segments passing through $\boldsymbol{p}_0$ with an orientation close to the prominent direction θ_n by simply collecting position-orientation pairs lying in the range $(\delta_p, \delta_\theta) \in [-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ that correspond to peaks in the corresponding length map $\boldsymbol{L}(\cdot, \cdot)$.

Whenever a line segment is detected, with position-orientation parameters $(\delta_p, \delta_\theta)$, we also obtain in a straightforward way the coordinates of its extremes:

$$[p_\pm] = \text{round}\left(\boldsymbol{p}_0 + E_\pm \frac{\boldsymbol{v}_{\theta_n+\delta_\theta}}{\|\boldsymbol{v}_{\theta_n+\delta_\theta}\|_\infty} + \delta_p \boldsymbol{v}^\perp_{\theta_n+\delta_\theta}\right), \tag{4.9}$$

where the subscript $\pm$ differentiates both extremes and $E_\pm$ denotes the maximum values of the length map of the corresponding half-planes.

To prevent multiple detections of a single line segment, each time a segment is detected for a central point $\boldsymbol{p}_0$ and prominent direction θ_n, we remove that prominent direction from all candidate matches $\boldsymbol{p}$ in the line segment. Multiple crossing segments are naturally extracted by collecting position-orientation pairs in all prominent directions $\{\theta_n, 1 \le n \le N\}$. To enable the detection of crossing segments with very close direction angles, a fine discretization of the angle histogram is required. Alternatively, we can use variable bin sizes for each prominent direction, in which case only (a small length interval around) the angle of an extracted line would be removed from all candidate matches $\boldsymbol{p}$ in the line segment. In the latter scenario, the corresponding length map may contain more than one peak, thus each one is dealt with independently.

A final remark regards avoiding that a few edge points of lines with position-orientation parameters outside the range $[-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ vote for spurious lines inside that range. If fact, this would happen whenever the uncertainty balls of those edge points intersect that range, as illustrated in Fig. 4.7. We explicitly detect these cases and ignore them by using an orientation limit slightly larger than Δ_θ (in all our experiments, we used a safe guard margin of $2°$) and only consider as detected segments those with estimated orientation within the original limits, *i.e.*, $\delta_\theta \in [-\Delta_\theta, \Delta_\theta]$.

Alg. 3 synthesizes the procedure to extract line segments, where the usage of *non-maxima suppression* in line 3 is not detailed, since it is similar to the standard procedure for extracting peaks from the accumulator array of the HT. The only difference is that, since STRAIGHT has detected all the candidate matches for each line segment, the parameters $(\delta_p, \delta_\theta)$ are more accurately estimated by fitting a line to the coordinates of the candidate matches with weights proportional to the magnitude of the image gradients. Naturally, other fitting criteria can easily be adopted in STRAIGHT.

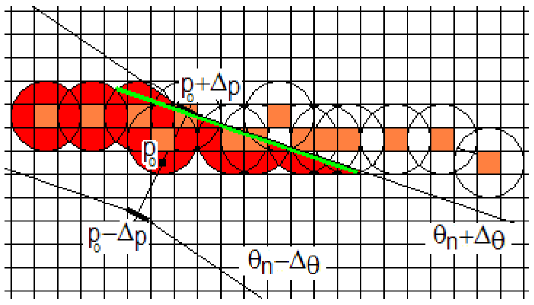


Figure 4.7: Edge points in a line outside the position-orientation range $[-\Delta_p, \Delta_p] \times [-\Delta_\theta, \Delta_\theta]$ and a spurious line segment, shown in green, that could be erroneously detected inside that range.

Algorithm 3 Extracting line segments passing through $\boldsymbol{p}_0$ with orientation close to θ_n.

```
1: input: $\boldsymbol{p}_0$, $\theta_n$, half-plane length maps $\boldsymbol{L}_+(\cdot,\cdot)$ and $\boldsymbol{L}_-(\cdot,\cdot)$, prominent directions $\{\boldsymbol{\Theta}(\cdot)\}$, angle range limit $\Delta_\theta$, uncertainty ball radius $R$
2: repeat
3:    $(\delta_p, \delta_\theta) = \arg\max [\boldsymbol{L}_+(\cdot,\cdot) + \boldsymbol{L}_-(\cdot,\cdot)]$ (find and remove peak using *non-maxima suppression*)
4:    if $|\delta_\theta| \leq \Delta_\theta$ then
5:       $[\boldsymbol{p}_+] = \text{round}\left(\boldsymbol{p}_0 + \boldsymbol{L}_+(\delta_p, \delta_\theta)\boldsymbol{v}_{\theta_n+\delta_\theta}/\|\boldsymbol{v}_{\theta_n+\delta_\theta}\|_\infty + \delta_p \boldsymbol{v}^\perp_{\theta_n+\delta_\theta}\right)$
6:       $[\boldsymbol{p}_-] = \text{round}\left(\boldsymbol{p}_0 + \boldsymbol{L}_-(\delta_p, \delta_\theta)\boldsymbol{v}_{\theta_n+\delta_\theta}/\|\boldsymbol{v}_{\theta_n+\delta_\theta}\|_\infty + \delta_p \boldsymbol{v}^\perp_{\theta_n+\delta_\theta}\right)$
7:       for the *candidate matches* $\boldsymbol{p}$ whose distance to $\overline{[\boldsymbol{p}_-][\boldsymbol{p}_+]}$ is smaller than (or equal to) $R$, remove from $\boldsymbol{\Theta}(\boldsymbol{p})$ the entry corresponding to $\theta_n$
8: until there are not prominent peaks in $[\boldsymbol{L}_+(\cdot,\cdot) + \boldsymbol{L}_-(\cdot,\cdot)]$
9: output: extremes of the extracted line segments, $\{[\boldsymbol{p}_-], [\boldsymbol{p}_+]\}$, and updated $\{\boldsymbol{\Theta}(\cdot)\}$
```

4.4 Hierarchical Implementation

Although the computational complexity of the pixel-centered approach described in the previous section is much smaller than an intensive approach, there are still some issues that need addressing. Because the discretization of $\boldsymbol{L}(\cdot,\cdot)$ must be fine, every time a new candidate match is found, a very large amount of positions in the length map need to be updated, which is a time-consuming operation. Furthermore, the number of pixels in the equidistant curves that fall within the search range and need to be scanned increases considerably with the size of the detected segment. Simultaneously, as the scanning of edge pixels proceeds and the corresponding updates are incorporated in the length map, the region of the map that remains updatable progressively becomes smaller. This occurs because more distant pixels correspond to smaller angle ranges, as explained in the previous section, and fewer $(\delta_p, \delta_\theta)$ positions still correspond to quasi-connected line segments. Since this narrowing of the updatable area was not taken into account in the previous section, most pixel checks are unnecessary and further computational cost optimizations are possible. This motivates the hierarchical implementation of STRAIGHT, as outlined in this section.

Our hierarchical approach progressively zooms in on the updatable regions, thus increasing its discretization density. The process starts with a length map that spans the initial wide location and angle ranges, as described in the previous section. Every time a set of equidistant curves are processed, the rectangular bounding box containing the updatable region (illustrated in the left image of Fig. 4.8) is upscaled, so that it takes up the complete length map (right image of Fig. 4.8). This way, although the length map has a constant size, it progressively addresses narrower location and angle ranges around $(\delta_p, \delta_\theta)$, effectively increasing the resolution of the estimates. Since the resolution can increase indefinitely, a coarse discretization of $\boldsymbol{L}(\cdot, \cdot)$ (we use an array of size 21×21) becomes sufficient to obtain long line segments and fewer pixels are tested, thus resulting in computationally efficient line segment extractions. Since, as described in the previous section, a length map may contain multiple disconnected updatable regions, corresponding to different line segments passing through $\boldsymbol{p}_0$, this process also divides the length map into multiple ones, each focused on a particular updatable region, and each estimation proceeds independently.

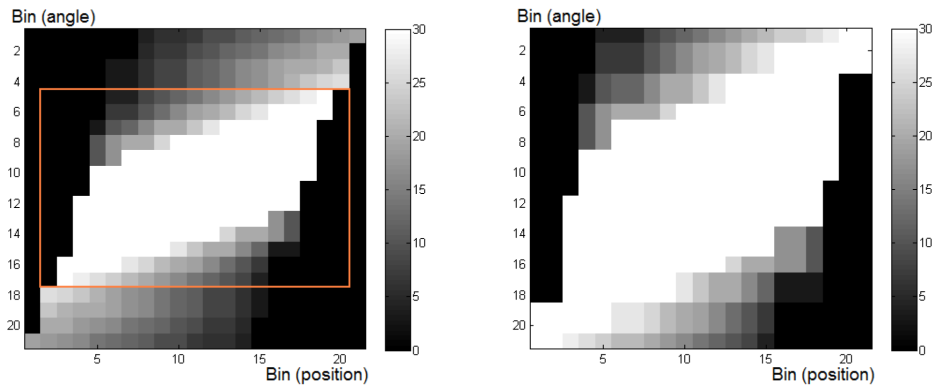


Figure 4.8: Illustration of the hierarchical implementation of STRAIGHT. Left: length map, with the bounding box of the updatable region. Right: the same region, after upscaling.

To implement the length map upscaling in the hierarchical STRAIGHT, we use the computationally simple Nearest Neighbor interpolation. An alternative to the discrete formulation underlying our hierarchical implementation is the continuous smooth kernel-based formulation of reference [Dahyot 2009].

4.5 Experiments

In the absence of an established database for benchmarking the performance of methods for line segment extraction, we single out demonstrative results of STRAIGHT, contrasting them with the ones obtained with the standard HT [Duda 1972] and the state-of-the-art LSD [von Gioi 2010] (the superiority of LSD when compared to several other methods is thoroughly demonstrated in [von Gioi 2010]). We first describe experiments with synthetic images to illustrate extreme cases that help to characterize the general behavior of STRAIGHT. Then, we present results obtained with several real world images that demonstrate its performance in practice. As detailed in section 3.5, we use parameters $M = 15$, $C = 0.7$ and $T = 10$, except where noted.

4.5.1 Synthetic images

We start by illustrating that STRAIGHT succeeds in cases tailored to the HT, *i.e.*, when processing images for which the HT exhibits clear superiority with respect to local methods. We use a synthetic binary image used in a review of several HT-based line segment extraction methods [Kälviäinen 1995]. In accordance with the conclusions of [Kälviäinen 1995], the HT succeeds in correctly extracting the lines from this image. In fact, although the multiple crossings make this image visually complex, the HT accumulator array exhibits the desired prominent peaks (see Fig. 2.6), capturing the fact that the lines are long and not in a very large number. The third and fourth images of Fig. 4.9 show the results of LSD and STRAIGHT, respectively. A pair of twin segments is extracted for each segment in the original image because these methods treat the binary image as any other, *i.e.*, as a grey-level one, and both light-to-dark and dark-to-light transitions are detected. In the third image of Fig. 4.9, we see that the local nature of the LSD limits its performance, particularly in resolving the line intersections, making it fail the extraction of several complete segments that cross each other. In contrast, the result of STRAIGHT, in the rightmost image, shows that it successfully extracts the majority of the line segments, regardless of the intersections.

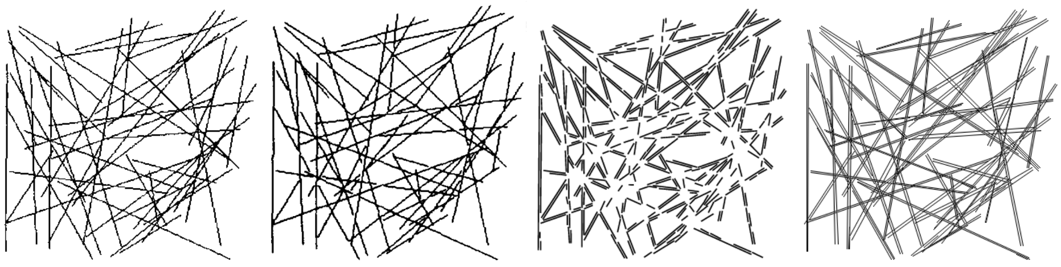


Figure 4.9: Clutterless image with prominent lines. From left to right: original binary image, result of the standard HT [Duda 1972], LSD [von Gioi 2010], and the proposed method STRAIGHT.

We now illustrate the behavior of the algorithms when dealing with the other extreme of the spectrum, *i.e.*, with images whose line segments are characterized by being frontiers of differently textured regions, rather than abrupt changes in a very smooth intensity level. We use the synthetic images in the left column of Fig. 4.10, which were generated by adding noise to a piecewise constant map. The top image simulates a scenario where a textureless objects occludes a textured one (*e.g.*, a wall in front of a tree) and the bottom one simulates two textured objects. The second column of Fig. 4.10 displays the results of the standard HT when its input is the edge map of the corresponding left image, obtained by using the Canny edge detector [Canny 1986]. We see that this procedure was not able to extract the boundary segments, originating a large number of false detections. This is due to the fact that the majority of the edge points detected by the Canny edge detector do not belong to the perceptually evident line segments that separate regions but rather to intensity transitions inside the textured region. Differently, LSD succeeds in interpreting the textures as not forming line segments but only captures parts of the real segments for the top image and almost none for the bottom one. This is due to the local nature of LSD, which makes it sensitive to the missing points in the line segments. The rightmost images of Fig. 4.10 display the results of STRAIGHT, showing that it succeeds in extracting the perceptually relevant lines as forming, in both cases, four complete line segments (the few short segments correspond to accidental connected alignments in the random texture).

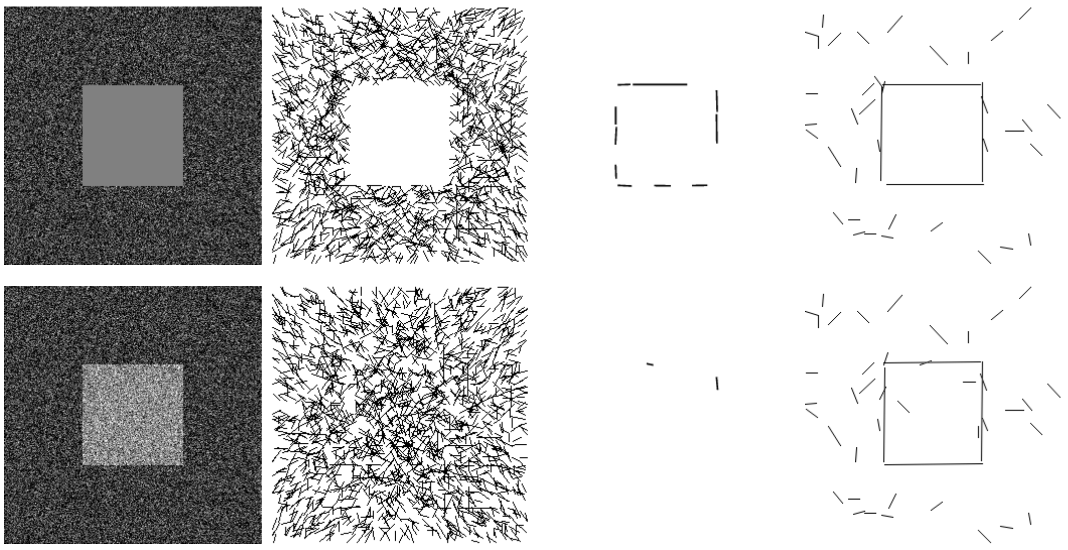


Figure 4.10: Textured images. From left to right: original image, Canny edge detector followed by standard HT [Duda 1972], LSD [von Gioi 2010], and STRAIGHT.

4.5.2 Real images

We start be showing the results obtained with the image used in Section 1.1 to clarify the limitations of the HT (Fig. 2.6). This image is challenging due to its dense packing of line segments of multiple lengths. In the top right image of Fig. 4.11, we display the results of LSD [von Gioi 2010], showing that a subset of the line segments are in fact detected. However, a closer look reveals that those are only the line segments that do not cross other structures and also that several longer segments are detected as fragmented ones. The results of STRAIGHT are in the two bottom images of Fig. 4.11. We see that our method succeeds in extracting the vast majority of the line segments in the image (exceptions are those which exhibit very low contrast). The fact that the extracted line segments are complete is particularly evident in the bottom right image, which displays only the line segments that have length greater than 50 pixels.

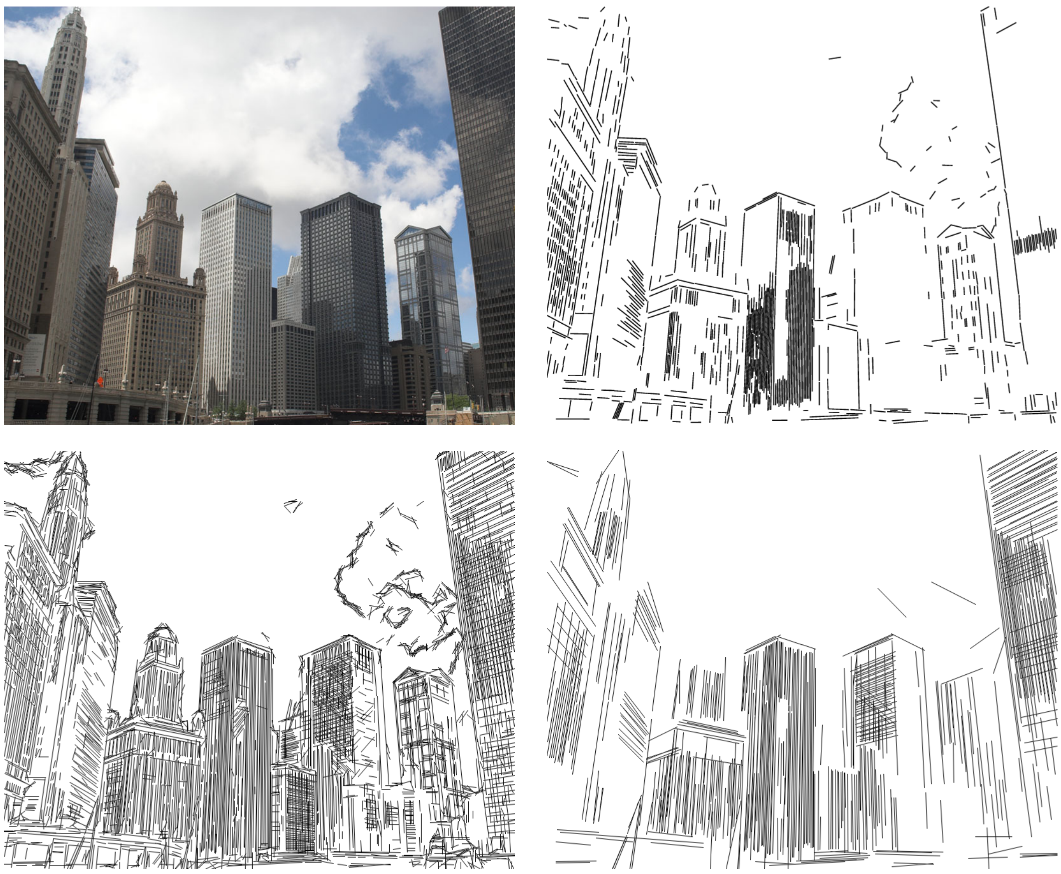


Figure 4.11: Top left: image. Top right: LSD [von Gioi 2010]. Bottom left: STRAIGHT. Bottom right: STRAIGHT (longer line segments).

To illustrate how the noise affects the extraction of line segments in real images, we report the results obtained with noisy versions of the same image. Fig. 4.12 synthesizes the results for two levels of zero-mean white Gaussian noise. We see that, with the increase of the noise level, LSD [von Gioi 2010] originates more segment fragmentations and a progressive failure to detect some line segments. The performance of STRAIGHT declines in a less steep way, as expected from its global nature.

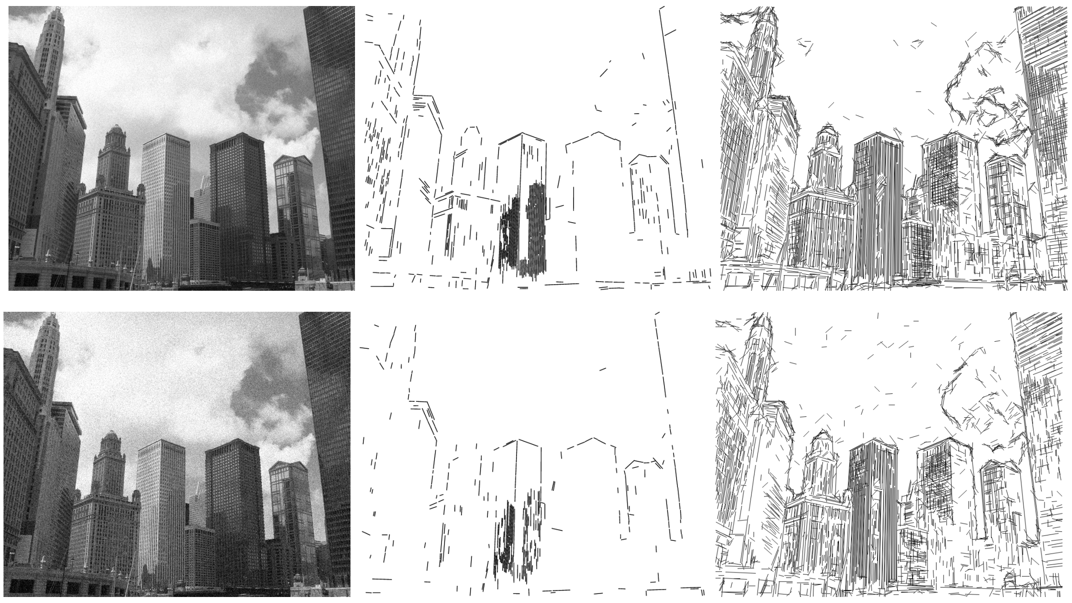


Figure 4.12: Left: noisy images ($\sigma = 10$ on the top and $\sigma = 20$ on the bottom). Center: LSD [von Gioi 2010]. Right: STRAIGHT.

Fig. 4.13 presents another illustrative case. It was obtained by processing an image containing a complex scene occluded by a net composed of very long line segments that cross multiple times. The result of LSD [von Gioi 2010] shows the net broken into short line segments (several sections of the net are not even extracted). On the other hand, our method was able to obtain almost all the complete line segments of the net, even in locations where the background is complex (exceptions are where the net has a very low contrast with respect to the background). The line segments extracted by our method that have length greater than 50 pixels, displayed in the bottom right image of Fig. 4.13, make this particularly evident.

Fig. 4.14 shows the results of processing an image containing mostly long line segments that cross multiple times. This image, obtained with a low-end camera, exhibits compression artifacts — it was obtained with a low-end camera and it is highly compressed (46 KB for 425×319 pixels). While LSD extracts the majority of lines as being artificially broken into very short line segments, our method was able to obtain almost all the complete segments. The ability of our method in capturing these long line segments is further illustrated by the large number of lines that have length greater than 50 pixels (shown in the bottom right image).

Finally, Fig. 4.15 presents results of using STRAIGHT with real images of various kinds. As desired, the vast majority of long line segments are extracted without artificial fragmentation, despite the multiple segment crossings. Also note that, although some of these images have edges that form curves, STRAIGHT succeeds in approximating these sections in a piecewise linear way, *i.e.*, by a sequence of rectilinear line segments.

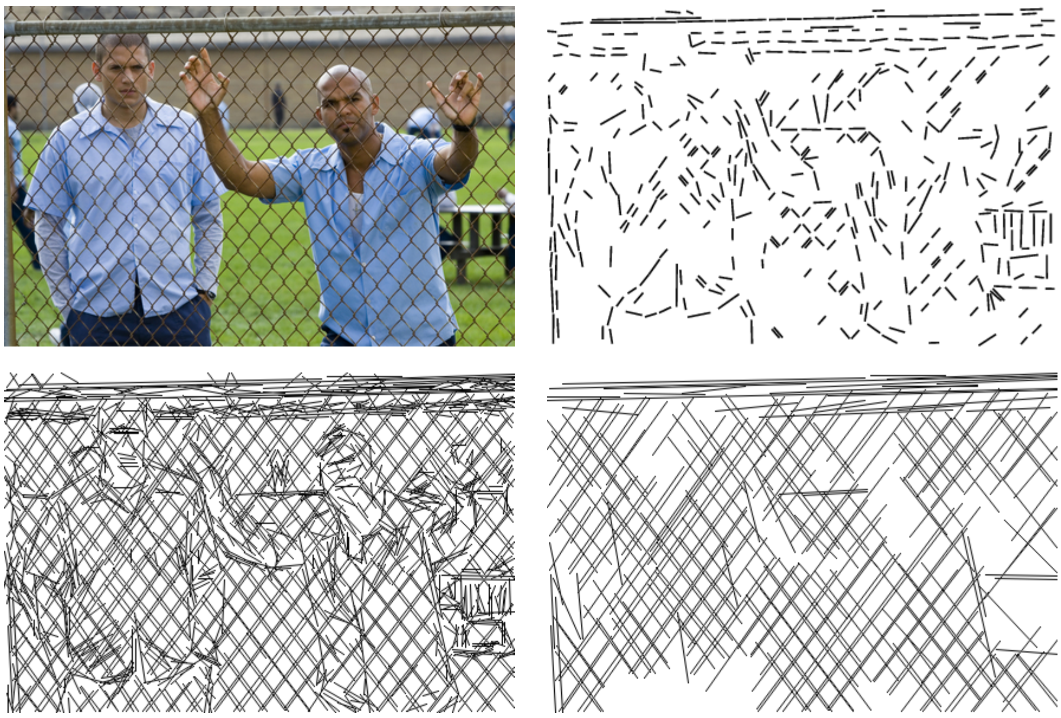


Figure 4.13: Top left: image (from “Prison Break", Fox Broadcasting Company©). Top right: LSD [von Gioi 2010]. Bottom left: STRAIGHT. Bottom right: STRAIGHT (longer line segments).

4.5.3 Computational complexity

The first step of STRAIGHT, *i.e.*, the computation of the local prominent directions (first macro-block of Fig. 4.1, described in Section 4.2) has a computational complexity that is fundamentally determined by the one of constructing the orientation histograms. In fact, the computation of the directional edge maps has a small cost that depends linearly on the number of pixels, determined by convolutions with small kernels, see expression (2.1). The construction of the local orientation histograms has in general a larger cost, due to the inherent counting process and the accounting for directional content. Naturally, this cost increases linearly with the number of detected edge points.

In what respects to the extraction of connected line segments, described in Section 4.3, each computation of the update region and the corresponding updating of the length map (last two blocks of Fig. 4.1) involve an approximately constant computational cost. Therefore, the computational burden of the process increases linearly with the number of edge points that need to be processed. Since spurious edge points that do not form line segments, *i.e.*, isolated edge points, originated, *e.g.*, by textures, are discarded when detecting local prominent directions, only the edge points lying in line segments need to be processed. The number of these edge points is thus given by the sum of the length of the line segments in the image.

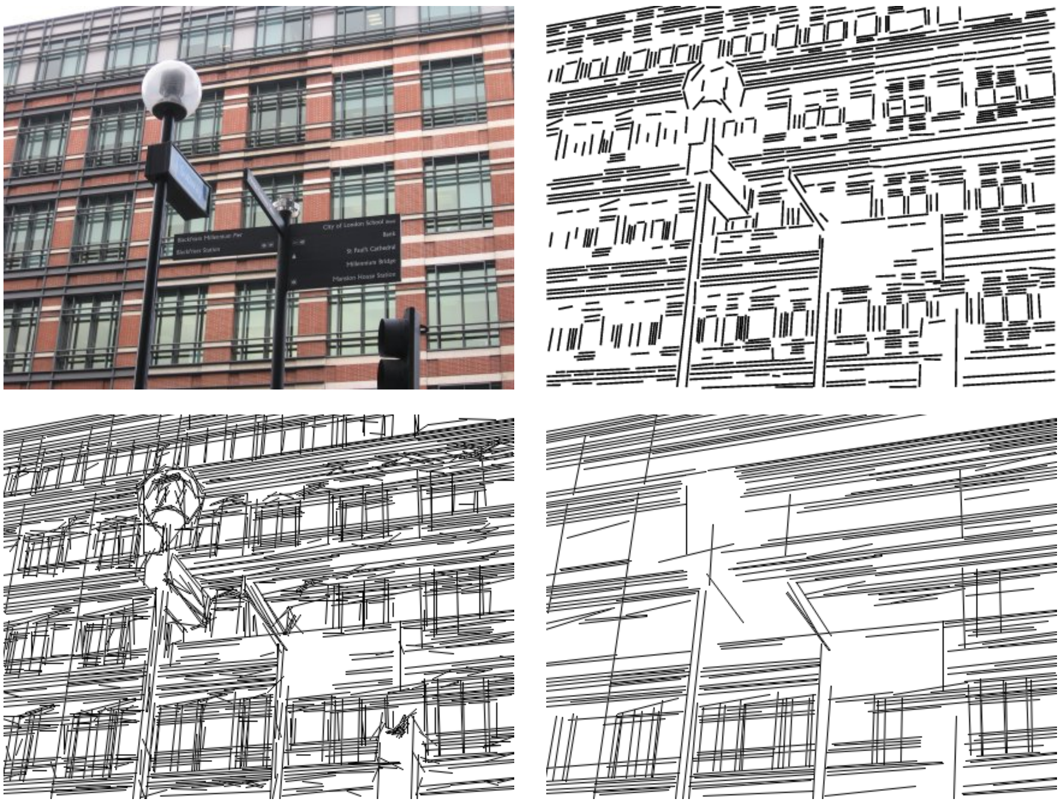


Figure 4.14: Top left: image. Top right: LSD [von Gioi 2010]. Bottom left: STRAIGHT. Bottom right: STRAIGHT (longer line segments).

Synthetically, we can say that the overall complexity of STRAIGHT increases linearly with the number of edge points, which also happens with the standard HT [Duda 1972]. The standard HT requires filling a single accumulator array, which basically depends linearly on the total number of edge points. According to reference [von Gioi 2010], the complexity of LSD increases linearly with the number of pixels of the image. In our experiments, we measured the running times of LSD as being consistently lower than one second (approximately 0.5 seconds[1] for the images in Figs. 4.9-4.14). The running times of STRAIGHT were measured as varying between approximately 100 and 1000 seconds. The standard HT required approximately 1 second to process the binary image in Fig. 4.9 and 10 seconds to process each of the edge maps of the textured images in Fig. 4.10. These numbers show that the approximately constant cost of processing each edge point with our implementation of STRAIGHT is significantly higher than the ones of the standard HT (and LSD). This is not surprising because: i) our code was not optimized; and ii) the good results obtained due to the enforcement of line segment connectivity naturally required more expensive computation.

Fig. 4.16 shows the computation time (in seconds) as a function of the image pixel

[1]All experiments were performed using standard C on an Intel© 2.67 GHz machine.

count, for a large set of images, in a standard C implementation and on an Intel© 2.67 GHz machine. Although the running time of STRAIGHT depends linearly on the number of edge points and not on the pixel count, we use the pixel count as a reference, since the latter is easier to measure. For visualization purposes, we consider that the number of edge points per pixel on the image is roughly a constant for the selected images, enabling us to indicate that STRAIGHT exhibits a computation time of about 6.6K pixels/second, for $N = 22$ directions, obtained experimentally.

In section 5.4, STRAIGHT is compared with the semi-local method for line segment extraction that is described in the following chapter.

4.6 Conclusion

We have presented a new method for line segment extraction, which we call STRAIGHT (Segment exTRAction by connectivity-enforcIng Hough Transform). Our method inherits the global accuracy of the HT and overcomes its limitations, particularly those that arise from not taking into account that line segments are *connected* sets of edge points. Our experiments show that STRAIGHT outperforms current methods for line segment extraction in challenging situations, *e.g.*, when dealing with complex images containing several crossing segments.

We end by pointing out that our approach may pave the way to other improvements in HT-like image edge analysis. In fact, as we saw, the standard HT is sensible to erroneous votes, which are eliminated by taking point connectivity into account. Thus, the detection of non-rectilinear shapes, *e.g.*, circles, in challenging scenarios, may also benefit from a similar treatment.

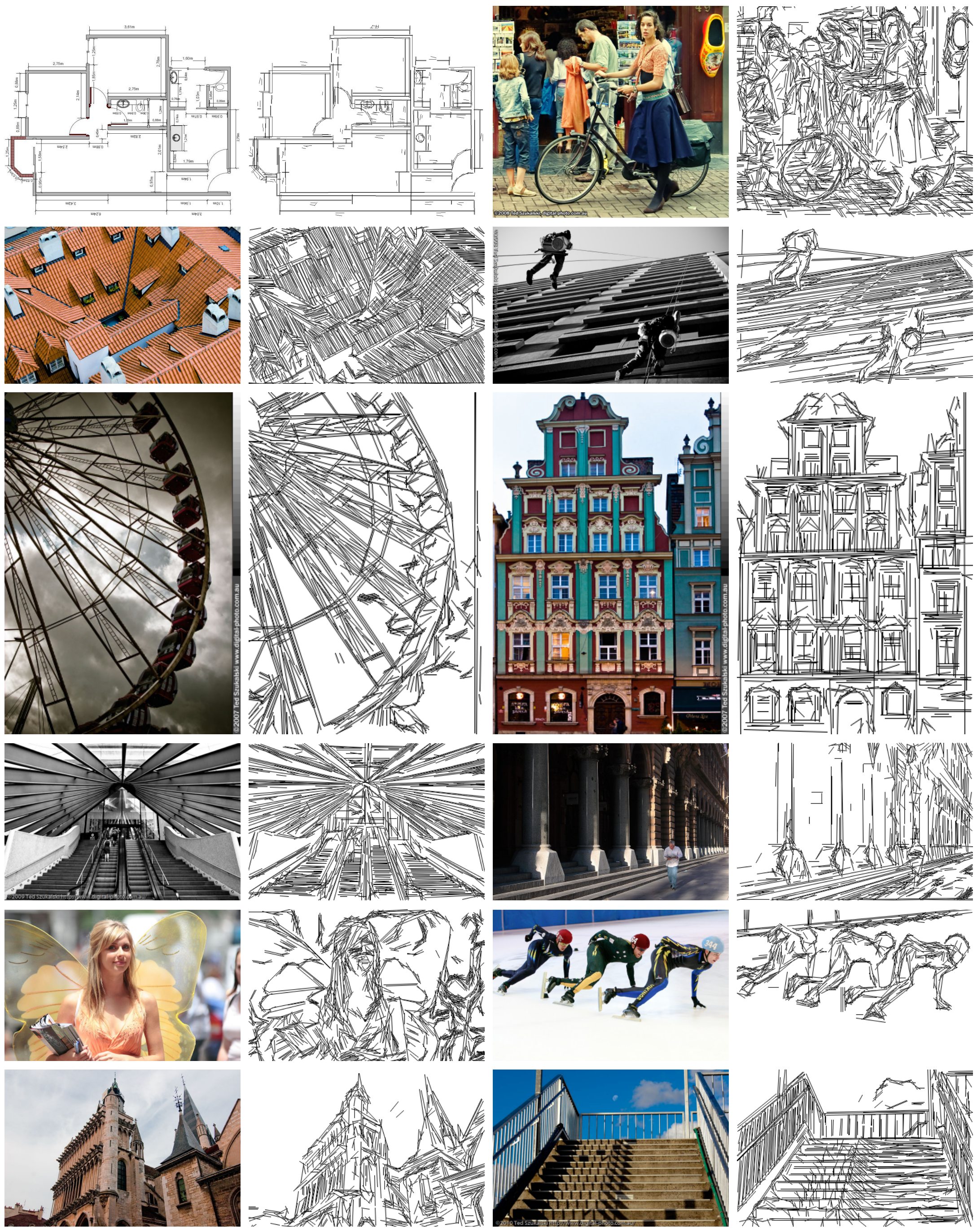


Figure 4.15: Results of STRAIGHT for several kinds of real images.

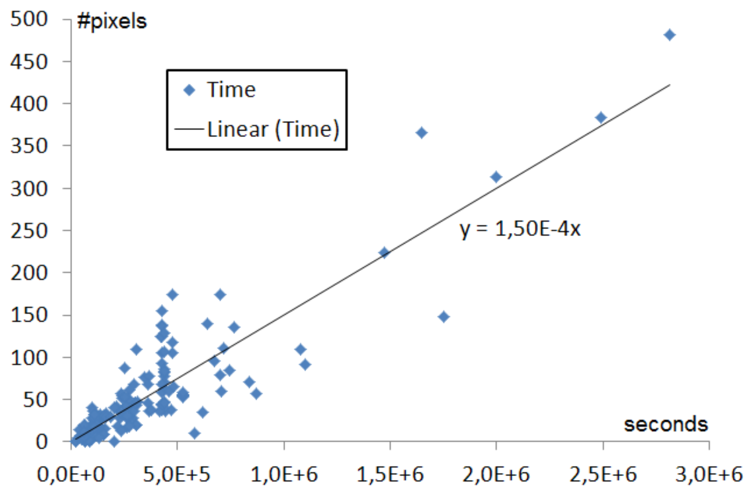


Figure 4.16: Computation time (in seconds) as a function of image pixel count, for a large set of images and in a standard C implementation. Roughly about 6.6K pixels per second are processed using $N = 22$ directions.

Chapter 5

Combining contextual and local edges for line segment extraction in cluttered images

5.1 Proposed approach

In the previous chapter, we propose a Hough Transform (HT)-based method that solves the main issues of the HT in dealing with complex images by enforcing connectivity in the voting process. In this chapter, we use the novel non-paired Total Variation parametric test edge detector proposed in section 3.2 that, by combining contextual and local edges and enforcing connectivity, outputs edge maps that do not exhibit gaps, despite noise and clutter in the images. For each orientation, the line segment extractor described in this chapter (and on paper [Guerreiro 2013a]) takes the output of the edge detector, finds connected edge regions through simple region growing, fits each region to a rectangle and then accepts or rejects each rectangle according to a fitness quality metric. Because this method uses the same ideas as local methods except that it uses our novel edge detector, which takes context into account, we refer to it as a semi-local method.

By avoiding expensive search mechanisms for overcoming gaps, as the costly scheme used in our previous HT-based method [Guerreiro 2012], our semi-local method results simple. Its simplicity enables its use in most practical applications and it obtains line segments of all lengths and widths. This is demonstrated in the experimental section, where we present illustrative results using synthetic and real images to compare this method with other methods: the standard HT [Duda 1972], the state-of-the-art of local methods LSD [von Gioi 2010], and our previous HT-based method [Guerreiro 2012].

The organization of the remaining of this chapter is as follows. In section 5.2.1 we provide a brief introduction to the region growing method we use and rectangle fitting is described in 5.2.2. The experimental results are reported in Section 5.3 and Section 5.5 concludes the thesis.

5.2 Combining contextual and local edges for line segment extraction in cluttered images

5.2.1 Region Growing

We obtain edges using the parametric non-paired two-sample Total Variation statistical edge detector detailed in section 3.2. This edge map $\boldsymbol{C}_{\theta_n}(x, y)$ is computed by combining contextual and local edges taking connectivity into account. Since the edge maps are already connected along lines along angles θ_n, we use a region growing method to join the various thin edge point segments into a single area and attribute a unique identification number to each. This is illustrated in Fig. 5.1. Region growing works by assigning a label to each set of edge points that have the same sign and are connected with each other, using an 8–neighbourhood.

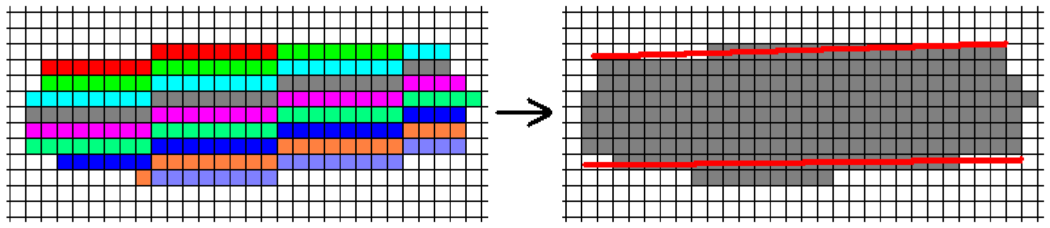


Figure 5.1: The multiple connected edges along lines (left), obtained in Section 3.2, are joined into a single area and lines are fit to the upper and lower limits (right).

5.2.2 Rectangle fitting

Rectangle fitting can occur in various ways (see [von Gioi 2010] for brief summary). In this method, we start by fitting a line to the upper and lower limits of each area, as illustrated in Fig. 5.1. Then, using the average angle of both fitted lines, $\theta_R = (\theta_{upper} + \theta_{lower})/2$, we obtain the start and end of each line segment.

To make sure that only rectilinear structures are detected, we validate each segment. In this thesis we require only that the lines that are fitted to the top and bottom of the connect area have similar angles, $\theta_{upper} \sim \theta_{lower}$, and the average angle should lie inside the permitted range, $\theta_R \in [\theta_n - 180°/2N, \theta_n + 180°/2N]$.

5.3 Experiments

We single out demonstrative results of this semi-local method, which we contrast with the ones obtained with the standard HT [Duda 1972], the state-of-the-art of local methods LSD [von Gioi 2010] (the superiority of LSD when compared to several other local methods is thoroughly demonstrated in [von Gioi 2010]), and our previous HT-based method, STRAIGHT [Guerreiro 2012]. We describe experiments with synthetic images, which help characterize the general behavior of this method. Then, we present results obtained with several real world images, which demonstrate its performance in practical application. Finally, we discuss the computational complexity of these methods. As detailed in section 3.5, we use parameters $M = 15$, $C = 0.7$ and $T = 10$, except where noted.

5.3.1 Synthetic images

We start by illustrating the behavior of the algorithms when dealing with an image made up of intersecting line segments of multiple lengths and widths, shown on the top left of Fig. 5.2. By comparing the edges computed by the Canny edge detector [Canny 1986] with the line segments that the HT extracts from them, on the top middle and right of Fig. 5.2, respectively, we conclude that the HT succeeds in correctly extracting the lines from this image. This occurs because line segments are long, not in a large number, and the HT does not require connectivity, therefore being able to overcome the multiple line crossings. On the other, the results of the LSD method, shown in the bottom left image of Fig. 5.2, illustrate that local methods fail to overcome line crossings and splits them. This occurs because local methods require absolute connectivity, *i.e.*, that edge points are perfectly chained together. In the particular case of the LSD, the state-of-the-art of local methods, edge points must have approximately constant direction. Although the results of STRAIGHT show that it is able to extract thin line segments regardless of the intersections, it is unable to deal with thick ones, originating multiple erroneous detections. Our semi-local method succeeds in extracting line segments of all lengths and widths, with few errors. A pair of twin segments is extracted for each segment in the original image because both light-to-dark and oposite transitions are detected.

We now illustrate the behavior of the algorithms in capturing transitions between differently textured regions. This simulates low signal-to-noise scenarios that occur when using very low thresholds in edge detection, for increased sensibility, where real transitions should be extracted successfully, while avoiding false ones. We use the synthetic images in the left column of Fig. 5.3, which were generated by adding noise to a piecewise constant map. The top image represents a simpler scenario, where one of the areas involved in the transition is perfectly smooth. The central image represents the same scenario, except that the mean value of both regions is now equal, making the variance the single discriminating factor. The bottom image simulates two smooth objects in a low signal-to-noise image. The second column of Fig. 5.3 displays the results of applying the HT to the images on the left. The Canny edge detector [Canny 1986] computes an edge map, which is the input to the HT. The high amplitude of noise and the reliance on local noise-prone derivative operators (*e.g.*, Sobel) by the Canny edge detector originates spurious edge points that prevent the HT from extracting the real boundary segments. This originates a large number of false detections and illustrates the lack of robustness in typical global methods. Due to the effect of noise, the local edge detection in LSD can not produce edges with constant direction along real segments and the LSD fails to extract line segments, except for parts of the top image. The results of STRAIGHT and the proposed semi-local method, in the two rightmost columns of

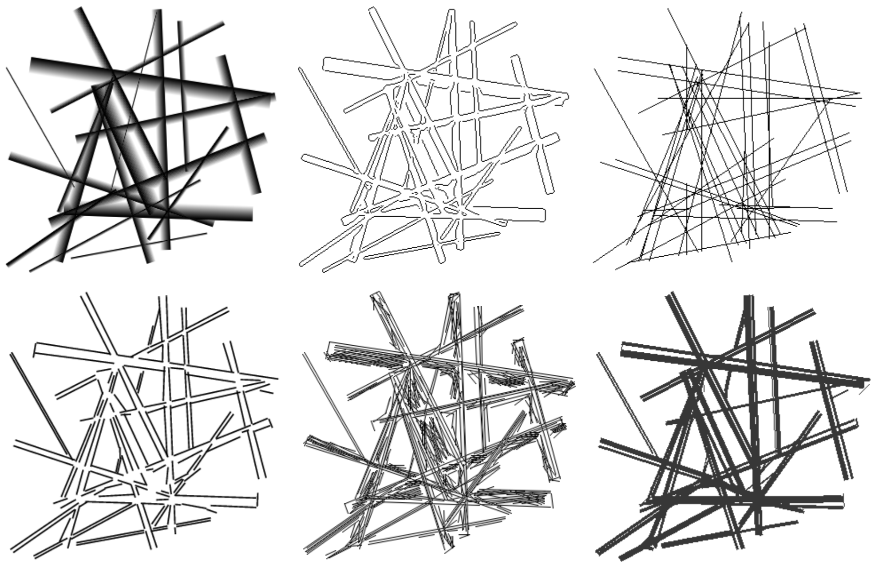


Figure 5.2: Image with prominent lines. Top left to right: original image, result of Canny edge detector [Canny 1986], and standard HT [Duda 1972]. Bottom left to right: LSD [von Gioi 2010], STRAIGHT [Guerreiro 2012], and the proposed semi-local method.

Fig. 5.3, show that both overcome noise and succeed in extracting the line segments for the top and bottom images (the few short segments correspond to accidental connected alignments in the random texture). By allowing samples with the same mean but different variances to be classified as edges, by using two-sample tests and, in particular, the TV distance, the proposed semi-local method is the only one that succeeds in obtaining most of the real segments of the figure in the middle row.

5.3.2 Real images

We start by showing a challenging image that was first used in [Guerreiro 2012] to demonstrate the ability of STRAIGHT in dealing with the dense packing of line segments of multiple lengths that cross each other. In the top right image of Fig. 5.4, we display the results of the HT [Duda 1972], showing that extraction fails altogether for not being able to cope with the large number of edge points (this is explained in detail in [Guerreiro 2012]). On the middle left image of Fig. 5.4, we display the results of LSD [von Gioi 2010], showing that a subset of the line segments are in fact detected. A closer look reveals that those are only the line segments that do not cross other structures and also that several longer segments are detected as fragmented ones. The results of STRAIGHT are shown in the middle right image of Fig. 5.4 and we see that it succeeds in extracting the vast majority of the line segments in the image (exceptions are those which exhibit very low contrast). Similar good results are obtained by the semi-local method we propose, shown in the bottom images of Fig. 5.4, with the difference that STRAIGHT needed about 56 seconds to process

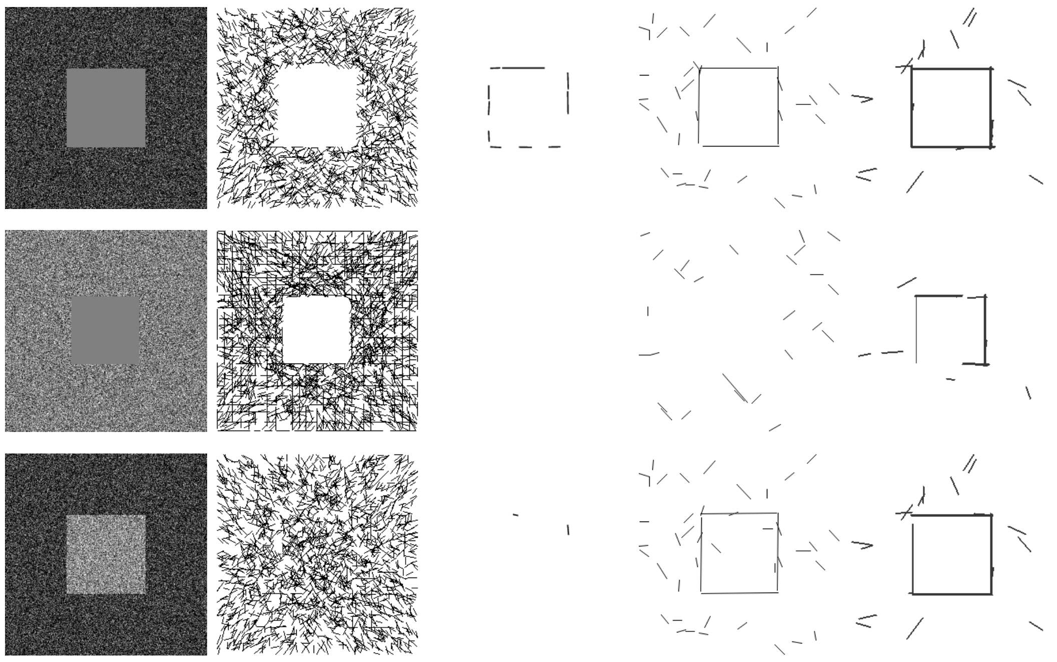


Figure 5.3: Textured images. From left to right: original image, result of the standard HT [Duda 1972] (preceded by the Canny edge detector), LSD [von Gioi 2010], STRAIGHT [Guerreiro 2012] and the proposed semi-local method.

this image while the semi-local method needed only about 7 seconds on the same machine. The bottom right image displays only the line segments that have length greater than 50 pixels, illustrating that line segments are not artificially broken in pieces.

Fig. 5.5 presents another illustrative case. It was obtained by processing an image containing a complex scene of line segments (many of which of low contrast) occluded by a net that is large and out-of-focus. The result of LSD [von Gioi 2010] shows that most low contrast segments were not extracted and that others are fragmented in multiple pieces. The fragmentation of line segments is improved in STRAIGHT [Guerreiro 2012] but low contrast segments are equally not extracted and the thick lines of the net originate a multitude of erroneous line segments. On the other hand, this semi-local method extracts most line segments, including low contrast and thick ones, with little fragmentation. The extraction of low contrast line segments is enabled by the better handling of noise of two-sample statistical tests.

Finally, Fig. 5.6 presents results of using the proposed semi-local method with real images of various kinds. As desired, the vast majority of long line segments are extracted without artificial fragmentation, despite the multiple segment crossings. Also note that, although some of these images have edges that form curves, our method succeeds in approximating these sections in a piecewise linear way, *i.e.*, by a sequence of rectilinear line segments.

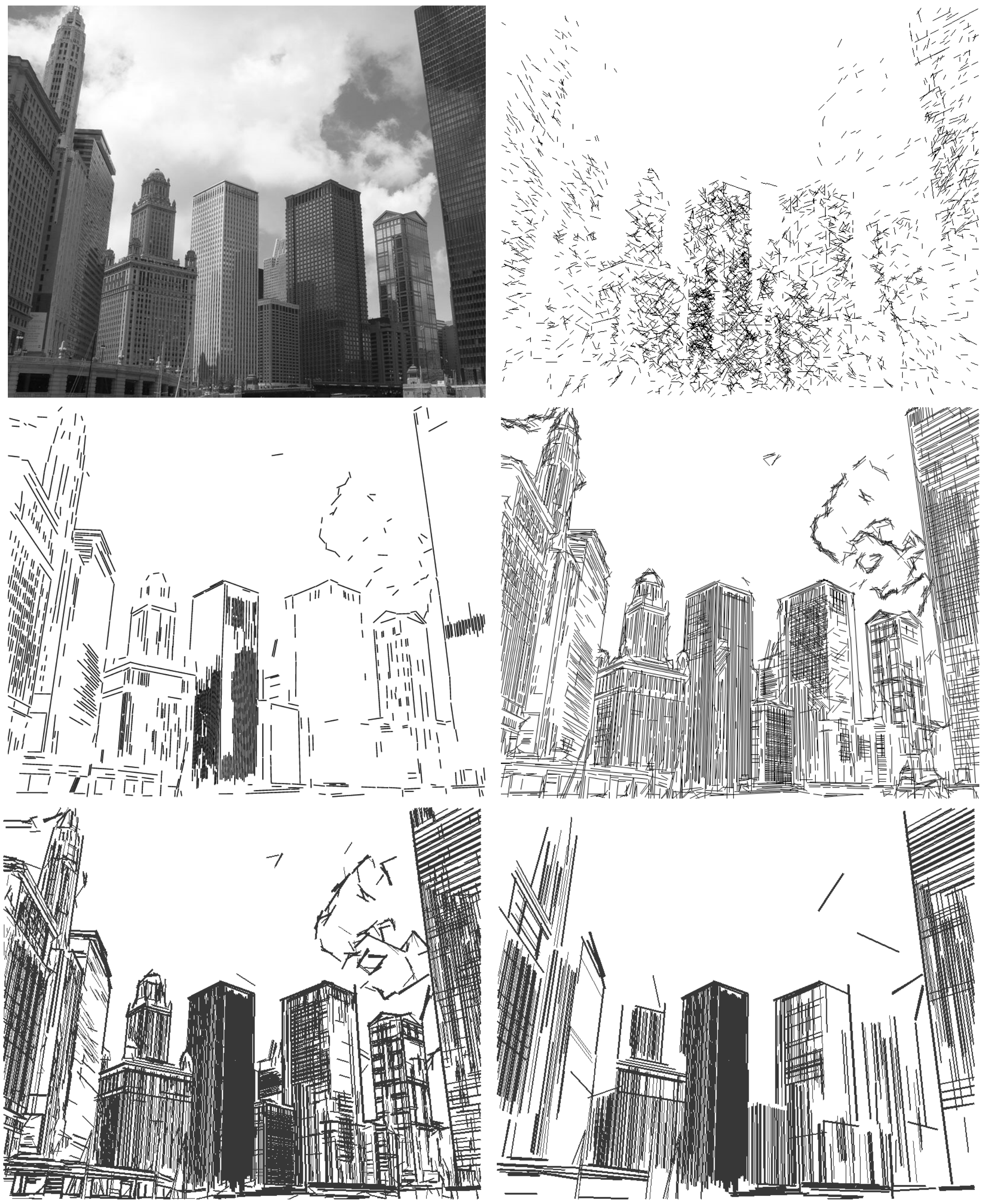


Figure 5.4: Top left: image. Top right: HT [Duda 1972]. Middle left: LSD [von Gioi 2010]. Middle right: STRAIGHT [Guerreiro 2012]. Bottom left: proposed semi-local method. Bottom right: proposed semi-local method (longer segments).

5.3.3 Computational complexity

The most computationally intensive portion of this method is the calculation of contextual and local edges and their combination into continuous edge maps. By

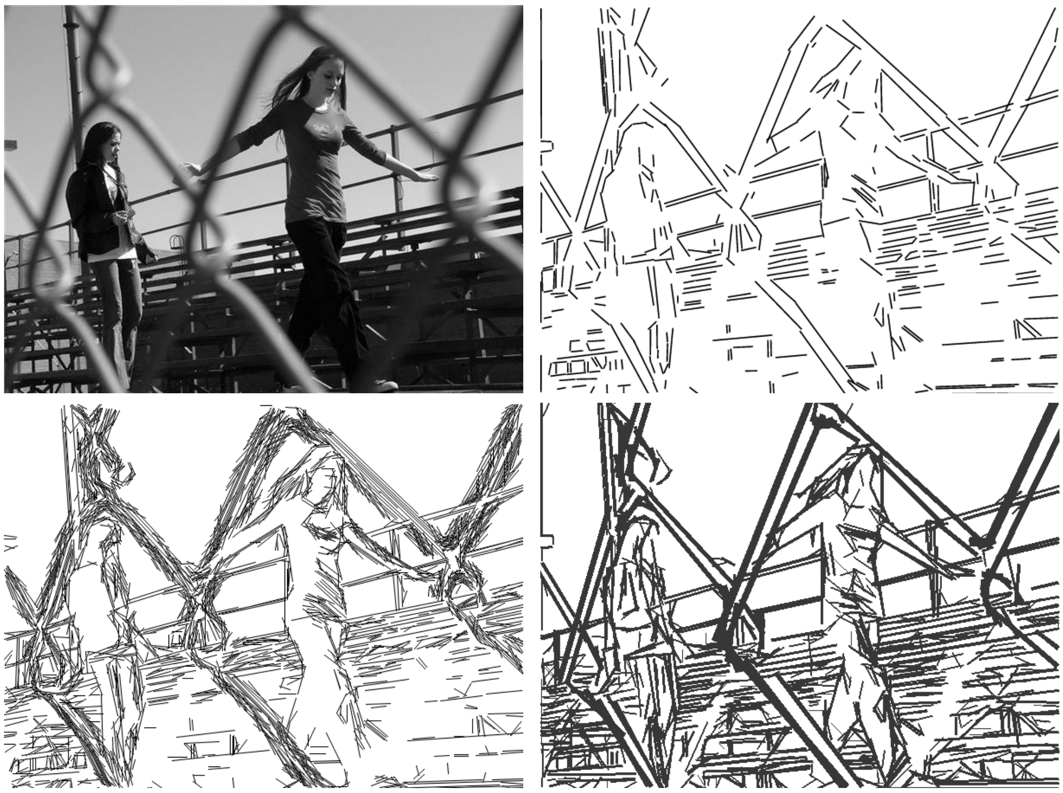


Figure 5.5: Top left: image. Top right: LSD [von Gioi 2010]. Bottom left: STRAIGHT [Guerreiro 2012]. Bottom right: proposed semi-local method.

using a running average framework, the calculation of contextual edges depends only linearly on the pixel count and the number of directions, N. Such linear dependency also occurs in the calculation of local edges and in their combination with contextual ones. This is confirmed in Fig. 5.7, which shows the computation time[1] needed by the proposed method (implemented in C code) to extract line segments for multiple images, as a function of pixel count and for $N = 22$ directions. The trend line for $N = 22$, the number of directions that are used in all experiments, indicates that about 166.6K pixels are processed at each second (*e.g.*, the 512×512 Lenna image takes about 1.5 seconds to compute).

The standard HT requires filling an accumulator array, which depend linearly on the total number of edge points. In our experiments, the standard HT required about 1 to 10 seconds to process each image in this thesis, using MATLAB© code.

Reference [von Gioi 2010] states that the complexity of the LSD method depends linearly with the image pixel count, as illustrated by a plot showing the calculation time needed to extract line segments in various images. In the worst case scenario, *i.e.*, images made up of noise, LSD is able to process about 240K pixels per second. Although the main advantage of local methods such as the LSD is its low com-

[1]All experiments were performed on an Intel© 2.67 GHz machine.

putational complexity, with the drawback of only dealing successfully with simple scenarios, the amount of pixels processed by the LSD is only about 3–4 times greater than the proposed semi-local method.

The complexity of STRAIGHT [Guerreiro 2012] increases linearly with the number of edge points, as the dominating factor in the calculations is the updating of the Hough space of each local HT. In our experiments, the computation time of STRAIGHT varies between approximately 100 and 1000 seconds (*e.g.*, the 512×512 Lenna image takes about 43 seconds to compute), using standard C code, and we obtained rough trend lines indicating that only about 6.6K pixels are processed each second. This complexity is about one order of magnitude greater than the method proposed in this chapter and is prohibitive for many applications. Although a portion of the computation times may be due to a non-optimized implementation, the theoretical analysis of STRAIGHT clearly shows that it is very complex, far exceeding that of any other method that we have tested.

Although only STRAIGHT and the semi-local method proposed here can deal with the complex images that arise in practice, the results above show that the computational complexity of this method is far below STRAIGHT. Furthermore, the complexity of this method is comparable with local methods, *i.e.*, it is only about 3–4 times more complex than the LSD method, despite the ability to handle complex scenarios. This indicates that our semi-local method is efficient in extracting segments of all lengths and widths in complex scenarios.

5.4 Comparison of proposed line segment extractors

In this thesis, we propose two line segment detectors. The first is denoted as STRAIGHT and is detailed in chapter 4, and the second is the semi-local method described in this chapter.

The first method, STRAIGHT, extends the classic Hough Transform (HT) methodology to enable the robust detection of line segments, by including connectivity in the voting process. We believe that the approach that was chosen, of including connectivity in the voting process, is the only HT approach in the literature that can successfully deal with images that include both a large amount of edge points and collinear line segments. For this reason, we believe that it is the only HT approach that can successfully deal with the complex images that are obtained in unconstrained real life scenarios. However, despite the virtue of this approach, its computational complexity can be deemed prohibitive for standard applications, since our tests reported computation times of between 100 and 1000 seconds for each image, of standard high-definition sizes. In the section where the future work is discussed, section 8.2, we point out that it would be very useful to create a more efficient method that makes use of the idea of including connectivity into the HT.

The semi-local method, on the other hand, simply replaces the edge detector of a standard local method with the edge detector presented in this thesis. This method improves the other family of classic approaches to extract line segments from the image, *i.e.*, local methods, and enables robust detections when the input image is complex. From a computational point of view, this method is only about three times more expensive than the state-of-the-art method Line Segment Detector [von Gioi 2010], thus simple enough to be used in standard applications.

Regarding the quality of the detections, both methods exhibit similar performance. Because STRAIGHT considers that each pixel contains an uncertainty, it is better suited for dealing with edges that are not aligned exactly in a line segment. The semi-local method, on the other hand, is able to extract line segments whose transition is not abrupt, which leads to erroneous detections in STRAIGHT. Finally, regarding the computational complexity, the semi-local method is by far the method that is most efficient in extracting line segments.

5.5 Conclusion

We have presented a new semi-local method for line segment extraction. This method combines contextual and local edges, with explicit handling of connectivity. Our experiments show that it outperforms current methods for line segment extraction in challenging situations, *e.g.*, when dealing with complex images containing several crossing segments of multiple widths, and that its computational efficiency is comparable with simple local methods. We use a contextual edge detection scheme based on two-sample statistical tests, which is a robust way to handle noise.

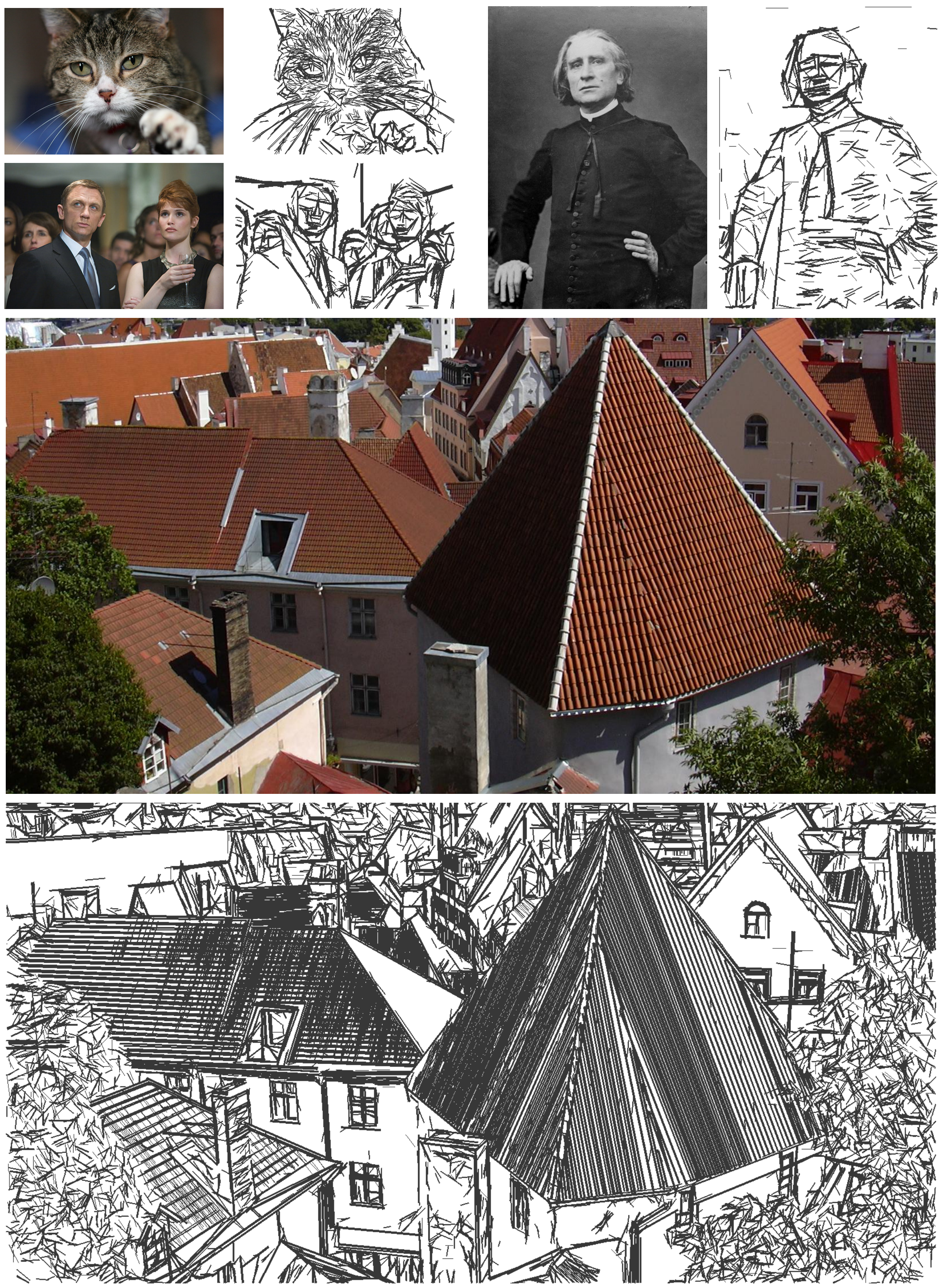


Figure 5.6: Results of the proposed semi-local method for several kinds of real images.

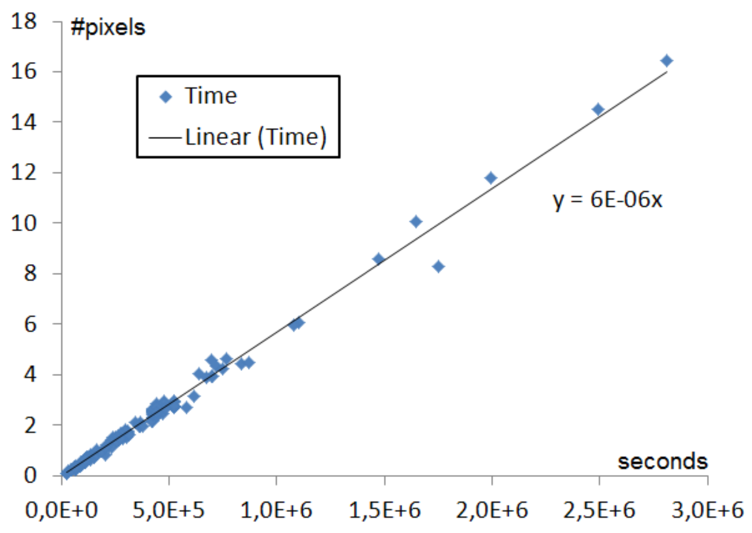


Figure 5.7: Computation time (in seconds) as a function of image pixel count, for a large set of images and in a standard C implementation. About 166.6K pixels per second are processed using $N = 22$ directions.

Chapter 6

Optimized filters for efficient multi-texture discrimination

6.1 Proposed approach

Our approach, presented in [Guerreiro 2013b, Guerreiro 2010], also follows the block diagram in Fig. 2.9. We propose the usage of *optimal FIR filters*, as illustrated above, where a standard filter bank is replaced by one made up of adjustable coefficients, obtained through supervised learning. Rather than the two-dimensional large footprint filters of typical filter banks, we use: a) one-dimensional filters, applied horizontally and vertically, to perform orientation-dependent discrimination; or b) ring-shaped filters, to perform rotationally-invariant discrimination of textures. These filters are simple to compute and we show that they suffice to extract the relevant textural features.

In the local energy function, we compute the first four moments of each filter output and weight their contribution to the overall classification. This results computationally very simple and we show that it approximates typical local energy functions, including the computationally complex and exceptionally discriminant clustering and histogram matching approach of [Malik 2001].

We use a Maximum-Likelihood classifier with normalized Euclidean distance to obtain the class of a texture sample. To learn the filter bank coefficients, we define an objective function to minimize, which, due to the use of moments, results non-convex. Supervised learning is then performed by using a Genetic Algorithm, whose properties enable an apt solution.

We conduct an experimental analysis of our method using the publicly available Brodatz [Brodatz 1966] and VisTex [vis 2002] albums. The Brodatz database has been used extensively to evaluate the performance of texture discrimination methods when dealing with images exhibiting mild variations. We conclude that our method outperforms state-of-the-art methods, such as GFB, and LBP, both in terms of accuracy and computational simplicity. By repeating the experiments in the VisTex database, we further conclude that the performance of our method is consistent among different databases. We report the actual parameters of the filters learned using the Brodatz album, to enable a quick implementation of a texture discrimination method using a similar database.

The organization of the remaining of the chapter is as follows. Section 6.2.1 introduces the framework of the optimized filters for efficient multi-texture discrimination. They are particularized for the cases of rotationally discriminant and rotationally invariant discrimination. Section 6.3 describes the usage of a Genetic Algorithm to learn the filter parameters in these two scenarios. In Section 6.4, we report the experimental results and section 6.5 concludes the chapter.

6.2 Optimized Filters for Efficient Multi-texture Discrimination

6.2.1 Framework for optimized filters

Consider a patch of size $p \times q$ of the input image, whose central point corresponds to pixel (x, y) and contains a texture. We name it *texture patch* and denote it as $\boldsymbol{T}_{xy} \in \mathbb{R}^{p\times q}$. To estimate its class, we define a *texture discrimination function*, $\boldsymbol{\phi} : \mathbb{R}^{p\times q} \to \mathbb{R}^{v}$, which will incorporate our optimized filter approach. This function takes a texture patch as input and computes a feature vector with measurements of the textural information contained in it, represented as $\boldsymbol{\phi}\left(\boldsymbol{T}_{xy}\right)$. We assume that the feature vectors belonging to class c constitute a random variable following a v-dimensional Gaussian distribution of average $\boldsymbol{\mu}_c$, which we denote *class centroid*, and covariance $\boldsymbol{\Sigma}_c$, *i.e.*, $p(\boldsymbol{\phi}\left(\boldsymbol{T}\right)|c) = \mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$. Assuming that all texture classes are equally likely, the maximum-likelihood class estimate for a new texture patch, $\hat{c}_{\mathrm{ML}}\left(\boldsymbol{T}\right)$, is given by

$$\begin{aligned}\hat{c}_{\mathrm{ML}}(\boldsymbol{T}) &= \arg\max_{c} p(\boldsymbol{\phi}(\boldsymbol{T})|c)\\ &= \arg\min_{c} \sqrt{\left(\boldsymbol{\phi}(\boldsymbol{T}) - \boldsymbol{\mu}_c\right)^T \boldsymbol{\Sigma}_c^{-1} \left(\boldsymbol{\phi}(\boldsymbol{T}) - \boldsymbol{\mu}_c\right)}\\ &= \arg\min_{c} \|\boldsymbol{\phi}(\boldsymbol{T}) - \boldsymbol{\mu}_c\|^2_{\boldsymbol{\Sigma}_c^{-1}}, \forall_{c\in\{1,\dots,C\}}, \end{aligned} \tag{6.1}$$

where $\|\cdot\|^2_{\boldsymbol{\Sigma}_c^{-1}}$ is the Mahalanobis distance and C is the total number of classes to be discriminated. The calculation of the Mahalanobis distance requires C multiplications with symmetric matrices $\boldsymbol{\Sigma}_c^{-1}$ of dimension $v \times v$, for each point in the image. Because our emphasis is in achieving computationally simple texture discrimination, we impose that matrix $\boldsymbol{\Sigma}_c$ is a diagonal matrix. The Mahalanobis distance then becomes the normalized Euclidean distance. We show how the $\boldsymbol{\phi}(\boldsymbol{T})$, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ quantities are computed later.

Although it is possible to define a discrimination filter, $\boldsymbol{\phi}(\cdot)$, that handles a complete texture patch directly, this is not the typical approach in texture discrimination methods – or nature. Julesz [Julesz 1981] studied extensively the way humans perceive textures and proposed the *theory of textons*. Textons are the basic subelements of textures, *i.e.*, primitives such as oriented edges, collinearities, endpoints of line segments, corners, dots. These primitives are then repeated within a texture according to certain placement rules. Julesz considered that humans discriminate textures by first identifying individual textons and then aggregating information with respect to their occurrence rates, locations, etc. Typical discrimination methods use this approach too: they analyze smaller portions of the texture patch – the textons – and combine the overall data into a discriminating feature vector.

To incorporate this approach into our framework, we process each texton within the texture patch with a *texton analysis filter* and then use an *integration function* to build the final feature vector. Let $\boldsymbol{J}_{xy} \in \mathbb{R}^{a\times a}$ be a texton, *i.e.*, a small portion of the texture patch, whose center corresponds to pixel (x, y) of the input image. Since the description of the primitives contained in each texton is conceptually related to feature and edge points, which typically span only a few pixels, we use $a = 7$ except where noted. Let $\boldsymbol{\psi} : \mathbb{R}^{a\times a} \to \mathbb{R}^{I}$ be texton analysis filters, undefined for now, where I represents the number of filters that are applied to each texton. A *texton feature vector* is then $\boldsymbol{\psi}\left(\boldsymbol{J}_{xy}\right)$. The individual texton feature vectors of a texture patch are aggregated in $\boldsymbol{\Psi} : \mathbb{R}^{p\times q} \to \mathbb{R}^{p\times q\times I}$,

$$\boldsymbol{\Psi}(\boldsymbol{T}_{xy}) = \begin{bmatrix} \boldsymbol{\psi}\left(\boldsymbol{J}_{x-\frac{p}{2},y-\frac{q}{2}}\right) & \cdots & \boldsymbol{\psi}\left(\boldsymbol{J}_{x-\frac{p}{2},x+\frac{q}{2}}\right) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\psi}\left(\boldsymbol{J}_{x+\frac{p}{2},x-\frac{q}{2}}\right) & \cdots & \boldsymbol{\psi}\left(\boldsymbol{J}_{x+\frac{p}{2},x+\frac{q}{2}}\right) \end{bmatrix}. \tag{6.2}$$

The aggregation of texton feature vectors, $\boldsymbol{\psi}\left(\boldsymbol{J}_{\cdot,\cdot}\right)$, makes $\boldsymbol{\Psi}(\boldsymbol{T}_{xy})$ a third-order tensor, which we denote as *aggregation tensor*. The integration function, $\boldsymbol{\gamma} : \mathbb{R}^{p\times q\times I} \to \mathbb{R}^{v}$, builds a texture feature vector from this tensor,

$$\boldsymbol{\phi}(\boldsymbol{T}_{xy}) = \boldsymbol{\gamma}(\boldsymbol{\Psi}(\boldsymbol{T}_{xy})). \tag{6.3}$$

In this scenario, a discrimination method is determined by defining functions $\boldsymbol{\psi}(\cdot)$ and $\boldsymbol{\gamma}(\cdot)$. This formulation is very general and can be particularized in various ways. In fact, most existing texture discrimination methods can be expressed in this manner.

6.2.2 Texton analysis filters

The texton analysis filters, $\boldsymbol{\psi} : \mathbb{R}^{a\times a} \to \mathbb{R}^{I}$, should be general enough to be able to extract discriminative data from textons and, simultaneously, simple enough to avoid overfitting and excessive computational complexity. In signal processing approaches, this consists of convolving a texton with the impulse response of a filter bank such as, *e.g.*, GFB.

We particularize our framework by defining that the texton analysis filters consist of convolving a texton with the impulse response of a learned filter bank of I different matrices, as in the *Optimal FIR filters* category of optimized filters (see section 2.3.3.2). We impose that the i-th value of the texton feature vector, $\boldsymbol{\psi}_i\left(\boldsymbol{J}_{xy}\right)$, is the sum of the pointwise multiplication of texton $\boldsymbol{J}_{xy}$ with matrix $\boldsymbol{W}_i$, for $i \in \{1, \ldots, I\}$ and matrices $\boldsymbol{W}_i$ defined later. More simply, the i-th layer of the aggregation tensor, $\boldsymbol{\Psi}_i(\boldsymbol{T})$, is a matrix given by the convolution of texture $\boldsymbol{T}$ with matrix $\boldsymbol{W}_i$,

$$\boldsymbol{\Psi}_i(\boldsymbol{T}) = \boldsymbol{T} * \boldsymbol{W}_i, \tag{6.4}$$

where $*$ is the convolution operator. The number of linear filters we apply to the texture patch, I, is an adjustable parameter. Matrices $\boldsymbol{W}_i$ are estimated in the training phase and, naturally, as I increases so does the overall discrimination ability, at the expense of a higher computational cost and more parameters to be learned.

In this thesis, we present two separate discrimination possibilities, one optimized for a rotationally discriminant scenario and another for a rotationally invariant one. This distinction occurs by restricting the elements of matrices $\boldsymbol{W}_i$ that are allowed to have non-zero values. In particular:

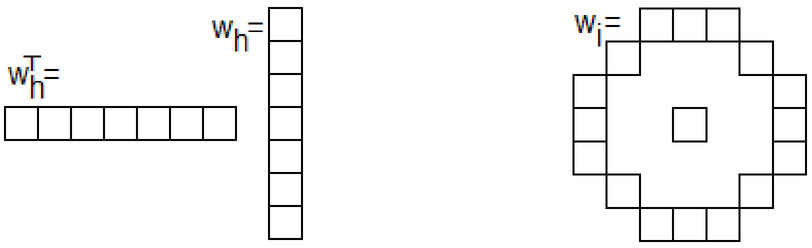
– **Method I – Rotationally discriminant filters** – The output of these filters change when arbitrary rotations are applied to the input texture patches. Most texture discrimination methods are of this type and perform well in texture patches within the same image. We implement these filters using pairs of matrices where the first one implements a vertical support filter, in which only the central $a \times 1$ values are allowed to have non-zero values. The second matrix implements a horizontal support filter where only the central $1 \times a$ values are allowed to have non-zero values. Because the vertical and horizontal filters serve the purpose of extracting general and informative texton data, the coefficients are the same in both scenarios, *i.e.*, the matrices are transposed versions of each other. In this scenario, the number of filters, I, is an even number but only $I/2$ independent sets of coefficients are learned. Fig. 6.1.a illustrates such filters, $\boldsymbol{W}_h$ and $\boldsymbol{W}_h^T$, where $h = \lceil i/2 \rceil$. The aggregation tensor $\boldsymbol{\Psi}(\boldsymbol{T})$ contains the result of I convolutions of $\boldsymbol{T}$ with all $\boldsymbol{W}_h$ and $\boldsymbol{W}_h^T$ matrices,

$$\begin{aligned} \boldsymbol{\Psi}_{2h-1}(\boldsymbol{T}) &= \boldsymbol{T} * (\boldsymbol{W}_h)^T, \\ \boldsymbol{\Psi}_{2h}(\boldsymbol{T}) &= \boldsymbol{T} * \boldsymbol{W}_h. \end{aligned} \tag{6.5}$$

– **Method II – Rotationally invariant filters** – The output of these filters doesn't change when arbitrary rotations are applied. Newer texture discrimination methods are often of this type and their higher generality makes them suitable for recognizing textures among different images, which is important for applications such as content-based image retrieval. This is the first optimized filter approach that is invariant to rotation. It uses matrices $\boldsymbol{W}_i$ where the non-zero elements form a circle of diameter a plus an element at the center of the window, as illustrated in Fig. 6.1.b. The element at the center of the window can function as a reference value for the non-zero elements in the circle, as in [Ojala 2002]. The aggregation tensor $\boldsymbol{\Psi}(\boldsymbol{T})$ contains the result of I convolutions of $\boldsymbol{T}$ with all $\boldsymbol{W}_i$, as in (6.4).

6.2.3 Local energy function

The integration function we propose as a local energy function, $\boldsymbol{\gamma} : \mathbb{R}^{p \times q \times I} \to \mathbb{R}^v$, computes simple yet powerful high-order statistics of the filter outputs. The statistics are the average, the standard deviation and modified standardized moments



(a) rotationally discriminant (b) rotationally invariant

Figure 6.1: Location of non-zero values in texton analysis filters

three and four, the skewness and kurtosis, respectively, of each layer of the aggregation tensor, $\boldsymbol{\Psi}_i(\boldsymbol{T})$,

$$\begin{aligned}
\kappa_1 &= \mu_1' \equiv \mu, \\
\kappa_2 &= \sqrt[2]{\mu_2' - {\mu_1'}^2} \equiv \sigma, \\
\kappa_3 &= \sqrt[3]{\left(\mu_3' - 3\mu_2'\mu_1' + 2{\mu_1'}^3\right)/\kappa_2^3}, \\
\kappa_4 &= \sqrt[4]{\left(\mu_4' - 4\mu_3'\mu_1' + 6\mu_2'{\mu_1'}^2 - 3{\mu_1'}^4\right)/\kappa_2^4}, \\
\mu_l' &= \mathrm{E}\left[\boldsymbol{\Psi}\left(\boldsymbol{T}\right)^l\right] = \frac{1}{pq}\sum_{y=1}^{q}\sum_{x=1}^{p}\left(\boldsymbol{\Psi}_i\left(\boldsymbol{T}, x, y\right)\right)^l,
\end{aligned} \tag{6.6}$$

where κ_j is the j-th statistic we compute and μ_l' is the l-th moment about the origin of $\Psi_i(\boldsymbol{T})$. The modification of the standardized moments $j = 3$ and 4, by applying the j-th root, is intended to *linearize* κ_j and enable a better approximation to a Gaussian distribution. This is supported by experimental results that show an improved classification rate in this scenario. Quantity μ_l' can be computed efficiently, for every l, by: a) raising each element of $\boldsymbol{\Psi}_i\left(\boldsymbol{T}\right)$ to the power l and; b) using a sliding window box filter.

The final discrimination vector, $\boldsymbol{\phi}(\boldsymbol{T})$, is obtained by arranging each $\kappa_j\left(\boldsymbol{\Psi}_i\left(\boldsymbol{T}\right)\right)$ in a discrimination vector of size $v = 4I$,

$$\boldsymbol{\phi}(\boldsymbol{T}) = \gamma\left(\boldsymbol{\Psi}(\boldsymbol{T})\right) = \begin{bmatrix} \kappa_1\left(\boldsymbol{\Psi}_1(\boldsymbol{T})\right) \\ \kappa_2\left(\boldsymbol{\Psi}_1(\boldsymbol{T})\right) \\ \kappa_3\left(\boldsymbol{\Psi}_1(\boldsymbol{T})\right) \\ \kappa_4\left(\boldsymbol{\Psi}_1(\boldsymbol{T})\right) \\ \vdots \\ \kappa_1\left(\boldsymbol{\Psi}_I(\boldsymbol{T})\right) \\ \kappa_2\left(\boldsymbol{\Psi}_I(\boldsymbol{T})\right) \\ \kappa_3\left(\boldsymbol{\Psi}_I(\boldsymbol{T})\right) \\ \kappa_4\left(\boldsymbol{\Psi}_I(\boldsymbol{T})\right) \end{bmatrix}. \tag{6.7}$$

By computing such comprehensive statistical measures of the filter outputs, our integration function is able to approximate typical local energy functions used in the literature, including the exceptionally discriminant but computationally complex one of Malik et al. [Malik 2001]. In this method, the I-dimensional filter outputs at each pixel are assigned to one of a set of previously identified textons. The different textons within a texture patch make up a histogram that is used to infer its texture class, by finding the closest stored class histogram centroid according to some histogram distance metric. The clustering of the filter outputs into textons, with the consequent loss of information, is deemed necessary in [Malik 2001] to enable the use of relatively simple one-dimensional histograms and histogram distance metrics. The alternative of using I-dimensional histograms of the filter outputs and I-dimensional histogram distance metrics is computationally prohibitive.

This alternative is simple to implement in our method, since the high-order statistics we compute of the filter outputs make up a parametric descriptions of such histograms. An I-dimensional histogram is described with a parametric set of only $4I$ numbers and the (normalized) Euclidean distance substitutes costly I-dimensional histogram distance metrics. In sum, our approximation reduces the accuracy of each histogram by representing it parametrically but, by not having to reduce its dimensionality, preserves important discriminant information. The learning scheme is expected to make the best use of the characteristics of our method.

6.3 Learning texture filters

6.3.1 Formulation as a minimization problem

The filter parameters $\boldsymbol{W}$ that constitute the texture discrimination function, $\boldsymbol{\phi}(.)$, are obtained through supervised learning. Consider a training set of N texture patches from C textures classes, and a map $k : \{1, ..., N\} \rightarrow \{1, ..., C\}$ that contains the knowledge of which class c corresponds to patch n. We obtain $\boldsymbol{W}$ that maximizes the overall probabilities of choosing class k_n given $\boldsymbol{T}_n$, for all $n \in \{1, \dots, N\}$,

$$\begin{aligned}
\hat{\boldsymbol{W}} &= \arg\max_{\boldsymbol{W}} \prod_{n=1}^{N} p\left(k_n | \boldsymbol{\phi}\left(\boldsymbol{T}_n\right)\right) \\
&= \arg\max_{\boldsymbol{W}} \prod_{n=1}^{N} \frac{p\left(\boldsymbol{\phi}\left(\boldsymbol{T}_n\right) | k_n\right)}{\sum_{c=1}^{C} p\left(\boldsymbol{\phi}\left(\boldsymbol{T}_n\right) | c\right)} \\
&\simeq \arg\min_{\boldsymbol{W}} \sum_{n=1}^{N} \frac{\left\|\boldsymbol{\phi}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{k_n}\right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}_{k_n}}}{\sum_{c=1}^{C} \left\|\boldsymbol{\phi}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{c}\right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}_{c}}} \\
&\simeq \arg\min_{\boldsymbol{W}} \sum_{n=1}^{N} \left\|\boldsymbol{\phi}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{k_n}\right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}_{k_n}}, \qquad (6.8)
\end{aligned}$$

where all classes are assumed equally likely; the denominator, which favors solutions where texture classes have different averages, is removed by imposing that all averages $\hat{\boldsymbol{\mu}}_C$ must be different; and the discrimination vector is assumed to be a vector with Normal distribution, for computational simplicity. Quantity $\hat{\boldsymbol{\mu}}_c$ represents the sample average and $\hat{\boldsymbol{\Sigma}}_c$ the sample covariance, which is forced to be a diagonal matrix for computational simplicity,

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_c &= \frac{1}{\sum_{\{n:k_n=c\}} 1} \sum_{\{n:k_n=c\}} \boldsymbol{\phi}(\boldsymbol{T}_n), \qquad (6.9) \\
\operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_c\right) &= \left[\frac{1}{\sum_{\{n:k_n=c\}} 1} \sum_{\{n:k_n=c\}} \boldsymbol{\phi}(\boldsymbol{T}_n)^2\right] - \hat{\boldsymbol{\mu}}_c^2.
\end{aligned}$$

For the training of the rotationally invariant estimator, instead of rotating the texture patches, we change the order of the parameters located in the circular area of $\boldsymbol{W}$, as illustrated in Fig. 6.2. The parameters in the outter circle shift their position but not the parameter in the center of $\boldsymbol{W}_i$, w_{i_0}. This procedure has the advantage of being computationally simpler than rotating texture patches and of not introducing rotation blur, which has been reported in [Ojala 2002] to introduce classification errors. We denote the amount of shift with letter $\theta \in \{1, \ldots, \Theta\}$, where $\boldsymbol{W}_i(\theta)$ indicates that the parameters in $\boldsymbol{W}_i$ (except $\boldsymbol{W}_{i_0}$) are shifted by θ positions. Θ is the number of parameters in the outter circle of $\boldsymbol{W}_i$.

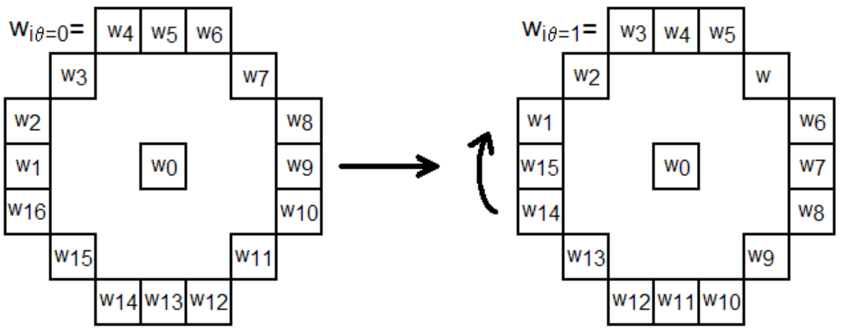


Figure 6.2: Illustration of the shift of parameters of $\boldsymbol{W}_i$ located in the outer circle. The central parameter remains unaltered

In this scenario, equations (6.8) and (6.9) become

$$\begin{aligned}
\hat{\boldsymbol{W}} &= \arg\min_{\boldsymbol{W}} \sum_{n=1}^{N} \sum_{\theta=1}^{\Theta} \left\| \boldsymbol{\phi}_{\boldsymbol{W}\theta}\left(\boldsymbol{T}_n\right) - \hat{\boldsymbol{\mu}}_{k_n} \right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}_{k_n}}, \\
\hat{\boldsymbol{\mu}}_c &= \frac{1}{\sum\limits_{\{n:k_n=c\}} \Theta} \sum_{\{n:k_n=c\}} \sum_{\theta=1}^{\Theta} \boldsymbol{\phi}_{\boldsymbol{W}\theta}\left(\boldsymbol{T}_n\right), \qquad (6.10) \\
\operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_c\right) &= \frac{1}{\sum\limits_{\{n:k_n=c\}} \Theta} \sum_{\{n:k_n=c\}} \sum_{\theta=1}^{\Theta} \boldsymbol{\phi}_{\boldsymbol{W}\theta}\left(\boldsymbol{T}_n\right)^2 - \hat{\boldsymbol{\mu}}_c^2.
\end{aligned}$$

6.3.2 Supervised learning using a Genetic Algorithm

The minimizations in (6.8) or (6.10) constitute optimization problems in the domain of the internal parameters of $\boldsymbol{\phi}(.)$, *i.e.*, $\boldsymbol{W}$. These optimization problems are non-trivial, due to the fact that the objective functions are non-convex on $\boldsymbol{W}$. We use a Genetic Algorithm (GA) [Mitchell 1996] for the purpose, but other methods such as, *e.g.*, particle swarm optimization [Poli 2008], could be used instead. Although GAs provide no guarantee of convergence to the global optimum, they have been used with success in several non-convex problems, since the crossover step moves the population away from the local optima that a traditional local algorithm (*e.g.*, gradient descent) might get stuck in. Although GAs are easy to implement, they have the disadvantage of being slow — the parameters of each filter whose results are reported in Section 6.4 took several hours to compute. However, since the training phase occurs offline and only once for each set of training images and parameters, the computation time of the GA does not influence the run-time performance of this method.

Other methods have used GA for texture discrimination: reference [Lam 2008] proposes a GA scheme to evolve discriminators of pixel histograms of different textures, reference [Song 2003] proposes to evolve Mathematical operations with pixels in textons, to enable fast discrimination, and, in [Aurnhammer 2007], better discriminating GLCM features are evolved. However, the discrimination ability of these methods is far below the state-of-the-art ones referred in Section 2.3.

The optimization procedure we propose is summarized as Alg. 4. The GA starts by creating an initial generation of G_p filter coefficients, $\boldsymbol{W}$, initialized with uniform random noise of magnitude G_n. For each member $\boldsymbol{W}_p$, texture feature vectors $\boldsymbol{\phi}_{\boldsymbol{W}_p}(\boldsymbol{T}_n)$ are computed, as are all class sample averages $\hat{\boldsymbol{\mu}}_c$ and covariances $\hat{\boldsymbol{\Sigma}}_c$. The classification error of each member, ε_p, is computed and the G_pG_e members with the smallest classification errors are selected, *i.e.*, the elite member set. A new generation is created by copying the elite member set of the previous generation and creating new G_pG_c crossover and G_pG_m mutation members. Crossover members are created by mixing the coefficients of two elite members, and mutation members, by taking an elite member and adding uniform random noise of magnitude G_n. The elite members are selected and mixed in an uniformly random way. The crossover step enables large jumps in the solution space and, since the mutation magnitude G_n decreases by G_u in every generation, the mutation step enables elite members to converge to their local optima. The new generation is evaluated and the process is repeated until the mutation magnitude has a low value and a good solution is achieved.

6.4 Experiments

We describe experiments using the Brodatz [Brodatz 1966] and the VisTex [vis 2002] photographic albums. The Brodatz album, illustrated on the top of Fig. 6.3 and Fig. 6.5, has been the *de facto* standard for evaluating texture analysis methods for many years, with hundreds of studies having been applied to its images. The VisTex album, illustrated on the bottom of Fig. 6.3, provides texture images that are different from the Brodatz ones but are also representative of real world scenarios. Other databases, *e.g.*, Columbia-Utrecht database [Dana 1999], have been used when the goal is to classify materials exhibiting dramatic appearance variability (arising from very different viewpoints and lighting). Since we particularly care with the analysis of the texture patterns under mild variations, we consider the Brodatz and VisTex databases to be adequate.

6.4.1 Training

For the Brodatz database, we use a training set of $N = 1025$ texture patches from $C = 41$ classes. Each class corresponds to an image from this album, from which 25 non-overlapping patches are extracted. Each image in the VisTex database corresponds to one of $C = 81$ classes, from which 9 non-overlapping patches are extracted, making up a total of $N = 729$ texture patches. The mean value of each image is set to a constant value, to enable a more accurate measurement of the true discriminative ability of our method. We use patches of size $p \times q = 60 \times 60$ for the Brodatz album and 80×80 for the VisTex one (as Fig. 6.3 illustrates, the repetitive

Algorithm 4 Genetic Algorithm for learning $\boldsymbol{W}$

```
 1: input: map $k_n$, patches $\boldsymbol{T}_n$, texton size $a$, number of filters $I$, GA parameters:
    population $G_p$, noise magnitude $G_n$, update $G_u$, rate of elitism $G_e$, cross-over
    $G_c$ and mutation $G_m$ (with $G_e + G_c + G_m = 1$)
 2: % initialize $\boldsymbol{W}$ with uniform random noise
 3: $\boldsymbol{W}_{pi} \sim G_n \mathcal{U}(-1, 1, a)$, $\forall p \in \{1, \ldots, G_p\}, i \in \{1, \ldots, I\}$
 4: repeat
 5:   % For all members $\boldsymbol{W}_p$ of the population
 6:   for $p = \{1, \ldots, G_p\}$ do
 7:     $\hat{\boldsymbol{\mu}}_c \leftarrow 0$, $\hat{\boldsymbol{\Sigma}}_c \leftarrow 0$, $\#c \leftarrow 0$, $\forall c = \{1, \ldots, C\}$
 8:     for $n = \{1, \ldots, N\}$ do
 9:       % compute texture feature vectors (eq. (6.7))
10:       $\boldsymbol{\phi}_{\boldsymbol{W}_p}(\boldsymbol{T}_n) \leftarrow \gamma\left(\boldsymbol{\Psi}_{\boldsymbol{W}_p}(\boldsymbol{T}_n)\right)$
11:       $\hat{\boldsymbol{\mu}}_{k_n} \leftarrow \hat{\boldsymbol{\mu}}_{k_n} + \boldsymbol{\phi}_{\boldsymbol{W}_p}(\boldsymbol{T}_n)$
12:       $\operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_{k_n}\right) \leftarrow \operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_{k_n}\right) + \boldsymbol{\phi}^2_{\boldsymbol{W}_p}(\boldsymbol{T}_n)$
13:     $\hat{\boldsymbol{\mu}}_c \leftarrow \hat{\boldsymbol{\mu}}_c / \#c$, $\forall c$
14:     $\operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_c\right) \leftarrow \operatorname{diag}\left(\hat{\boldsymbol{\Sigma}}_c\right) / \#c - \hat{\boldsymbol{\mu}}_c^2$, $\forall c$
15:     % compute classification error (eq. (6.8))
16:     $\varepsilon_p = \sum_{n=1}^{N} \left\| \boldsymbol{\phi}_{\boldsymbol{W}_p}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{k_n} \right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}_{k_n}}$
17:   % decrease magnitude of mutation noise
18:   $G_n \leftarrow G_n G_u$
19:   % create population for next generation
20:   save $G_p G_e$ best members, $W_p \leftarrow W_{\arg\min_{p'} \varepsilon_{p'}}$
21:   crossover $G_p G_c$ members, $W_p \leftarrow \operatorname{mix}\left(W_{p'}, W_{p''}\right)$
22:   mutate $G_p G_m$ members, $W_{pi} \leftarrow W_{p'i} + G_n \mathcal{U}(-1, 1, a)$
23: until $G_n <$ threshold
24: output: $\boldsymbol{W}_0$
```

pattern of the textures in the Brodatz album is typically smaller than the one of the textures in the VisTex album) and texton patches of size $a \times a = 7 \times 7$. Although there are no supported quantitative ways to determine the parameters of the GA described in Section 6.3, it is typical that the number of members that are used in each generation is at least one order of magnitude larger than the number of parameters being generated, so that sufficient variability exists within the gene pool. In our scenario, in which the parameters being estimated are continuous (instead of, *e.g.*, binary), the required variability is much higher, so we use a population of $G_p = 1000$ members, which is about two orders of magnitude. We use $G_e = 10\%$ elitism rate, $G_c = 50\%$ crossover rate and $G_m = 40\%$ mutation rate. For initialization, we use random uniform noise of mutation magnitude $G_n = 1$ and it decreases by $G_u = 0.8$ in every generation. When the mutation magnitude is so low that it no longer meaningfully affects the elitist members, *e.g.*, 10^{-3}, the algorithm stops.

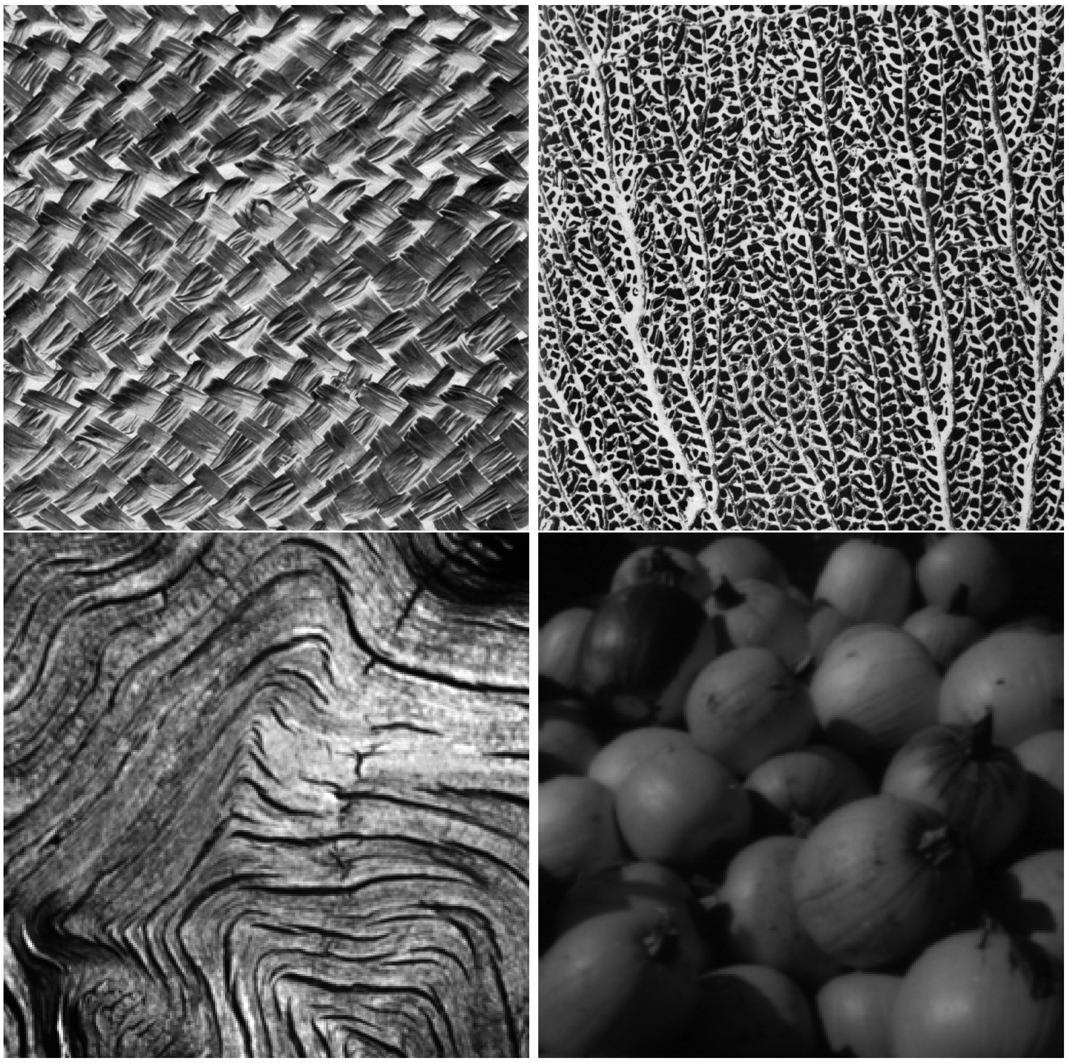


Figure 6.3: Top: "Raffia weave" (D18) and "fossilized sea fan" (D87) from the Brodatz database; Bottom: "wood" and "food" from the VisTex database

6.4.2 Filter performance

6.4.2.1 Comparison with other methods

We evaluate the filters obtained in the training phase, $\phi(.)$, in three scenarios. In the first one, we classify texture patches from the same classes used in the training phase and using the same parameters. We compute a filter using only the Brodatz album and another one using the VisTex album. This illustrates applications where we know beforehand the set of textures we need to discriminate, so we compute a filter $\phi(.)$ that is highly discriminant and efficient in this scenario. In the second scenario, we use both filters on the database 1 (DB1) of [Partio 2007], a large comparative study of texture discrimination methods. The texture patches have size 60×60, as in the first scenario of the Brodatz database. By doing so, we are testing the

Table 6.1: Correct classification rates for proposed Methods I and II and the state-of-the-art (see details in the text)

Method I				Method II				GLCM	GFB	LBP
I	Brodatz 60×60	DB1 60×60	DB1 160×160	I	Brodatz 60×60	DB1 60×60	DB1 160×160	DB1 160×160	DB1 160×160	DB1 160×160
2	91.90%	78.67%	98.70%							
4	99.02%	87.73%	99.48%	4	74.87%	49.84%	79.65%			
6	99.32%	91.13%	99.90%	6	79.82%	51.40%	83.45%			
I	VisTex 80×80	DB1 60×60	DB1 160×160	I	VisTex 80×80	DB1 60×60	DB1 160×160	66.7%	92.2%	90.7%
2	87.65%	79.06%	98.25%							
4	98.76%	87.71%	99.76%	4	65.80%	54.35%	92.28%			
6	99.38%	87.60%	99.74%	6	73.89%	60.21%	89.36%			

generalization ability of filters $\phi(.)$, since 45% of the classes in database DB1 are not present in the Brodatz database and there are no common classes with the VisTex database. In the third scenario, we use both filters on DB1 but with texture patches of size 160×160, the same size used in the tests reported in [Partio 2007]. This allows us to directly compare the performance of our method with other popular methods. DB1 [Partio 2007] consists of $N = 960$ non-overlapping texture patches from $C = 60$ texture classes, where each corresponds to an image of the Brodatz album.

The results we obtained are shown in Table 6.1. The correct classification rates of our method on the textures used for training, shown in the left Brodatz or VisTex columns, illustrate the validity of our approach. Method I achieves classification rates above 90% for the Brodatz album, even for $I = 2$, *i.e.*, a single horizontal and vertical convolution, and respective high-order statistics, which are very simple to compute. This is especially noteworthy since the texture patches are relatively small, 60×60, which makes classification more challenging, as we argue in the following section. Method II achieves an inferior but still positive performance. By enabling high discrimination in a multi-texture scenario, in a computationally simple way, our method surpasses current optimized filter approaches.

Using the learned filters on database DB1 [Partio 2007] yields lower classification rates, displayed in the *DB1* – 60×60 columns, particularly for smaller I. The still high correct classification rate for Method I shows that the learning step creates filters that are able to discriminate textures in general. However, the reduction in this rate rate also shows the effect of overfitting to the training set. In Method II, the lower classification rates on the training set and the sharper decrease when applied to database DB1 indicates that this implementation of a rotationally invariant classifier requires more filters than Method I for proper generalization to occur.

To compare our method with the state-of-the-art, we use the DB1 database and texture patches of size 160×160, as in the comparative study reported in [Partio 2007]. Since our filters are learned using 60×60 or 80×80 texture patches, in the databases described in Section 6.4.1, imperfect generalization partly accounts for the errors we obtain. Even in this scenario, our results, shown in columns *DB1* – 160×160, show that Method I, even with $I = 2$, achieves an accuracy of 98.70% for the Brodatz album and 98.25% for the VisTex one. These correct classification rates are significantly higher than the ones of the methods tested in [Partio 2007] and, in particular, than the ones obtained with GLCM, GFB and LBP. Naturally, increasing I, yields even higher accuracy. Table 6.1 also shows that the rotationally invariant Method II is able to successfully classify 83.45% and 89.36% of the patches in the Brodatz and VisTex albums, respectively, with $I = 6$, regardless of their rotation angle.

6.4.2.2 Influence of texture patch size

Table 6.1 reports an improved performance of our method for database DB1, obtained by simply by increasing the texture patch size from 60×60 to 160×160. To elaborate on the influence of texture patch size, Fig. 6.4 shows the correct classification rates of Method I, for $I = 2$ (left) and $I = 4$ (right), as a function of $p = q$ used in the test phase, *i.e.*, the patch size (the training phase used patches of size 60×60 for the Brodatz database and 80×80 for the VisTex database, as described in Section 6.4.1). The gray lines in the plots correspond to each of the classes in the training set and the black line is the average correct classification rate. All plots show that, as the texture patch size increases, the classification rate increases accordingly.

The two lower correct classification rates for $I = 2$, for the Brodatz album, occur for images D32 and D29, shown in the left and center of Fig. 6.5. The image on the right of Fig. 6.5 is D49, the one that leads to the higher correct classification rate. By observing these textures, we note that the repetitive pattern of the images on the left and the center is quite large, thus requiring a large patch to capture all the necessary discriminating features. On the other hand, the image on the right is mostly composed of high frequency light-to-dark transitions, which can be captured by using a small patch. Method I with $I = 4$ and for the Brodatz album, on the other hand, successfully discriminates all textures with smaller patches. This occurs due to its increased resources. Note that, for texture patches of size 160×160, as used in the experimental tests of [Partio 2007], both methods show near-perfect classification for most classes. Note that, because the repetitive patterns in the images of the VisTex database are much larger than in the Brodatz database, larger texture patch sizes are needed for lower classification error.

The patch size is an important factor in discrimination methods. Patches should be small enough to reduce texture boundary issues but also large enough to include sufficient repetitive textural patterns for good discrimination. For deterministic textures, characterized by a set of primitives and a placement rule (*e.g.*, a tile floor), larger patch sizes are typically required. Stochastic textures, on the other hand, do not have easily identifiable primitives (*e.g.*, granite, bark, sand) and typically require smaller patch sizes.

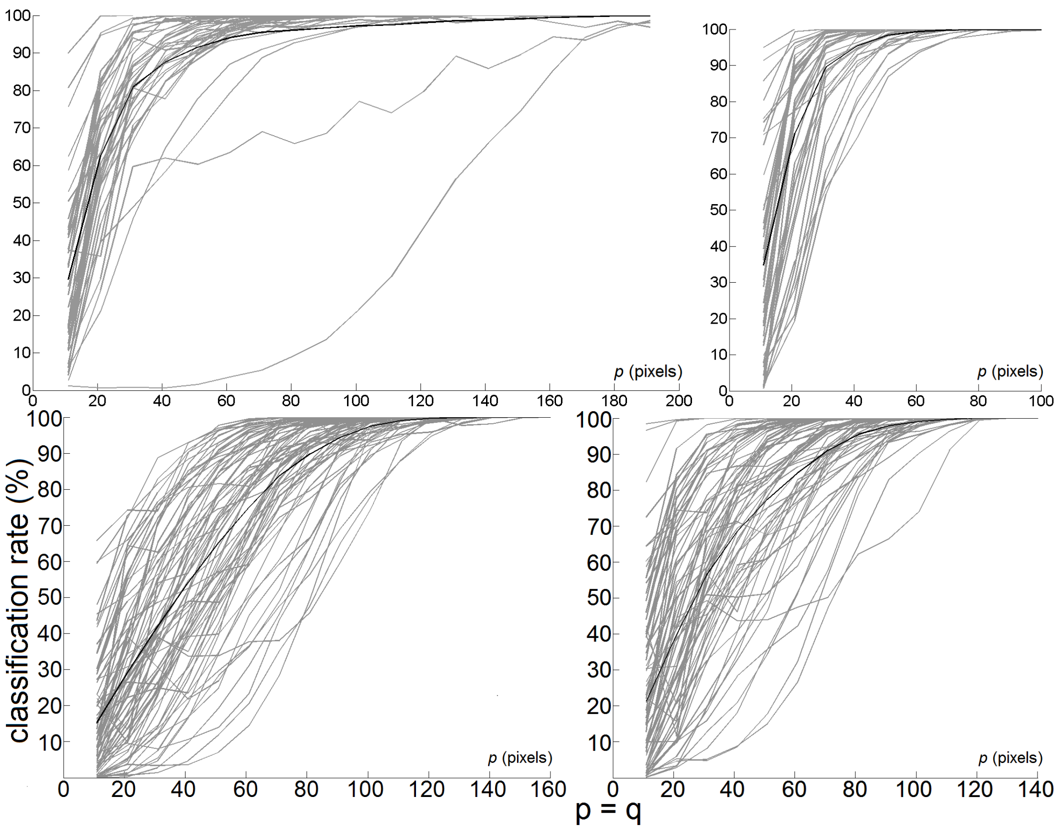


Figure 6.4: Correct classification rate of each texture class as a function of texture patch size $p \times q$ ($p = q$), using Method I and $I = 2$ (left) and $I = 4$ (right). The average value is represented by the darker line. Top: Brodatz database; Bottom: VisTex database

6.4.2.3 Influence of the number of moments

As mentioned in Section 6.2.3, statistical moments make up parametric descriptions of histograms. Naturally, the larger the number of moments, more complex are the histograms allowed to be. Fig. 6.6 shows how the number of moments influences the correct classification rate of Method I, for several values of I and using the Brodatz database. The method is trained as described in Section 6.4.1 with the exception that fewer moments are used. As expected, fewer moments decrease the classification accuracy. For higher values of I, the effect of coarser histogram descriptions is compensated by the extra filters.

6.4.2.4 Influence of the texton patch size

Now, Method I is trained as described in Section 6.4.1, for several I and using the Brodatz database, with the exception that textons of distinct sizes are used. Fig. 6.7 shows the classification rates for various texton sizes. For texton size $a = 3$, the correct classification rates are already close to their maximum values, and larger

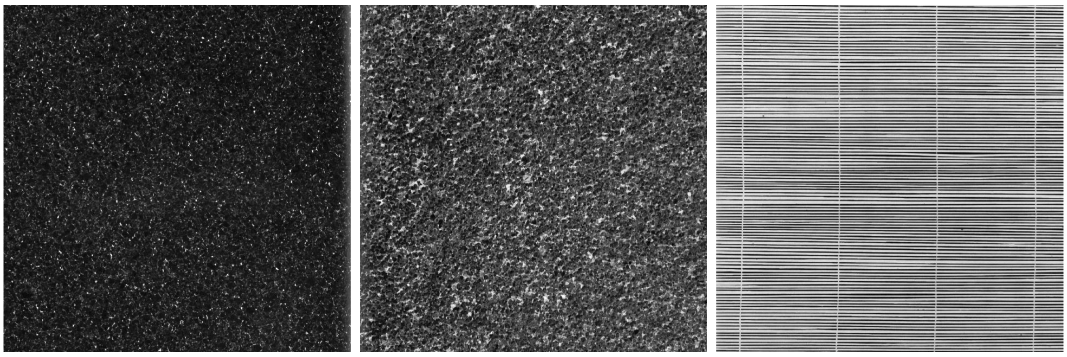


Figure 6.5: "Pressed cork" (D32), "beach sand" (D29), and "screening straw" (D49) from the Brodatz database. Method I, with $I = 2$, has trouble classifying D32 and D29 but not D49; however, with $I = 4$, all these texture are correctly classified

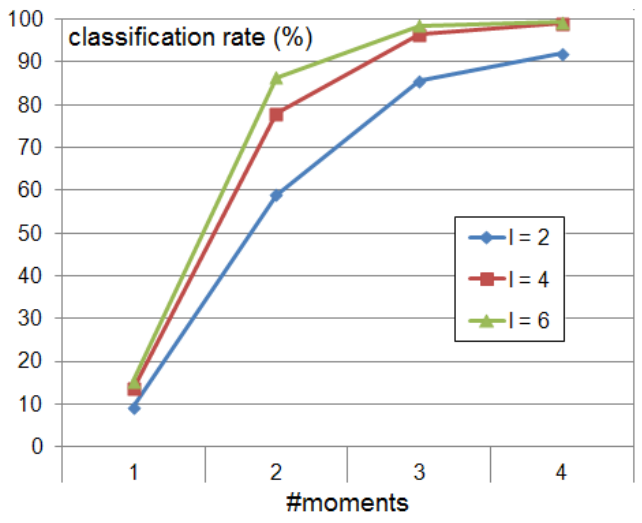


Figure 6.6: Influence of the number of moments on the correct classification rate of Method I, for the Brodatz database

textons, containing more contextual information, only improve classification up to about $a = 7$. This shows that the most discriminant data (for the Brodatz album) is contained in small 3×3 windows, thus supporting the theory of textons of Julesz. It also underlines the simplicity of our method, by showing that convolutions with kernels as small as 3×1 are sufficient for high discrimination.

6.4.2.5 Segmentation performance

Fig. 6.8 and Fig. 6.9 show 256×256 images composed by different Brodatz and VisTex textures and corresponding segmentation results, obtained by Method I, for several values of I. These segmentation results were obtained for a patch size of 27×27 and show that, for the Brodatz database, misclassification errors are concentrated mostly on the border areas. Since the repetitive patterns of the textures in the

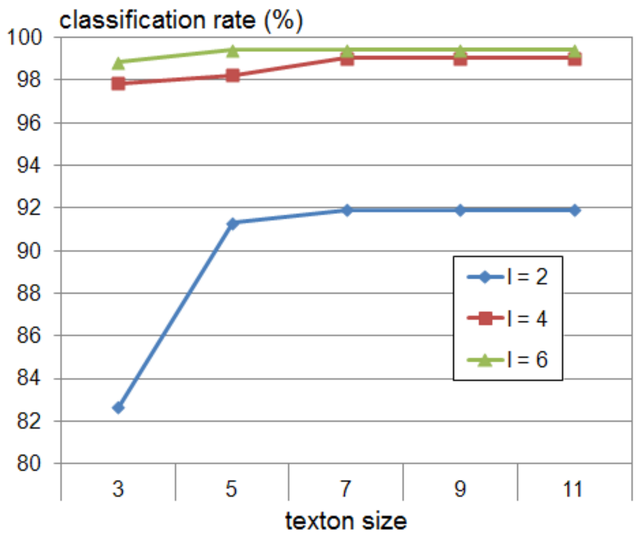


Figure 6.7: Influence of the texton patch size on the correct classification rate of Method I, for the Brodatz database

Table 6.2: Filter coefficients $\boldsymbol{W}$ resulting from the training step (see Section 6.4.1), for Method I, with $I = 2$ and $I = 4$

I	$\boldsymbol{W}$							
1	$\boldsymbol{W}_1 =$	0.943558335	−0.233047068	−0.305871248	−0.5472932	−3.62117529	1.29731488	2.3787837
2	$\boldsymbol{W}_1 =$	0.7133761	−1.53790593	1.42311716	−0.413867027	0.050681863	0.09082524	−0.3170155
	$\boldsymbol{W}_2 =$	−0.217909262	−0.102130122	−0.8199126	−0.344204336	−0.1386514	−0.107457839	−1.08223581

VisTex database are larger, more classification errors occur in Fig. 6.9.

6.4.3 Filter analysis

Table 6.2 shows the filter coefficients that were obtained in Section 6.4.1 for Method I, with $I = 2$ and $I = 4$, using the Brodatz database.

The frequency response of filter $\boldsymbol{W}_1$, for Method I, with $I = 2$, is plotted in Fig. 6.10 (top). It shows that $\boldsymbol{W}_1$ is essentially a wide band-pass filter that attenuates the lowest and highest frequencies. This suggests that textures are identified simply by analyzing the statistics of the transition magnitudes. The low frequency attenuation indicates that the mean value of each texton is not useful for texture discrimination. Because the textures in the Brodatz album are not very sharp, the high-frequency attenuation is probably meant to reduce the effect of noise. Fig. 6.10 (bottom) plots the frequency response of filters $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{W}_3$, for Method I, with $I = 6$. It shows a finer (and redundant) distribution of the frequency spectrum by each filter, which explains the increased accuracy as I increases.

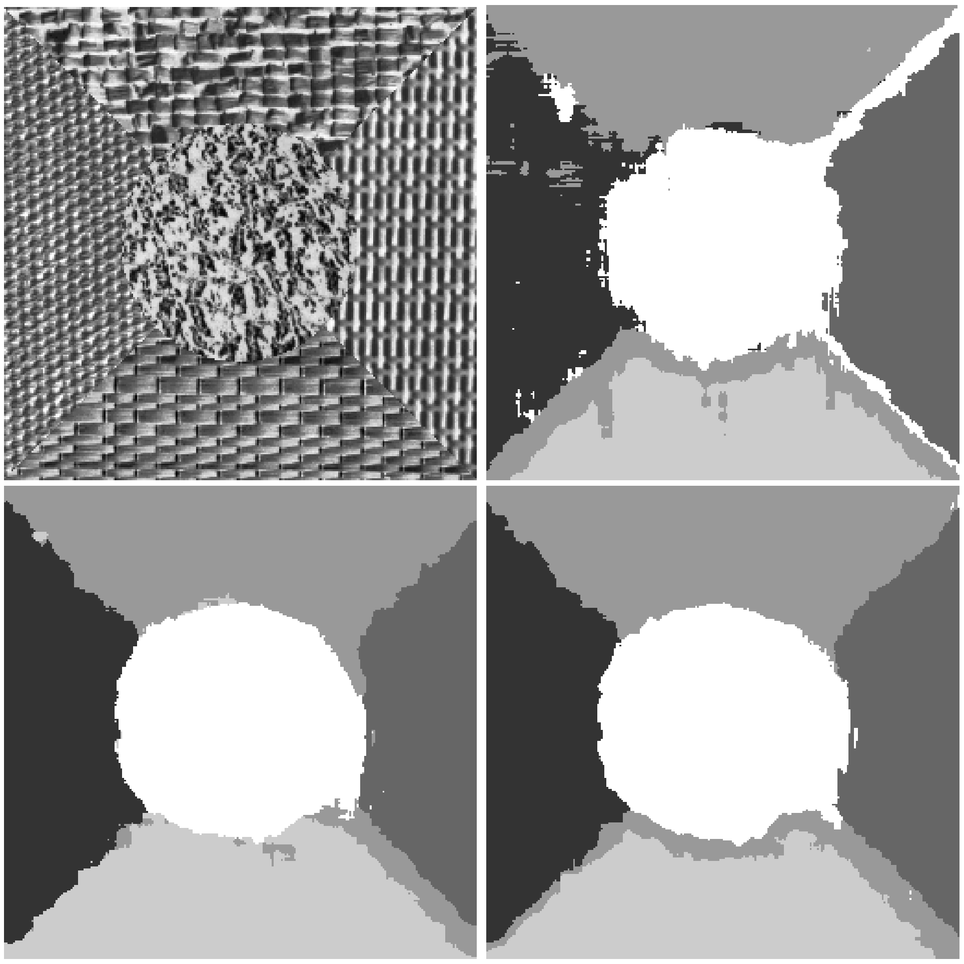


Figure 6.8: Union of Brodatz textures (top left) and segmentation results using Method I, with $I = 2$ (top right), $I = 4$ (bottom left), $I = 6$ (bottom right)

6.5 Discussion and conclusions

This chapter presents a computational simple and highly discriminant approach to texture discrimination. It is based on optimized filters and its general methodology is particularized to rotationally discriminant and rotationally invariant classifiers. The performance of our method exceeds current optimized filter approaches in multi-texture scenarios or when rotation invariance is needed. The discriminative ability of our method is comparable or superior to state-of-the-art methods such as GLCM, GFB or LBP. We believe that the disadvantage of requiring a supervised learning stage, is far outweighted by these advantages. In fact, the computational complexity of most state-of-the-art methods is at least an order of magnitude greater than ours.

Similarly to filter bank approaches, and unlike LBP or GLCM, our method has the advantage of having a very simple and intuitive foundation – it simply consists of a number of linear convolutions followed by statistical measures. This also suggests

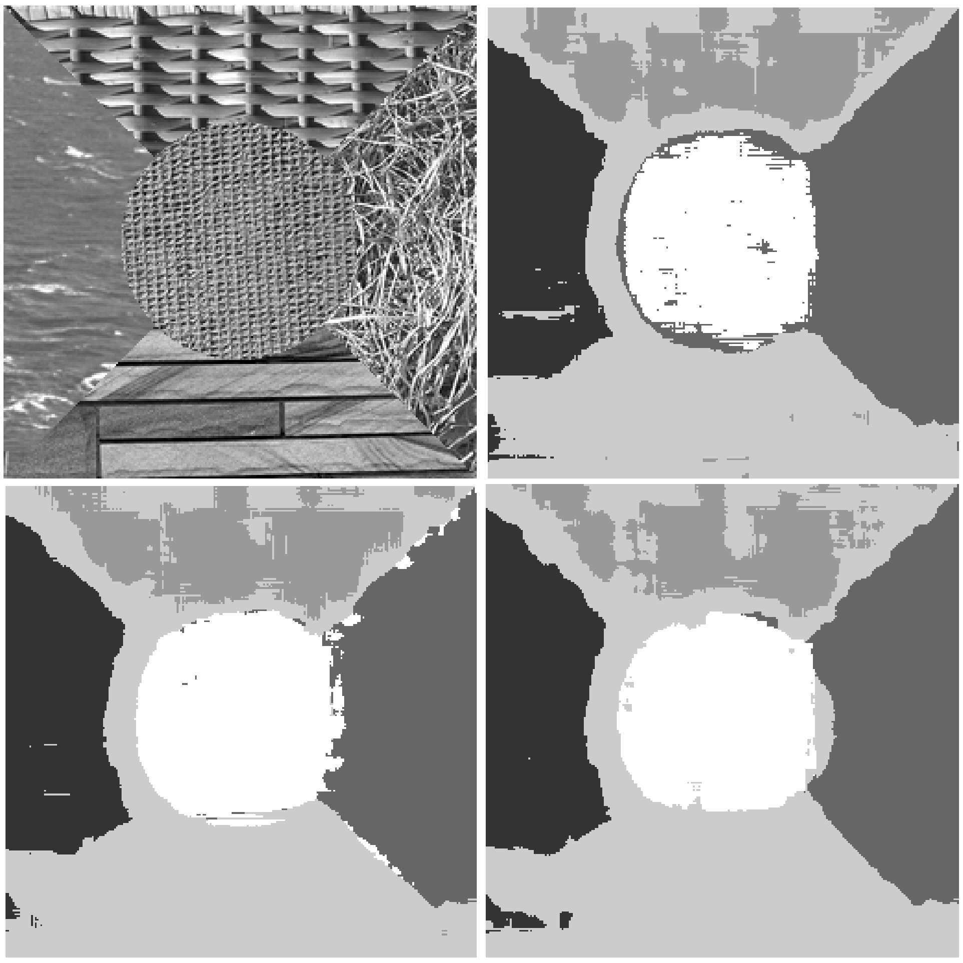
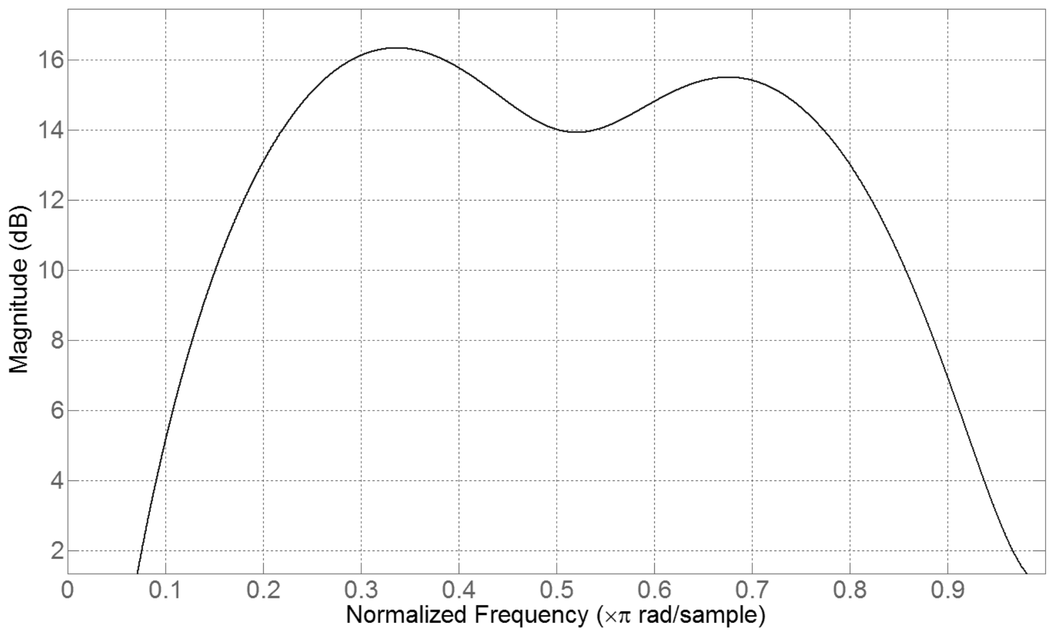


Figure 6.9: Union of VisTex textures (top left) and segmentation results using Method I, with $I = 2$ (top right), $I = 4$ (bottom left), $I = 6$ (bottom right)

that it would be easy to adapt the fitness function to application-specific scenarios, *e.g.*, classes with different a priori likelihoods, and learn optimal filters for that effect. In standard methods, such adaptations are not straightforward.



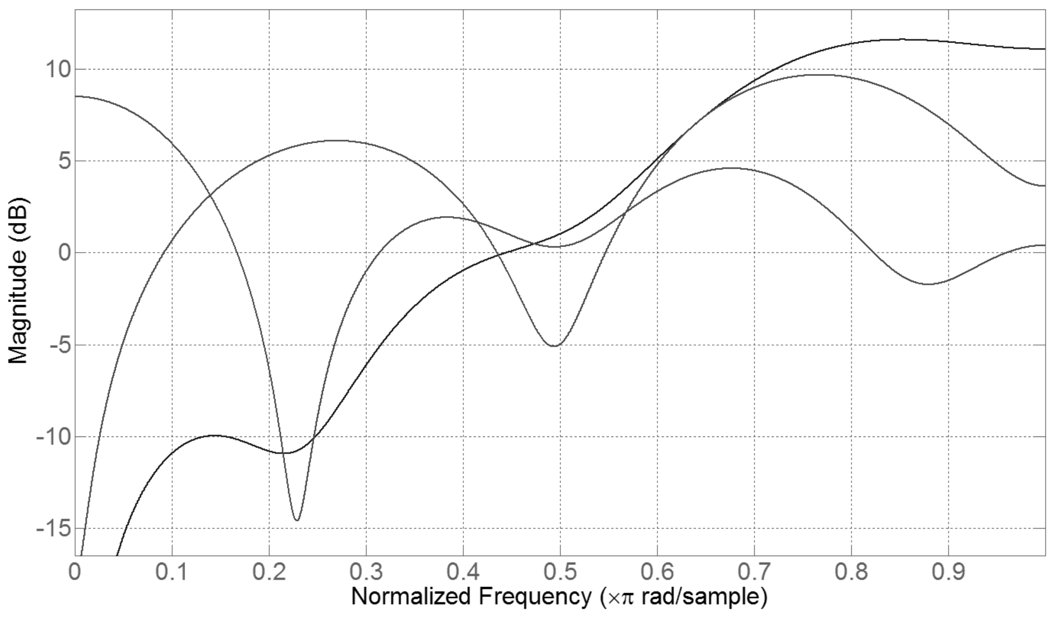


Figure 6.10: Frequency response of filters $\boldsymbol{W}_i$ of Method I, with $I = 2$ (top) and $I = 6$ (bottom), obtained using the Brodatz database

Chapter 7 Combination of all methods

7.1 Proposed approach

Although the merit of each individual method was shown in the previous chapters and are the main focus of this thesis, we provide a brief illustration that the combination of the low level methods dealt within this thesis enable better image understanding of complex images. We do not claim that the techniques used in this chapter are novel or even comparable to the state-of-the-art — we are merely showing an example that combines edge detection, line segment and texture detection.

We chose to combine our methods by first detecting regions that contain textures, since the edges and line segments detected in them are erroneous. Edge detection and line segment extraction is then applied to the remaining area. By having fewer erroneous line segments, the task of image understanding becomes simpler.

Since the method in chapter 6 deals with texture discrimination, *i.e.*, the ability to distinguish between various types of textures, and not with texture detection, we modify it slightly for this goal. In chapter 6, the classes are known beforehand and therefore it is possible that, once a candidate set of filter coefficients is processed by the Genetic Algorithm, class centroids and the covariance matrices (to be used in the weighted Euclidean distance, inside the cost function in (6.8)), can be computed for every class. If the classes (and even the number of classes) in an image are not known beforehand, it is preferable to not depend on specific covariance matrices but, instead, to have a covariance matrix that is applicable to all classes. In this scenario, we simply assign weights to the moments of the filter convolution outputs in the classification task. We proceed simply by replacing class-specific covariance matrices in equation (6.8) with a constant covariance matrix,

$$\begin{aligned}\hat{\boldsymbol{W}} &= \arg\max_{\boldsymbol{W}} \sum_{n=1}^{N} p\left(k_n | \boldsymbol{\phi}\left(\boldsymbol{T}_n\right)\right) \\ &\simeq \arg\min_{\boldsymbol{W}} \sum_{n=1}^{N} \left\| \boldsymbol{\phi}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{k_n} \right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}} . \end{aligned} \tag{7.1}$$

In chapter 6, the class centroids and covariance matrices were computed analytically using equation 6.10. To obtain a generic covariance matrix, various alternatives are possible. Firstly, we can obtain covariance matrices for each class in the training set, as before, and then combine them using, *e.g.*, their average or their median values. In this chapter, we aggregate the estimation of the covariance matrices to the estimation of the filter coefficients by the Genetic Algorithm,

$$\begin{aligned}\{\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}\} &= \arg \max_{\boldsymbol{W},\boldsymbol{\Sigma}} \sum_{n=1}^{N} p\left(k_n | \boldsymbol{\phi}\left(\boldsymbol{T}_n\right)\right) \\ &\simeq \arg \min_{\boldsymbol{W},\boldsymbol{\Sigma}} \sum_{n=1}^{N} \left\|\boldsymbol{\phi}(\boldsymbol{T}_n) - \hat{\boldsymbol{\mu}}_{k_n}\right\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}} . \end{aligned} \tag{7.2}$$

Because we make the simplifying approximation that the covariance matrices are diagonal matrices, only a few extra parameters need to be estimated. We solve the minimization problem (7.2) using the Genetic Algorithm detailed in section 6.3, using the extra restriction that the values of the diagonal of $\hat{\boldsymbol{\Sigma}}$ are non-negative.

After obtaining parameters $\{\boldsymbol{W}, \boldsymbol{\Sigma}\}$, we detect textured regions by first computing the magnitude of the gradient of the texture discrimination function, $\boldsymbol{\phi}(\cdot)$, using the weighted Euclidean distance,

$$\begin{aligned}\frac{\partial \boldsymbol{\phi}(\boldsymbol{T}, x, y)}{\partial x} &= \|\boldsymbol{\phi}(\boldsymbol{T}, x+1, y) - \boldsymbol{\phi}(\boldsymbol{T}, x, y)\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}} \\ \frac{\partial \boldsymbol{\phi}(\boldsymbol{T}, x, y)}{\partial y} &= \|\boldsymbol{\phi}(\boldsymbol{T}, x, y+1) - \boldsymbol{\phi}(\boldsymbol{T}, x, y)\|^2_{\hat{\boldsymbol{\Sigma}}^{-1}} \\ |\nabla \boldsymbol{\phi}(\boldsymbol{T})|_{(x,y)} &= \sqrt{\frac{\partial \boldsymbol{\phi}(\boldsymbol{T}, x, y)^2}{\partial x} + \frac{\partial \boldsymbol{\phi}(\boldsymbol{T}, x, y)^2}{\partial y}}. \end{aligned} \tag{7.3}$$

Then, we downscale the resulting *texture gradient* magnitude, $|\nabla \boldsymbol{\phi}(\boldsymbol{T})|$, by 8 in both directions to obtain the gradient of a more representative area. We define that the points below a threshold constitute a texture region and apply dilation and erosion to obtain a region without gaps. Finally, we upscale the image to the original size using nearest neighbor interpolation, to be used as a mask for the original image.

7.2 Experiments

We first obtain the coefficients that minimize the function (7.2), using, as before, supervised learning with a Genetic Algorithm. We use, as in section 6.4.1, a training set of $N = 1025$ texture patches from $C = 41$ classes, where each class corresponds to an image taken from the Brodatz database, from which 25 non-overlapping patches are extracted. The mean value of each image is set to a constant value, and we use patches of size $p \times q = 60 \times 60$ and texton patches of size $a \times a = 7 \times 7$. The parameters of the Genetic Algorithm are the same as in section 6.4.1.

Fig. 7.1 shows the results that are obtained with the combination of the methods proposed in this thesis on the image first shown in Fig. 1.1. The image on the top left shows the original image and, the top right, the estimated texture areas, according to the method described above. The bottom left image shows the result that would have been obtained if the method detailed in chapter 5 (which uses the TV parametric edge detection method we detail in chapter 3) had been applied directly to the original image. But because some of the areas have already been identified as belonging to textured areas and the line segments in those areas are erroneous, the line segment detector is applied only to the texture-free areas. The result is shown on the bottom right image (the textured area in indicated in gray, for completeness).

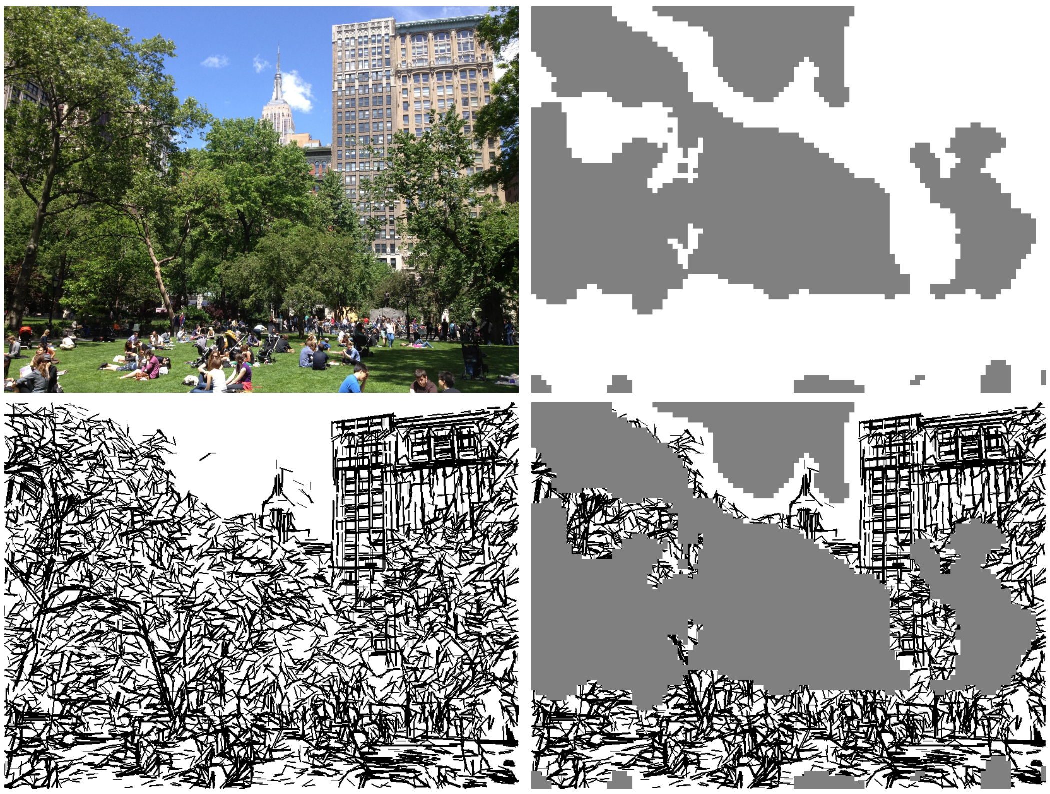


Figure 7.1: Using texture detection to eliminate erroneous line segments. Top left: image; Top right: detected texture region (in gray); Bottom left: line segments extracted using the line segment extractor of chapter 5; Bottom right: image with texture region (in gray) and line segments outside the texture regions.

7.3 Discussion and conclusions

The chapter provides a very simple illustration that the combination of low level methods such as those dealt within this thesis enable better image understanding of complex images.

Chapter 8
Conclusion and future work

8.1 Conclusion

The high availability of image and video capturing devices created many image processing applications such as autonomous vehicles [Urmson 2007] and surveillance [Thida 2013]. Because these applications occur in unconstrained scenarios, the input images are often complex, *i.e.*, they contain many edges, line segments, textures, etc. Because these applications often rely on low-level methods such as, *e.g.*, edge detection, line segment extraction, and multiple texture discrimination, these methods should cope with complex images. In this thesis, we have shown that many of the popular techniques are not suitable for dealing with complex images.

Regarding **edge detection**, we have shown that the Canny edge detector [Canny 1986] does not have a suitable strategy for dealing with large amounts of noise or clutter (*i.e.*, densely concentrated details on the image), missing the detection of many real edges and returning false edges due to noise. We also showed that, although the Statistical edge detector [Bovik 1986] can deal with noisy images due to its statistical framework, it cannot cope with cluttered real images.

We proposed two edge detectors that combine what we denoted as *contextual edges*, obtained with large footprints that deal well with noise; and *local edges*, obtained with small footprints that enable good localization, to achieve an edge detector with both qualities, as idealized by Canny. The contextual edges are obtained by using two-sample tests, either the non-paired Total Variation (TV) parametric test or the paired nonparametric sign test. We confirmed experimentally that our edge detectors can deal successfully with images with large amounts of noise and clutter, combining robustness to noise and good localization.

Regarding **line segment extraction**, we have shown that the Hough transform (HT) [Hough 1962, Duda 1972] cannot detect lines reliably when the image has many edge points, as is often the case for complex images; and also that the typical adaptation of the HT to deal with line segments does away with many of the attractive features of the HT. On the other hand, local methods such as the popular Line Segment Detector (LSD) [von Gioi 2010] also lack robustness to deal with complex images, originating many interrupted segments.

We proposed two methods for extracting line segments. The first, which we call STRAIGHT, added connectivity to the voting process of the Hough Transform, inheriting the global accuracy of the HT and overcoming its limitations in the extraction of line segments. The second is a semi-local method that uses our edge detector in the standard framework of local methods, resulting in a computationally simple method. Our experiments show that both of our methods outperform popular ones when dealing with complex images.

Regarding **multi-texture discrimination**, we have shown that Gray-Level Co-occurrence Matrices [Haralick 1973] are computationally expensive and have low discrimination rates; that the popular Gabor Filter Banks approaches [Jain 1991, Malik 2001] have high classification rates but are computationally very complex; and that Local Binary Patterns [Ojala 2002] has high discrimination rate but is not simple to compute. This illustrates that current methods for multi-texture discrimination are either not very discriminant or are computationally too complex for several real-life applications.

We proposed a method that takes inspiration from Gabor Filter Bank approaches but replaces its computationally expensive elements with simple learned alternatives. The result is a method that is two to three orders of magnitude faster than most multi-texture discrimination methods and with a discrimination rate that is comparable or superior to state-of-the-art methods. We believe that the disadvantage of requiring a supervised learning stage, is far outweighted by these advantages.

8.2 Future work

This thesis introduced methods that can cope with unconstrained real images. However, this work also raised new questions that could be investigated in the future.

Regarding edge detection, we concluded that the use of elongated footprints enabled the handling of noise and clutter. However, because elongated footprints have narrow angular responses, the processing has to be repeated separately for many directions, to cover the entire range of angles. This means that, in practice, the values of the pixels in the image have to be used many times, which, naturally, leads to computational complexity and memory transfer bandwidth issues. It would be useful to not process the several directions independently but to, somehow, do so in an integrated way, to optimize the use of pixel value accesses.

Because the footprint that was used to determine contextual edges consists of two line segments that are steered to the desired angle, the edge detector is only effective in finding edges that are aligned along line segments or curves with large curvature radii. It would be useful to extend the design of footprints to include curved segments, spanning segments that are curved to the left to segments that are curved to the right, to enable the detection of faster curving contours. Because, in this scenario, all directions and curvatures would have to be tested, the number of times that each pixel is accessed would increase, leading to further computational complexity. This could be made practical if, as argued above, the information could be obtained using fewer pixel accesses.

In this thesis, the size of the footprints for determining contextual edges has been set to a constant $M = 15$ pixels. As detailed earlier, this value was set experimentally, as the one that achieves the best compromise between a good handling of noise and clutter and computational complexity, for the images that were used in this thesis. A method to determine the ideal size of such footprints automatically could be useful. This size could be estimated as a constant for the entire image but, perhaps more interestingly, this could occur locally, depending on local estimates of what is needed to achieve a *clear* result. It is possible to imagine a process in which only a few pixel values are analyzed (*e.g.*, three) and, if it is already clear that there is a contextual edge at that location (with some sufficient degree of statistical certainty), no more pixels would be analyzed. And if the information was deemed insufficient to reach a conclusion, more pixels would be taken into account.

In this thesis and in all the suggestions above, the footprint that is used to determine if there is a contextual edge is narrow. A possible way to improve detections could be to analyze wider areas, *i.e.*, to have wider footprints, but still enabling the detection of cluttered areas. The sheer fact that more pixels would be analyzed enhances the contextual data that is taken into account when making decisions, which would also be important in locating the start and end positions of contours.

Regarding line segment extraction, this thesis extended the ideas behind the Hough Transform to include connectivity. Although this method was shown to have good performance, the notion of including connectivity in the voting process of the Hough Transform should receive further attention. In particular, to create methods that are computationally simpler.

Bibliography

[Ade 1983] F Ade. *Characterization of textures by Eigenfilters.* Signal Processing, vol. 5, no. 5, pages 451–457, 1983. 23

[Arras 1997] K. Arras and R. Siegwart. *Feature Extraction and Scene Interpretation for Map-Based Navigation and Map Building.* In Proceedings of SPIE, Mobile Robotics XII, volume 3210, pages 42–53, 1997. 20

[Attneave 1954] F. Attneave. *Some informational aspects of visual perception.* Psychological Review, vol. 61, no. 3, pages 183–193, May 1954. 2, 7

[Aurnhammer 2007] M. Aurnhammer. *Evolving Texture Features by Genetic Programming.* pages 351–358. 2007. 85

[Ballard 1982] D. H. Ballard and C. M. Brown. Computer vision. Prentice Hall Professional Technical Reference, 1st édition, 1982. 15

[Bengio 2009] Y. Bengio. *Learning Deep Architectures for AI.* Foundations and Trends in Machine Learning, vol. 2, no. 1, pages 1–127, January 2009. 1

[Bertero 1988] M. Bertero, T. Pogcio and V. Torre. *Ill-posed problems in early vision.* In Proceedings of the IEEE, pages 869–889, 1988. 18

[Borkar 2011] A. Borkar, M. Hayes and M. Smith. *Polar randomized Hough Transform for lane detection using loose contraints of parallel lines.* In Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, pages 1037–1040, Prague, Czech Republic, 2011. 14

[Bovik 1986] A. Bovik, T. Huang and D. Munson. *Nonparametric tests for edge detection in noise.* Pattern Recognition, vol. 19, no. 3, pages 209–219, April 1986. 8, 13, 36, 37, 38, 101

[Bovik 1991] A. Bovik. *Analysis of multichannel narrow-band filters for image texture segmentation.* IEEE Transactions on Signal Processing, vol. 39, no. 9, pages 2025–2043, 1991. 24

[Brodatz 1966] P. Brodatz. Textures: A photographic album for artists and designers. Dover Publications, NY, 1966. 5, 77, 85

[Burns 1986] J. Burns, A. Hanson and E. Riseman. *Extracting straight lines.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, pages 425–455, 1986. 19, 20, 35

[Canny 1986] J. Canny. *A Computational Approach To Edge Detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, pages 679–698, 1986. 2, 3, 7, 9, 10, 18, 28, 36, 38, 39, 56, 67, 68, 101

[Chetverikov 2000] D. Chetverikov and Z. Foldvári. *Affine-invariant texture classification using regularity features.* In Texture Analysis in Machine Vision, Series in Machine Perception and Artificial Intelligence, pages 69–88, River Edge, NJ, USA, 2000. World Scientific Publishing Co., Inc. 22

[Coggins 1985] J. Coggins and A. Jain. *A spatial filtering approach to texture analysis.* Pattern Recognition Letters, vol. 3, no. 3, pages 195 – 203, 1985. 22

[Dahyot 2009] R. Dahyot. *Statistical Hough Transform.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 8, pages 1502–1509, August 2009. 35, 48, 54

[Dana 1999] K. Dana, B. Van-Ginneken, S. Nayar and J. Koenderink. *Reflectance and Texture of Real World Surfaces.* ACM Transactions on Graphics, vol. 18, no. 1, pages 1–34, 1999. 85

[DasGupta 2010] A. DasGupta. Asymptotic theory of statistics and probability. Springer, 2010. 31

[Desolneux 2006] A. Desolneux, L. Moisan and J. Morel. Gestalt theory and image analysis, a probabilistic approach. February 2006. 19, 20

[Du 2010] S. Du, B. van Wyk, C. Tu and X. Zhang. *An improved Hough transform neighborhood map for straight line segments.* IEEE Transactions on Image Processing, vol. 19, pages 573–585, 2010. 14

[Duda 1972] R. Duda and P. Hart. *Use of the Hough Transformation to detect lines and curves in pictures.* Communications of the ACM, vol. 15, no. 1, pages 11–15, 1972. 2, 4, 14, 15, 17, 44, 48, 55, 56, 60, 65, 66, 68, 69, 70, 101

[Dunn 1995] D. Dunn and W. Higgins. *Optimal Gabor filters for texture segmentation.* IEEE Transactions on Image Processing, vol. 4, no. 7, pages 947–964, 1995. 24

[Duprat 2005] O. Duprat, B. Keck, C. Ruwwe and U. Zölzer. *Hough Transform with weighting Edge-maps.* In Fifth IASTED International Conference on Visualization, Imaging, & Image Processing, Benidorm, Spain, September 2005. 18

[Etemadi 1992] A. Etemadi. *Robust Segmentation of Edge Data.* In Proceedings of IEEE International Conference on Image Processing and its Applications, pages 311–314, Maastricht, The Netherlands, 1992. 20

[Faugeras 1992] O. Faugeras, R. Deriche, H. Mathieu, N. Ayache and G. Randall. *Parallel image processing.* chapitre The depth and motion analysis machine, pages 143–175. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1992. 20

[Fesharaki 1994] M. Fesharaki and G. Hellestrand. *A new edge detection algorithm based on a statistical approach.* In Proceedings of International Symposium on Speech, Image Processing and Neural Networks, volume 1, pages 21 – 24, Hong Kong, China, April 1994. 8, 14

[Fischler 1981] M. Fischler and R. Bolles. *RANdom SAmple Consensus: a paradigm for model fitting with applications to image analysis and automated cartography.* Communications of the ACM, vol. 24, pages 381–395, 1981. 19

[Fränti 1998] P. Fränti, E. Ageenko, H. Kälviäinen and S. Kukkonen. *Compression of line drawing images using Hough transform for exploiting global dependencies.* In Proceedings of the Fourth Joint Conference on Information Sciences, volume 4, pages 433–436, Research Triangle Park, North Carolina, USA, 1998. 14

[Freeman 1991] W. Freeman and E. Adelson. *The Design and Use of Steerable Filters.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, pages 891–906, 1991. 12, 28

[Galun 2007] M. Galun, R. Basri and A. Brandt. *Multiscale Edge Detection and Fiber Enhancement Using Differences of Oriented Means.* IEEE International Conference on Computer Vision, vol. 0, pages 1–8, 2007. 8, 12

[Gebäck 2009] T. Gebäck and P. Koumoutsakos. *Edge detection in microscopy images using curvelets.* BMC Bioinformatics, vol. 10, no. 1, 2009. 8, 12

[Goldenshluger 2004] A. Goldenshluger and A. Zeevi. *The Hough Transform estimator.* Annals of statistics, vol. 32, page 1908, 2004. 2, 15, 18

[Guerreiro 2010] R. Guerreiro and P. Aguiar. *Learning simple texture discrimination filters.* In Proceedings of IEEE International Conference on Image Processing, pages 261–264, 2010. 4, 77

[Guerreiro 2011] R. Guerreiro and P. Aguiar. *Incremental local Hough Transform for line segment extraction.* In Proceedings of IEEE International Conference on Image Processing, Brussels, Belgium, September 2011. 4, 43

[Guerreiro 2012] R. Guerreiro and P. Aguiar. *Connectivity-Enforcing Hough Transform for the Robust Extraction of Line Segments.* IEEE Transactions on Image Processing, 2012. 4, 43, 65, 66, 68, 69, 70, 71, 72

[Guerreiro 2013a] R. Guerreiro and P. Aguiar. *Extraction of line segments in cluttered images via multiscale edges.* In Proceedings of IEEE International Conference on Image Processing, pages 3045–3048, 2013. 4, 65

[Guerreiro 2013b] R. Guerreiro and P. Aguiar. *Optimized filters for efficient multi-texture discrimination.* Pattern Analysis and Applications, 2013. 4, 77

[Guil 1995] N Guil, J Villalba and E Zapata. *A fast Hough Transform for segment detection.* IEEE Transactions on Image Processing, vol. 4, no. 11, pages 1541–1548, 1995. 17

[Guru 2004] D. Guru. *A simple and robust line detection algorithm based on small eigenvalue analysis.* Pattern Recognition Letters, vol. 25, no. 1, pages 1–13, 2004. 17, 19, 20

[Haralick 1973] R. Haralick, I. Dinstein and K. Shanmugam. *Textural features for image classification.* IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, pages 610–621, 1973. 3, 21, 102

[Hartley 2004] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, second édition, 2004. 19

[Hough 1962] P. Hough. *Method and means for recognizing complex patterns.* U.S. Patent 3.069.654, December 1962. 2, 14, 101

[Husoy 1993] J. Husoy, T. Randen and T. Gulsrud. *Image texture classification with digital filter banks and transforms.* In Proc., SPIE International Symposium on Optical Applied Science and Engineering, pages 260–271, San Diego, California, July 1993. 22

[Illingworth 1987a] J. Illingworth and J. Kittler. *The adaptive Hough Transform.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, pages 690–698, 1987. 16, 48

[Illingworth 1987b] J Illingworth and J Kittler. *A survey of efficient Hough Transform methods.* In Proceedings of Alvery Vision Club Conference, pages 319–326, Cambridge, 1987. 16, 17, 35

[Jain 1991] A. K. Jain and F. Farrokhnia. *Unsupervised texture segmentation using Gabor filters.* Pattern Recognition, vol. 24, no. 12, pages 1167–1186, 1991. 3, 21, 22, 102

[Jain 1996] A. Jain and K. Karu. *Learning Texture Discrimination Masks.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, pages 195–205, February 1996. 24

[Jang 2001] J. Jang and K. Hong. *Linear band detection based on the Euclidean distance transform and a new line segment extraction method.* Pattern Recognition, vol. 34, no. 9, pages 1751 – 1764, 2001. 18

[Ji 2011] J. Ji, G. Chen and L. Sun. *A novel Hough Transform method for line detection by enhancing accumulator array.* Pattern Recognition Letters, vol. 32, no. 11, pages 1503–1510, August 2011. 14, 18

[Julesz 1981] B. Julesz. *Textons, the elements of texture perception, and their interactions.* Nature, vol. 290, 1981. 78

[Kälviäinen 1995] H. Kälviäinen, P. Hirvonen, L. Xu and E. Oja. *Probabilistic and non-probabilistic Hough Transforms: overview and comparisons.* Image and Vision Computing, vol. 13, pages 239–252, 1995. 16, 17, 55

[Kamat-Sadekar 1998] V. Kamat-Sadekar and S. Ganesan. *Complete description of multiple line segments using the Hough Transform.* Image and Vision Computing, vol. 16, no. 9-10, pages 597 – 613, 1998. 18

[Kim 1998] J. Kim and R. Krishnapuram. *A robust Hough Transform based on validity.* In IEEE International Conference on Fuzzy Systems, volume 2, pages 1530 – 1535, Anchorage, USA, May 1998. 16, 17

[Kosecka 2002] J. Kosecka and W. Zhang. *Video Compass.* In Proceedings of European Conference on Computer Vision, pages 657–673, Copenhagen, Denmark, 2002. Springer-Verlag. 14

[Krüger 2001] W. Krüger. *Robust and efficient map-to-image registration with line segments.* Machine Vision and Applications, vol. 13, pages 38–50, 2001. 14

[Lam 2008] B. Lam, V. Ciesielski and A. Song. *Visual Texture Classification and Segmentation by Genetic Programming.* In Genetic and Evolutionary Computation for Image Processing and Analysis. Hindawi Pub. Co., 2008. 84

[Laws 1980] K. Laws. *Texture image segmentation.* Ph.d. dissertation, Image Processing Institute, University of Southern California, 1980. 22

[Lee 2006] Y. Lee, H. Koo and C. Jeong. *A straight line detection using Principal Component Analysis.* Pattern Recognition Letters, vol. 27, pages 1744–1754, 2006. 17

[Li 1986] H. Li, M. Lavin and R. Le Master. *Fast Hough Transform: A hierarchical approach.* Computer Vision, Graphics, and Image Processing, vol. 36, pages 139–161, 1986. 16, 48

[Lim 2002] D. Lim and S. Jang. *Comparison of two-sample tests for edge detection in noisy images.* Journal of the Royal Statistical Society: Series D (The Statistician), vol. 51, no. 1, pages 21–30, 2002. 14

[Lindeberg 1996] T. Lindeberg. *Edge Detection and Ridge Detection with Automatic Scale Selection.* International Journal of Computer Vision, vol. 30, pages 465–470, 1996. 2, 7

[Liu 2011] J. Liu, Y. Chen and X. Tang. *Decomposition of Complex Line Drawings with Hidden Lines for 3D Planar-Faced Manifold Object Reconstruction.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, pages 3–15, 2011. 14

[Mahalanobis 1994] A. Mahalanobis and H. Singh. *Application of correlation filters for texture recognition*. Applied Optics, vol. 33, no. 11, pages 2173–2179, Apr 1994. 23

[Malik 2001] J. Malik, S. Belongie, T. Leung and J. Shi. *Contour and Texture Analysis for Image Segmentation*. International Journal of Computer Vision, vol. 43, no. 1, pages 7–27, 2001. 3, 5, 21, 22, 77, 82, 102

[Martin 2004] D. Martin, C. Fowlkes and J. Malik. *Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 5, pages 530–549, May 2004. 8

[Micusík 2008] B. Micusík, H. Wildenauer and J. Kosecka. *Detection and matching of rectilinear structures*. In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pages 1–7, Anchorage, Alaska, USA, 2008. 14

[Mirmehdi 2008] M. Mirmehdi, X. Xie and J. Suri. Handbook of texture analysis. Imperial College Press, UK, 2008. 21, 22

[Mitchell 1996] M. Mitchell. An introduction to genetic algorithms. MIT Press, Cambridge, MA, USA, 1996. 84

[Nevatia 1980] R. Nevatia and K. Babu. *Linear feature extraction and description*. Computer Graphics and Image Processing, vol. 13, no. 3, pages 257 – 269, 1980. 19

[Nguyen 2007] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis and R. Siegwart. *A comparison of line extraction algorithms using 2D range data for indoor mobile robotics*. Autonomous Robots, vol. 23, pages 97–111, 2007. 19, 20

[Ojala 2002] T. Ojala, M. Pietikäinen and T. Mäenpää. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pages 971–987, 2002. 3, 21, 80, 83, 102

[Oskoei 2010] M. Asghari Oskoei and H. Hu. *A Survey on Edge Detection Methods*. Rapport technique ISSN 1744 - 8050, School of Computer Science & Electronic Engineering, University of Essex, Colchester, United Kingdom, February 2010. 7, 10

[Partio 2007] M. Partio, B. Cramariuc and M. Gabbouj. *An ordinal co-occurrence matrix framework for texture retrieval*. Journal on Image and Video Processing, 2007. 3, 21, 87, 88, 89

[Perona 1990] P. Perona and J. Malik. *Scale-space and edge detection using anisotropic diffusion*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, pages 629–639, 1990. 7

[Poli 2008] R. Poli. *Analysis of the publications on the applications of particle swarm optimisation*. Journal of Artificial Evolution and Applications, vol. 2008, pages 4:1–4:10, jan 2008. 84

[Randen 1997] T. Randen. *Filter and Filter Bank Design for Image Texture Recognition*. Ph.d. dissertation, Stavanger College, Norwegian University of Science and Technology, 1997. 23, 24

[Schmid 1997] C. Schmid and A. Zisserman. *Automatic Line Matching across Views*. In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pages 666–671, San Juan, Puerto Rico, 1997. 14

[Sheskin 2007] D. Sheskin. Handbook of parametric and nonparametric statistical procedures. Chapman & Hall/CRC, 4 édition, 2007. 4, 13, 30

[Slater 1996] D. Slater and G. Healey. *3D Shape Reconstruction by Using Vanishing Points*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, pages 211–217, 1996. 14

[Song 2003] A. Song and V. Ciesielski. *Fast texture segmentation using genetic programming*. In Proceedings of the Congress on Evolutionary Computation, pages 2126–2133, Canberra, 8-12 December 2003. IEEE Press. 84

[Song 2005] J. Song and M. Lyu. *A Hough Transform based line recognition method utilizing both parameter space and image space*. Pattern Recognition, vol. 38, no. 4, pages 539 – 552, 2005. 18

[Stephens 1991] R. Stephens. *Probabilistic approach to the Hough transform*. Image and Vision Computing, vol. 9, no. 1, pages 66 – 71, 1991. 16

[Teutsch 2011] M. Teutsch and T. Schamm. *Fast Line and Object Segmentation in Noisy and Cluttered Environments using Relative Connectivity*. In Proceedings of the Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, USA, July 2011. 18

[Thida 2013] M. Thida, Y. Yong, P. Climent-Pérez, H. Eng and P. Remagnino. *A Literature Review on Video Analytics of Crowded Scenes*. In P. Atrey, M. Kankanhalli and A. Cavallaro, editeurs, Intelligent Multimedia Surveillance, pages 17–36. Springer Berlin Heidelberg, 2013. 1, 101

[Thune 1997] M. Thune, B. Olstad and N. Thune. *Edge detection in noisy data using finite mixture distribution analysis*. Pattern Recognition, vol. 30, no. 5, pages 685 – 699, 1997. 8, 14

[Tivive 2007] F. Tivive and A. Bouzerdoum. *A Nonlinear Feature Extractor for Texture Segmentation*. Proceedings of IEEE International Conference on Image Processing, 2007. 24

[Toyoda 2005] T. Toyoda and O. Hasegawa. *Texture classification using extended higher order local autocorrelation features.* In Texture 2005: 4th international workshop on texture analysis and synthesis, pages 131–136, 2005. 22

[Tüceryan 1990] M. Tüceryan and A. Jain. *Texture Segmentation Using Voronoi Polygons.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 2, pages 211–216, 1990. 22

[Tüceryan 1993] M. Tüceryan and A. Jain. *Texture analysis.* In Handbook of Pattern Recognition & Computer Vision, pages 235–276, River Edge, NJ, USA, 1993. World Scientific Publishing Co., Inc. 21, 22

[Unser 1986] M. Unser. *Local linear transforms for texture measurements.* Signal Processing, vol. 11, pages 61–79, July 1986. 23

[Urmson 2007] C. Urmson, J. Anhalt, J. Bagnell, C. Baker, R., J. Dolan, D. Duggins, D. Ferguson, T. Galatali, H. Geyer, M. Gittleman, S., M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Seo, R. Simmons, S. Singh, J. Snider, A. Stentz, W. Whittaker and J. Ziglar. *Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge.* Rapport technique CMU-RI-TR-, Robotics Institute, http://archive.darpa.mil/grandchallenge/, April 2007. 1, 101

[vis 2002] *VisTex: Vision Texture Database.* Maintained by the Vision and Modeling group at the MIT Media Lab. web page: `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/`, 2002. 5, 77, 85

[von Gioi 2010] R. von Gioi, J. Jakubowicz, J. Morel and G. Randall. *LSD: A Fast Line Segment Detector with a False Detection Control.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 4, pages 722–732, 2010. 2, 4, 14, 17, 19, 20, 35, 44, 55, 56, 57, 58, 59, 60, 65, 66, 68, 69, 70, 71, 73, 101

[Watt 2000] A. Watt. 3d computer graphics. Numeéro vol. 1 de 3D Computer Graphics. Addison-Wesley, 2000. 29

[Weldon 1996] T. Weldon and W. Higgins. *Design Of Multiple Gabor Filters For Texture Segmentation.* In IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 4, pages 2243–2246, Atlanta, USA, May 1996. 24

[Williams 2006] I. Williams, D. Svoboda, N. Bowring and E. Guest. *Improved statistical edge detection through neural networks.* In 10th Conference on Medical Image Understanding and Analysis, pages 56–60, Manchester, UK, July 2006. 8, 14

[Wilson 1979] L. Wilson, N.Y. Chen, R.B. Kelley and J.R. Birk. *Image Feature Extraction Using Diameter Limited Gradient Direction Histograms.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, no. 1, pages 228–235, 1979. 20

[Wilson 2003] R. Wilson and C. Li. *A Class of Discrete Multiresolution Random Fields and Its Application to Image Segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 1, pages 42–56, 2003. 22

[Wu 1991] X. Wu. *An Efficient Antialiasing Technique.* In Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '91, pages 143–152, New York, NY, USA, 1991. ACM. 29

[Xia 2006] Y. Xia, D. Feng and R. Zhao. *Morphology-based multifractal estimation for texture segmentation.* IEEE Transactions on Image Processing, vol. 15, no. 3, pages 614–623, 2006. 22

[Xiao 2003] J. Xiao and M. Shah. *Two-Frame Wide Baseline Matching.* In Proceedings of IEEE International Conference on Computer Vision, pages 603–609, Nice, France, 2003. 20

[Xiaoming 2007] M. Xiaoming, Z. Liang-pei and L. Ping-xiang. *An Approach For Edge Detection Based On Beamlet Transform.* In Fourth International Conference on Image and Graphics, pages 353–357, Aug 2007. 8, 12

[Yalniz 2010] I. Yalniz and S. Aksoy. *Unsupervised detection and localization of structural textures using projection profiles.* Pattern Recognition, vol. 43, no. 10, pages 3324 – 3337, 2010. 22

[Yang 1997] M. Yang, J. Lee, C. Lien and C. Huang. *Hough Transform Modified by Line Connectivity and Line Thickness.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 8, pages 905–910, August 1997. 18

[Yi 2009] S. Yi, D. Labate, G. Easley and H. Krim. *A Shearlet Approach to Edge Analysis and Detection.* IEEE Transactions on Image Processing, vol. 18, no. 5, pages 929–941, May 2009. 8, 12

[Yuen 1991] S. Yuen. *Connective Hough Transform.* In Proceedings of the British Machine Vision Conference, pages 127 – 135, Glasgow, UK, September 1991. 17

[Zhou 2006] J. Zhou, W. Bischof and A. Sanchez-Azofeifa. *Extracting Lines in Noisy Image Using Directional Information.* In Proceedings of IEEE International Conference on Pattern Recognition, volume 2, pages 215–218. IEEE Computer Society, 2006. 16