

國立臺灣大學電機資訊學院電信工程學研究所



碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

具有雜訊穩健性的改進式邊緣及脊偵測演算法
和雙邊濾波器

Improved Edge and Ridge Detection Algorithm and Bilateral
Filter with Robustness to Noise

黃品文

Pin-Wen Huang

指導教授：丁建均 博士

Advisor: Jian-Jiun Ding, Ph.D.

中華民國 113 年 1 月

January , 2024

誌謝



雖然我在研究所的研究歷程不算順遂，直到碩士三年級上學期才完成了學位論文，幸好最終我還是順利地通過了。其中我也受到了相當多的人的幫助，只有我的話是很難順利把這篇論文完成的，所以想在這邊感謝一下與論文有關的人們。

首先我要感謝我的指導教授丁建均教授，除了關於研究方向的指引及建議以外，老師的指導也讓我學到了不少做事和研究的方法，以及在我研究進度落後時也都有給我相關的幫助和鼓勵。此外實驗室以前學長姐提供的資料和教學也幫了我不少忙，比方說很多人在用的論文模板。

接下來我要感謝我的家人們，我的父母一直有給予我情感方面的支持，讓我在稍微低落的時候能繼續堅持下去。也幫我分擔了許多煩惱，讓我能在沒有後顧之憂的情況下努力。另外我的妹妹也給予了我關於文獻搜尋的有用的建議，感謝 Web of Science 和 scopus。

然後也謝謝實驗室兩年半遇到的同學們，還有合租的室友們，還有朋友跟我妹妹，時常在日常生活中給我各種大事小事上的建議和幫助，以及為我的生活帶來笑聲與歡樂。

最後也感謝其他的各種幫助過我的人，以及被我麻煩到的人。我的人生中或多或少都給其他人添了些麻煩，當然這不是什麼好事，但也是感謝這些事我才能走到這裡所以就謝謝各位讓我添麻煩了，在此也表達我的歉意。

文筆不夠真摯就請各位見諒了。

MANDARIN ABSTRACT (中文摘要)



在影像處理領域中，邊緣偵測是一個基礎而重要的問題，因此一直都有受到持續討論。我們觀察到在傳統的邊緣偵測方法中，有一個雖然關鍵卻較少受到關注的子領域：脊偵測；此外有個能在邊緣偵測上派上用場，卻沒那麼常被實際運用到的演算法：雙邊濾波器。

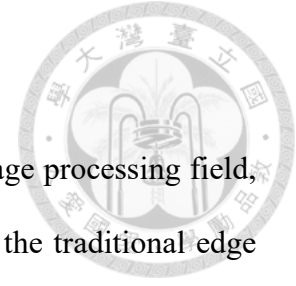
在論文的前半部分中，我們先整理了包含邊緣偵測的三個領域的內容，並簡單介紹了該領域歷年來的發展，以及對該領域內不同的方法按照類別做了總整理。其中也包含了領域中一些方法的介紹和理論說明。

而在後半部分，我們則基於前人提出過的方法，嘗試提出了關於雙邊濾波器以及脊偵測的改進方法。在雙邊濾波器這部分，我們透過影像平滑和線性組合來避免了傳統雙邊濾波器常有的無法保存特徵細節的問題。而在脊偵測這部分，我們則是透過了多尺度的 LoG 偵測器來求出脊特徵在影像特定位置上的大小，再配合另一個脊偵測演算法來求出該位置周圍的變化幅度來判斷是否為脊特徵，藉此也避開了該方法原有的無法選定脊特徵大小的問題。

最後我們則透過實驗去重現前半部分中部分比較重要的方法，以及我們提出的方法在不同圖像上的效果，以此比較我們的方法的實際表現。

關鍵字：邊緣偵測、雙邊濾波器、脊偵測

ABSTRACT



Edge detection is a fundamental and important issue in the image processing field, and thus has been continuously discussed. We observe that among the traditional edge detection methods, there is a sub-domain that has received less attention although it is crucial: ridge detection, and an algorithm that can be useful for edge detection but is not so often used in practice: the bilateral filter.

In the first half of the thesis, we first organize the content of the three areas, and give a brief overview of the development of the area over the years, as well as a summary of the different methods in the area. An introduction and theoretical description of some of the methods in the field are also included.

In the second half of the thesis, we try to present improved methods for bilateral filters and ridge detection, based on the methods proposed by the previous researchers. In the part of bilateral filter, we avoid the problem of not preserving feature details, which is often found in traditional bilateral filters, by image smoothing and linear combination. For ridge detection, we use a multi-scale LoG detector to determine the size of a ridge feature at a specific location in the image, and then use another ridge detection algorithm to detect ridge by variations, which avoids the problem of not being able to select the size of the ridge feature.

Finally, we experimentally reproduce some of the more important methods in the first half of the thesis and compare the performance of our method with the results of our proposed method on different images.

Keyword: edge detection, ridge detection, bilateral filter

CONTENTS



| | |
|---|----------|
| 口試委員會審定書 | # |
| 誌謝 | i |
| MANDARIN ABSTRACT (中文摘要)..... | ii |
| ABSTRACT | iii |
| CONTENTS | iv |
| LIST OF FIGURES | vii |
| LIST OF TABLES | ix |
| Chapter 1 Introduction..... | 1 |
| 1.1 Motivation..... | 1 |
| 1.2 Thesis organization | 1 |
| Chapter 2 Edge detection | 2 |
| 2.1 Edge Detection..... | 2 |
| 2.1.1 Introduction | 2 |
| 2.1.2 Image differentiation | 2 |
| 2.1.3 Thresholding | 4 |
| 2.1.4 Non Maximum Suppression..... | 4 |
| 2.1.5 Edge fixing | 6 |
| 2.2 Gradient and Gaussian based methods | 7 |
| 2.2.1 Derivative convolution kernels | 7 |
| 2.2.2 Gradient calculation | 8 |
| 2.2.3 Image smoothing | 9 |
| 2.2.4 Gaussian blur | 10 |



| | | |
|------------------|---|-----------|
| 2.2.5 | Canny edge detector | 11 |
| 2.2.6 | Laplacian of Gaussian edge detector(LoG)..... | 13 |
| 2.2.7 | Multi-detector and multi-scale fusing | 14 |
| 2.3 | Other hand-crafted methods | 15 |
| 2.3.1 | Color edge methods..... | 15 |
| 2.3.2 | Fuzzy methods | 17 |
| 2.3.3 | Texture methods | 18 |
| 2.3.4 | Wavelet methods | 19 |
| 2.3.5 | Edge-preserving smoothing methods..... | 19 |
| 2.4 | Learning based methods | 20 |
| 2.4.1 | Traditional learning-based method..... | 20 |
| 2.4.2 | Convolutional Neural Network | 20 |
| 2.4.3 | Transformer model | 21 |
| 2.5 | Demonstration of survey methods | 21 |
| Chapter 3 | Ridge filter | 24 |
| 3.1 | Introduce | 24 |
| 3.2 | Undirected filters | 24 |
| 3.3 | Oriented Filters | 24 |
| 3.3.1 | Low-Pass filter based methods..... | 25 |
| 3.3.2 | Gaussian based methods | 26 |
| 3.3.3 | Hilbert transform | 27 |
| 3.4 | Non-filter methods..... | 27 |
| 3.4.1 | Hessian matrix..... | 27 |
| 3.4.2 | Weingarten | 28 |
| 3.4.3 | Separatrices and Drainage patterns | 28 |

| | | |
|------------------|---|-----------|
| 3.4.4 | Divergence | 29 |
| 3.4.5 | Polynomial-fitting methods..... | 30 |
| Chapter 4 | Bilateral filter | 31 |
| 4.1 | Introduce of bilateral filter..... | 31 |
| 4.2 | Improvement of bilateral filter..... | 32 |
| 4.2.1 | Speed Improvement | 32 |
| 4.2.3 | Joint/Cross Bilateral Filter | 33 |
| 4.3 | Other methods..... | 34 |
| 4.3.1 | Anisotropic diffusion..... | 34 |
| 4.3.2 | Guided image filter | 35 |
| Chapter 5 | Proposed Bilateral filter | 37 |
| 5.1 | Algorithm..... | 37 |
| 5.2 | Experiment..... | 38 |
| Chapter 6 | Proposed Ridge detector..... | 39 |
| 6.1 | Algorithm..... | 39 |
| 6.1.1 | Introduction | 39 |
| 6.1.2 | Multiscale directional filter | 40 |
| 6.1.3 | Ridge scoring | 41 |
| 6.1.4 | Local maximum and refinement | 42 |
| 6.2 | Experiment..... | 42 |
| 6.2.1 | Scoring method | 42 |
| Chapter 7 | Conclusion and future work | 48 |
| 7.1 | Conclusion | 48 |
| 7.2 | Future work..... | 48 |
| REFERENCE | | 50 |

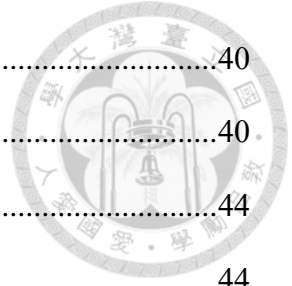


LIST OF FIGURES



| | | |
|-----------|---|----|
| Fig. 2.1 | The partial derivative map..... | 3 |
| Fig. 2.2 | The NMS Diagram | 5 |
| Fig. 2.3 | Demonstration of trihedral junction fixing [5] | 6 |
| Fig. 2.4 | Gradient calculation method by using first order derivatives..... | 9 |
| Fig. 2.5 | The Gaussian kernel | 10 |
| Fig. 2.6 | Derivative of Gaussian(DoG) kernel | 12 |
| Fig. 2.7 | The demonstration of double threshold | 12 |
| Fig. 2.8 | The LoG kernel..... | 13 |
| Fig. 2.9 | 1D-case of edge detect..... | 14 |
| Fig. 2.10 | Edge maps with different scale of DoG..... | 14 |
| Fig. 2.11 | An example of color adjacency in PMI[10] | 17 |
| Fig. 2.12 | An example of fuzzy function [6]..... | 18 |
| Fig. 2.13 | The textons and texture map [9] | 18 |
| Fig. 2.14 | An example of 2D wavelet decomposition. [8]..... | 19 |
| Fig. 3.1 | Ziou, Gouton and Gaussian filters [21] | 25 |
| Fig. 3.2 | The steerable filter [32] | 26 |
| Fig. 3.3 | A demonstration of finding ridge by divergence [31] | 29 |
| Fig. 4.1 | Adaptive bilateral filter based on polynomial fitting [36]..... | 33 |
| Fig. 4.2 | A contrast between bilateral filter and Guided image filter. [37] | 35 |
| Fig. 5.1 | Experiment of two 1D signals | 38 |
| Fig. 5.2 | Experiment of 2D image..... | 38 |
| Fig. 6.1 | Pixels classification in Improved Harris' Algorithm [26]..... | 39 |
| Fig. 6.2 | Flow chart of proposed ridge detection method | 40 |

| | | |
|----------|--|----|
| Fig. 6.3 | The output on ridge with different LoG widths..... | 40 |
| Fig. 6.4 | Relation between LoG widths and center response..... | 40 |
| Fig. 6.5 | Example of GUF1.. .. | 44 |
| Fig. 6.6 | Example of RITE..... | 44 |
| Fig. 6.7 | Example for Sketch..... | 45 |
| Fig. 6.8 | Example of RITE..... | 47 |



LIST OF TABLES



| | | |
|-----------|---|----|
| Table 2.1 | Kernels of derivative..... | 8 |
| Table 2.2 | Experiment outputs of survey edge detection methods | 22 |
| Table 6.1 | Comparison of evaluation scores..... | 45 |
| Table 6.2 | Average running time in dataset GUF1..... | 46 |
| Table 6.3 | Comparison of evaluation scores on RITE..... | 47 |

Chapter 1 Introduction



1.1 Motivation

In recent years, deep learning has improved the performance of edge detection to a level that is difficult to achieve with traditional techniques. However, this has led to the decline of traditional methods based on theoretical interpretation of edges, and we believe that there should still be room for development in this section.

Therefore, we would like to approach the two related domains, ridge detection and bilateral filter, to see if we can make some breakthroughs among the three fields from these two domains, which are not often interacted with edge detection domains these years, and to improve the understanding of the low-level features of images.

1.2 Thesis organization

In Chapters 2 to Chapter 4, we will introduce edge detection, ridge detection, and bilateral filter, respectively. This section is presented as an overview, so the main focus will be on the presentation of the more meaningful techniques in this area, as well as an introduction for some of the more useful or innovative methods and the explanation of their algorithm.

In Chapter 5, we present the improved bilateral filter and ridge detector, and then we will evaluate the effectiveness of these proposed methods through the designed scoring function. And in Chapter 6, we will demonstrate implementations of the edge detection, ridge detection, and bilateral filter algorithms mentioned in the previous three chapters and compare them with the proposed method.

Finally, we will summarize the performance of the proposed method in Chapter 7 and discuss the future work.

Chapter 2 Edge detection



2.1 Edge Detection

2.1.1 Introduction

In computer vision, image feature is the structures that contains some specific characteristics, such as edges, corners, blobs and ridges. Among these features, edges are a low-level feature which has a strong image brightness variation. Because edges are usually boundaries between different regions, this feature can help in other image processing and computer vision domains, like image recognition, face recognition, target tracking, corner detection, image compressing, image segmentation, and others.

Edge detector is the processing algorithm which detect and locate the edge feature from an input digital image, and then outputs an edge map that records information such as location and intensity of the edge features. Although edges are low-level image feature, detecting them is often a complex task because they are often interfered with by noise or shadows in the image, and sometimes they have different intensities and sources. And because of this, various types of edge detectors have been continually proposed.

With reference to the papers I have read and the following two papers [1][2][3] on edge detection survey, I will first briefly explain the basic techniques related to edge detection in this chapter 2.1, then categorize edge detection into three broad categories: A, traditional hand-crafted methods, and learning based methods, and introduce them in the remaining three chapters 2.2, 2.3 and 2.4.

2.1.2 Image differentiation

In most cases, edge features will separate two regions with different intensity, so

the edge pixels between those must have a significant change in intensity. These changes can be detected by the partial derivatives or gradient, so assume that the input image is a grayscale image, then the image differentiation, i.e. partial derivatives of its intensity function could be a judgement of edge. The mathematical definition of partial derivative is (1), so set h as 1 pixel width and apply in this formula (1), and then we get this formula (2).

$$\frac{\partial f(x)}{\partial x_i} = \lim_{h \rightarrow 0} \frac{(f(x_1, \dots, x_i + h, \dots, x_n) - f(x_1, \dots, x_n))}{h} \quad (1)$$

$$\frac{\partial I(x, y)}{\partial x} = I(x + 1, y) - I(x, y), \quad \frac{\partial I(x, y)}{\partial y} = I(x, y + 1) - I(x, y) \quad (2)$$

Partial derivative can detect the slope on its direction. Therefore, the edge perpendicular to the derivative direction will be detected by the partial derivative. So if we calculate the partial derivative of an 2D image on the x-axis and y-axis direction, then we can detect the edge of all directions, as Fig. 2.1(b)(c) show.

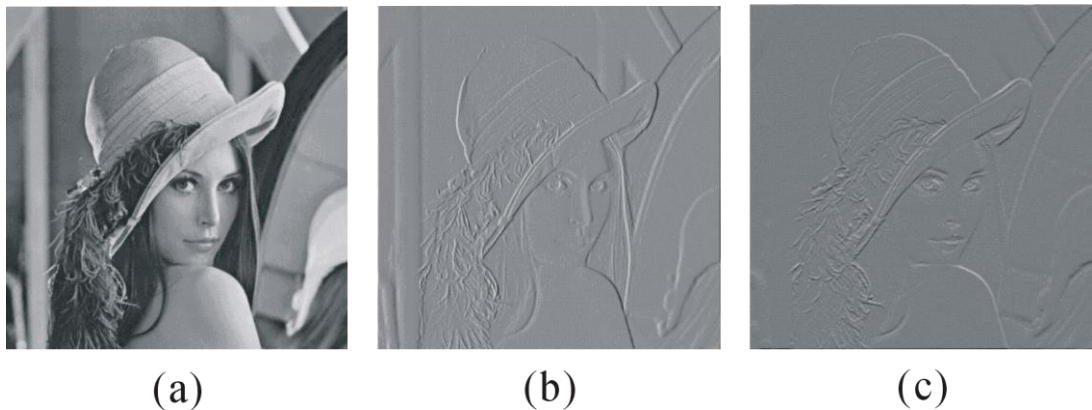


Fig. 2.1 The partial derivative map.

After getting the two partial derivatives, some edge detectors just calculate the summation of derivative absolute value as edge map, i.e. $\left| \frac{\partial I}{\partial x} \right| + \left| \frac{\partial I}{\partial y} \right|$, and others use the definition of image gradient ∇I and calculate the gradient norm $\|\nabla I\|$ as edge map, i.e.

$\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$. The formula of gradient and its magnitude and angle are shown below.

$$\nabla I = \frac{\partial I}{\partial x} \mathbf{x} + \frac{\partial I}{\partial y} \mathbf{y}$$



$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}, \quad \text{angle}(\nabla I) = \tan^{-1}\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right) \quad (4)$$

2.1.3 Thresholding

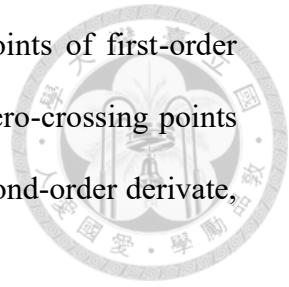
Threshold is a simple method to convert the edge score map to a binary edge map. The method requires a threshold value, and label the pixels larger than threshold value as edges.

The threshold value can be decided by human, or be calculated by algorithm. This value should be larger than the value of noise or background part, and smaller than the edges in image. We can see that if the gap between value of edge pixels and others are bigger, the easier the threshold can separate them, so the performance of threshold depends not only on the selection of threshold value, but also on how good the edge detector is. However, it's almost impossible to separate them perfectly. Besides, edges in different image also have different magnitude, so the same threshold might not be optimized for all edge segments in one image.

2.1.4 Non Maximum Suppression

Non Maximum Suppression(NMS) is an edge thinning method which can better localize edge pixels. To explain how NMS works, let's take a look at the 1d model of edge first. An edge would be like a cliff if we consider the intensity of image as height, so the 1d edge will look like a slope on the figure (Fig.5(a)). To find the edge on the

intensity function, we can find local maxima or local minima points of first-order derivative (first-order derivative might be negative too), or find the zero-crossing points (i.e. positive on one side and negative on the other side) on the second-order derivative, and NMS is just like the maximum calculation we use in 1d edge.



Edges on edge map are like lines of mountains, so using local maximum calculation on edge map will only get some peaks points, not the ridges we want. And the NMS is just the method. Just like the height of mountain, the magnitude of edge will be higher when it is closer to the ridge top, and gradient of magnitude will be oriented towards the ridge top too, as the Fig.6.

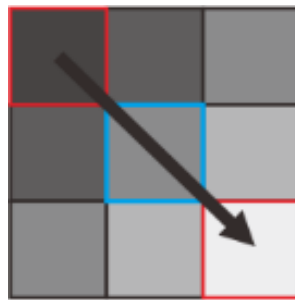


Fig. 2.2 The NMS Diagram

NMS is based on this fact and proposes a condition to filter out the redundant pixels. For each pixel, we will compare it with its two neighbor pixels on the direction of gradient. The neighbor pixels will only contain the other 8 pixel inside the 3*3 region, and gradient direction will be chosen as one of the 4 directions in 0° , 45° , 90° , 135° (no need to consider the opposite direction). For example, the direction of gradient in Fig.6(b) is from top left to bottom right, so we would compare the top left pixel and bottom right pixel to the middle pixel. And if and only if the magnitude of middle pixel is greater than both of two pixels on the gradient direction, that means it is the closest pixel to the ridge on the gradient direction and satisfied the NMS condition.



2.1.5 Edge fixing

This kind of methods use the morphology mathematical calculation. One example is that sometimes a long edge segment will have a short break, perhaps only several pixels wide. A solution of it is to perform morphological closing calculation, which can merge any two disjoint blocks. However, the adjacent edges will also be merged by closing calculation, so some methods will just merge the two line segments whose end points are close enough to each other [5]. After finding endpoints, it will connect these two endpoints along the local maximum pixels on edge score map.

Another example method also proposed by [5] is the repairing of trihedral junction. In Fig. 2.3(a), The three places mark by number are trihedral junctions. Edges in trihedral junction regions are usually less obvious, as the Fig. 2.3(b) shows, and therefore we need to detect these regions and fix them.

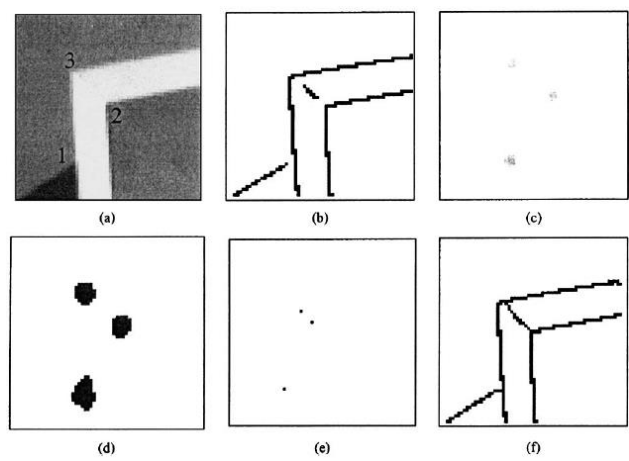


Fig. 2.3 Demonstration of trihedral junction fixing [5]

Image curvature are often used to detect corners or trihedral junctions, and the formula of curvature is:

$$k = \frac{I_{xx}I_y^2 - 2I_xI_y + I_{yy}I_x^2}{|\nabla I|^3} \tag{5}$$

Where the ∇I means image gradient, I_x, I_y mean first order image derivatives, and I_{xx}, I_{yy} mean second order image derivatives. After we get the curvature (the Fig. 2.3 (c)), find the regions where curvature is big enough (Fig. 2.3 (d)) and locate the edge endpoints inside the regions (Fig. 2.3 (e)). Then if any chain of pixels exist that are local maxima of edge score and are connected to an edge endpoint, these pixels are added to the edge map.

2.2 Gradient and Gaussian based methods

2.2.1 Derivative convolution kernels

As mentioned earlier, the image differential is a simple method to obtain edge scores, and convolution kernels are initially used to calculate better partial derivatives for the edge detection. Convolution is a widely used mathematical operation in image processing. Simply speaking, convolution can detect features in the image that are similar to the corresponding convolution kernel matrix. The formula for 2D matrix convolution are show below.

$$F[x, y] = I[x, y] * H[x, y] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I[x - i, y - j] \cdot H[i, j] \quad (6)$$

Where I is the image function, H is the convolution kernel matrix (kernel), F is the output feature function.

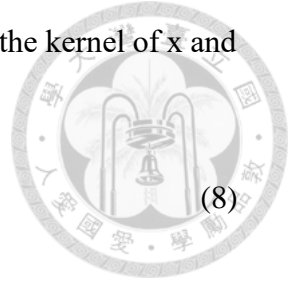
Therefore, we can also use convolution kernels to derive the image partial derivative.

$$\frac{\partial I(x, y)}{\partial x} = I(x, y) * H_x(x, y), \quad \frac{\partial I(x, y)}{\partial y} = I(x, y) * H_y(x, y) \quad (7)$$

Where H_x and H_y are the partial derivative kernels of x and y. Take the previous

partial derivative formula as an example, in that derivative formula, the kernel of x and y directions, H_x and H_y , are:

$$H_x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, H_y = [1 \quad -1] \quad (8)$$



$$\left(\frac{\partial I(x, y)}{\partial x} = I(x + 1, y) - I(x, y), \frac{\partial I(x, y)}{\partial y} = I(x, y + 1) - I(x, y) \right)$$

There are many kinds of convolution kernels for computing derivatives, and here is a table about 3 famous kernels invented at that time for edge detection.[1]

Table 2.1 Kernels of derivative

| Method | Kernels of first derivative on x & y directions |
|---------|---|
| Roberts | $H_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, H_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ <p>(*It's not on x and y direction, but still perpendicular to each other)</p> |
| Sobel | $H_x = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, H_y = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ |
| Prewitt | $H_x = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, H_y = \frac{1}{3} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ |

Different kernels have different pros and cons, for example, the Prewitt kernel perform better on the vertical or horizontal straight edge. There are many kernels with different designs and functions, and even different sizes and directions.

2.2.2 Gradient calculation

Gradient is a function of image which return vector which calculate the slope and angle of that pixel, so the vector norm of gradient can also be seen as an edge score.

There are mainly two methods to compute the gradient of pixels form a digital

grayscale image, and both of which need the calculation of partial derivatives. The first calculating method, which we already mention in previous section, is to sum up two partial derivative vector in the x-axis and y-axis directions, as the Fig. 2.4(a) shown.

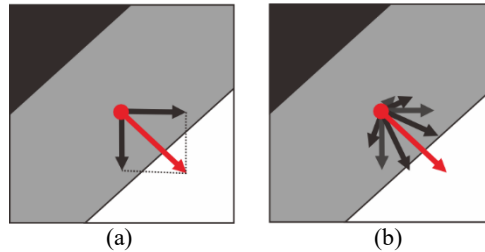


Fig. 2.4 Gradient calculation method by using first order derivatives

The formula of its vector form ∇I , gradient magnitude $|\nabla I|$ and gradient angle $\text{angle}(\nabla I)$ are shown below. \mathbf{x} and \mathbf{y} mean the unit vector on the direction of x-axis and y-axis.

The other definition of gradient is the maximum derivative of derivatives on all directions. Therefore, we need to choose some angles which evenly distribute between 0° and 180° and smaller than 180° (for example: these four direction $0^\circ, 45^\circ, 90^\circ, 135^\circ$), and then find the maximum partial derivative. The magnitude and angle of gradient will be the partial derivative and its angle (Fig. 2.4(b)).

Both methods have their own advantages and disadvantages, but the former method is more simple, because the latter method usually need more partial derivatives to achieve a good gradient, while the former only need 2. Besides, the calculation of derivative in x or y directions is easier than which of other directions, so most of the earlier edge detectors, such as Sobel and Prewitt, use the former method.

2.2.3 Image smoothing

The disadvantage of the gradient methods is that derivatives from simple kernel are easily interfered by image noise. Noise is an unavoidable problem in images and can affect edge detection and interfere with edge pixel location. Although noise can be

removed by the image smoothing techniques, it's a challenge to remove noise while minimizing the loss of image features and blurring of edges, so how to take both into account is the main point[1][3].

One of these is median filter and mean filter. Mean filter will take the weighted average of window, which containing target pixels and its neighbor pixels, as the new pixel intensity, and one of the most commonly used weighting functions is the Gaussian function. Median filter[61] will take the median number inside window as new pixel value, which makes it particularly effective against certain types of noise(for example impulse noise).

2.2.4 Gaussian blur

Gaussian blur is a widely-used image smoothing method, and as its name, Gaussian blur can blur the input image to eliminate noise. Although the blurring will reduce both noise and detail of image, but most of the edge feature can still be reserved.

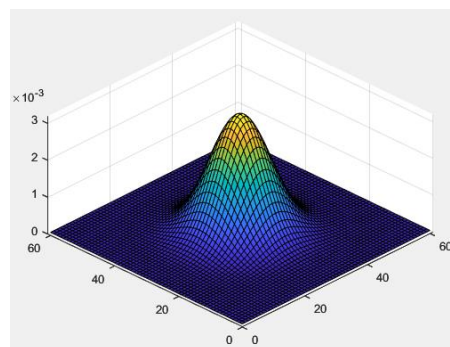


Fig. 2.5 The Gaussian kernel

Gaussian blur is applied by doing a convolution between the input image and Gaussian function (Fig. 2.5) which is also called normal distribution. And the formula of Gaussian kernel is at here:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (9)$$

where the scale constant σ is the scale size of Gaussian kernel. From the aspect of signal processing, Gaussian is a kind of low-pass filter, and noise is usually high-frequency signal. Therefore the scale constant determines how high the frequency of the noise is blocked by Gaussian filter. In other words, the larger the scale constant, the lower the allowable frequency, and the wider and stronger the noise will be removed, the greater the blurring will be.

The fact that Gaussians have the property of scaling also makes Gaussian-based methods capable of detecting edges of corresponding widths, perhaps making them ultimately widely used by later edge detection methods. In the next paragraph we will introduce two of the earliest classical methods for Gaussian-based edge detection, Canny edge detector[4] and Laplace of Gaussian(LoG) [7].

2.2.5 Canny edge detector

Canny edge detector is proposed by Canny [4] in 1986. Canny used mathematics to prove that derivate of Gaussian is one of the optimal solutions which satisfied these three criteria: good detection, good localization and single response. And due to the nature of convolution, we can use a single kernel to calculate the derivate of Gaussian, as the following equations show:

$$f(x) * g(x) = g(x) * f(x) \quad (10)$$

$$[f(x) * g(x)] * h(x) = f(x) * [g(x) * h(x)] \quad (11)$$

$$\frac{\partial}{\partial x}(I(x, y) * G(x, y)) = \left(\frac{\partial}{\partial x} * G(x, y) \right) * I(x, y) \quad (12)$$

Derivate of Gaussian kernel is a basis of many later edge detectors. However, because the kernel of derivative is much smaller than the kernel of Gaussian or derivate of Gaussian, so just calculate the derivatives from image blurred by Gaussian is faster,

and that's what Canny do in his edge detection algorithm.

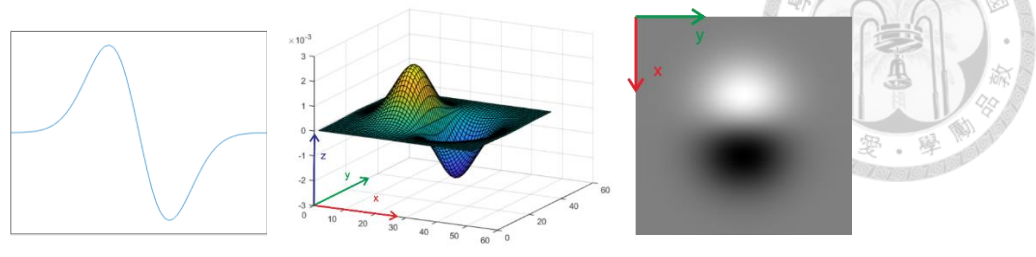


Fig. 2.6 Derivative of Gaussian(DoG) kernel

Except for Gaussian blur, Canny also used NMS and double threshold to thinning the edge and make edge better. We have introduced NMS before, and double threshold is an improved version of threshold. There is an observation that some edge segments are deep in some parts and particularly shallow in others, which makes us uneasy to use a single edge to detect the total edge segments. Therefore, the double threshold method uses two threshold value, the high threshold T_1 and low threshold T_2 . The pixels larger than high threshold T_1 will satisfied the edge condition, and the edge smaller than high threshold T_1 , but larger than low threshold T_2 can also satisfied the condition.

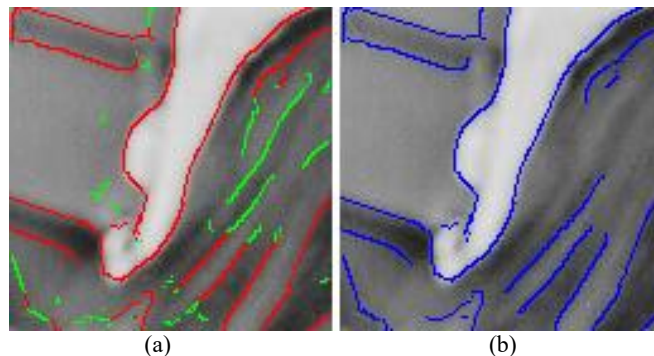
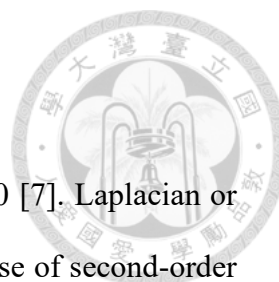


Fig. 2.7 The demonstration of double threshold

As the Fig. 2.7 shows, the shallow edge parts (green edge) which connected to deep edge parts (red edge) can be reserved, while the lonely shallow edge are abandoned.



2.2.6 Laplacian of Gaussian edge detector(LoG)

Laplacian of Gaussian(LoG) edge detector is proposed at 1980 [7]. Laplacian or Laplace operator is an operator which can be consider as the 2D case of second-order derivative, and Laplacian of Gaussian is just the literally what it means, the Laplacian of image blurred by Gaussian smoothing. The formula of Laplacian and LoG are shown below.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{13}$$

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{14}$$

The figure of LoG is also shown below. The shape of this kernel function looked like a Mexican hat, so it's also called Mexican hat function (Fig. 2.8).

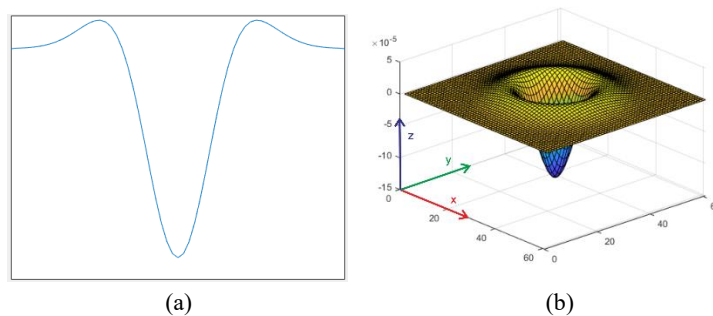


Fig. 2.8 The LoG kernel (a) 1D case (b) 2D case

There are two ways to find the location of the edge, the former is to find the extremum points of first-order derivative (Fig. 2.9(b)), and the latter is to find the zero-crossing of second-order derivative(Fig. 2.9(c)). Therefore, the zero-crossing of Laplacian can be used as edge feature.

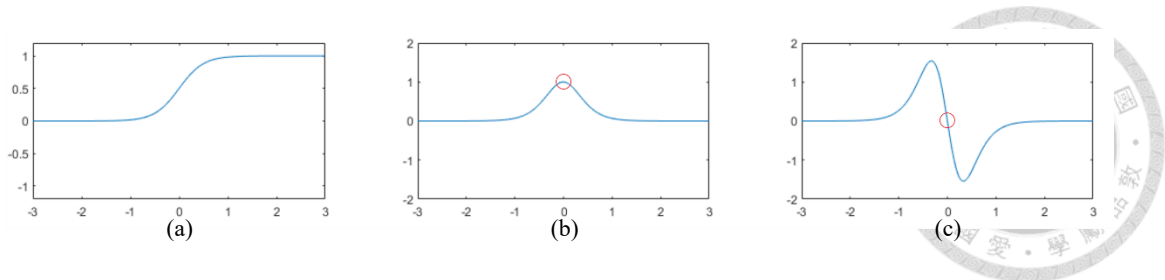


Fig. 2.9 1D-case of edge detect (a) Original ridge intensity function $I(x)$ (b)(c)

First and second derivative of intensity function $I(x)$

The detection of zero-crossing pixels is easy, just picks up those pixels whose positive and negative are different to their neighbor pixels. In order not to detect two pixels in one zero-crossing, the algorithm will only consider the down and right neighbor pixels (i.e. only positive x & y directions).

LoG is also useful in other image processing too, but in order to save computation, it is sometimes approximated by a graphically similar Difference of Gaussian function, as the equation (15) shown.

$$\nabla^2 G(x, y, \sigma) \approx G(x, y, \sigma_1) - G(x, y, \sigma_2) \quad (15)$$

2.2.7 Multi-detector and multi-scale fusing

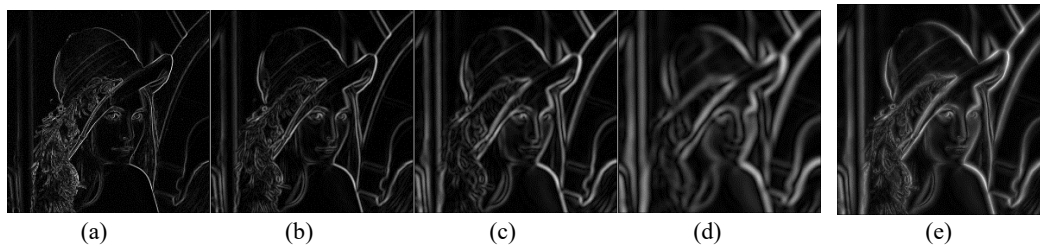
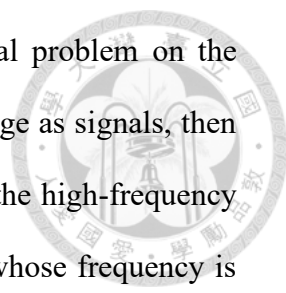


Fig. 2.10 Edge maps with different scale of DoG.

(a)-(d) Edge maps from small scale to big (e) Fusing edge map.

Edge is a complex feature and the results from a single detector are often insufficient, so some methods will apply multiple edge score form different detectors to produce a single edge map. Common functions used in merging of different edge score map are the Euclidean norm or maximum function.



Multi-scale method is one of the solutions of another normal problem on the Gaussian based methods. If we consider the information of the image as signals, then the gradient based edge detector is a high-pass filter which detect the high-frequency edge feature, but Gaussian smoothing will block the noise signal whose frequency is higher, so the Gaussian-based edge detectors will be a band-pass filters whose scale is decided by the size of Gaussian kernel. We can see that in Fig. 2.10(a)-(d). A solution of it, which was also mentioned as a solution by Canny[4] in the same thesis, is to fuse multiple Gaussian edge score with different scales, and that's exactly the multi-scale methods. As Fig. 2.10 (e) show, the performance of multi-scale edge map **become** better than map of single scale.

2.3 Other hand-crafted methods

Gaussian-Based method still has some problem to be fixed, so many new methods were proposed after Canny and LoG. Some of them are improvements of the old methods, while others try to find new solutions to get better performance.

These improvement novel methods can be broadly classified into two categories. The first kind of methods focus on the optimization of edge map computation, and they usually utilize multiple different edge maps by different methods, and then use them to generate a better edge map. An example is the previously mentioned multi-scale method. Another kind of methods are the pre-processing and post-processing, which are the processing performed on input images to make the edge map better, such as Gaussian blur or NMS and threshold. Of course there are some edge detection use the both methods.

2.3.1 Color edge methods

As shown in Section 2.2, the gradient-based and Gaussian-based methods are

designed for gray-scale images, and in the case where the input image is a color image, the image is usually converted directly to grayscale by the brightness formula (16).

$$I(x, y) = 0.229 \cdot R(x, y) + 0.587 \cdot G(x, y) + 0.114 \cdot B(x, y) \quad (16)$$

A color image is composed by three color channels: red, green and blue. As a result, color information will be lost while a color image is converted to a grayscale image.

There are two main types of color edge methods: the first type is to calculate the edge scores for each color channel separately and then just combine them; the other class is to use the color vector. In a color image, a color vector consisting of three color intensities will be assigned to a pixel, and the distance between the color vectors will be used as the difference between pixels in the edge score calculation. The distance can be calculated by different formulas, such as Euclidean distance, or χ^2 distance[5],

$$\chi^2(g, h) = \frac{1}{2} \sum \frac{(g_i - h_i)^2}{g_i + h_i} \quad (17)$$

Where the two colors are $g = (g_1, g_2, g_3)$ and $h = (h_1, h_2, h_3)$. Besides, there's also an approach to calculate color gradient by matrix multiplication.

Another way to think about it is the color space conversion. One benefit of this method is that non-RGB color spaces, such as the Lab color space and YCbCr color space, sometimes appear better on the edge detection, and another is that these color space can separate brightness and color components, which helps us greatly in reducing the effects of shadows. The color-boundary method is also an example which uses the Red-Green and Blue-Yellow color channels in its calculation.

Apart from these methods, a thesis [10] proposed a method based on Pointwise Mutual Information(PMI) which use the information theory to help the edge judgement. This paper finds that many false edges, such as those produced by stripes or leaves, the neighboring pixels will be certain color combinations, i.e. color adjacency. Take Fig.

2.11 as example, the color adjacency of the stripes on the zebra (red circle) is black and white, and the color adjacency of grass and trees (blue circle) in background is two different green.

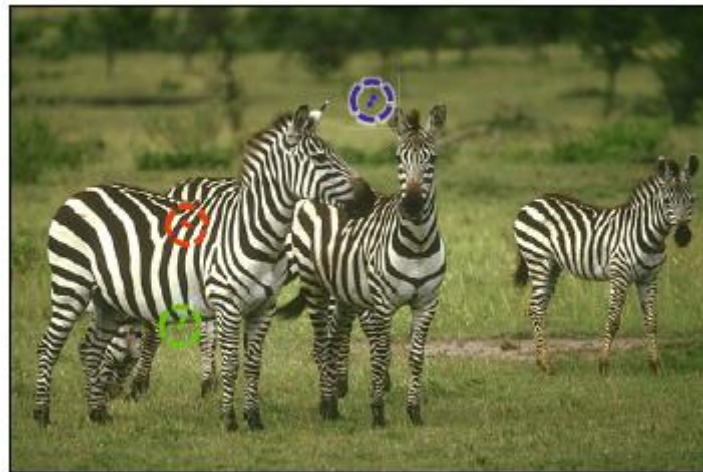


Fig. 2.11 An example of color adjacency in PMI[10]

And this paper performed PMI on the pixel colors of the input images to analyze the probability of two colors being adjacent. If any two pixels have high probability of color adjacency, then the probability of edge between them will be reduced.

2.3.2 Fuzzy methods

Fuzzy methods are based on the fuzzy logic which uses the concept of fuzzy and uncertainty. Unlike Boolean logic which expresses things as 0 and 1, the definition of edge in fuzzy methods is related to uncertainty, so some studies try to apply fuzzy theory to edge detection [2]. Besides, fuzzy logic is usually nonlinear (while convolution is linear), so it can do things that can't be done in other linear methods.

Fuzzy methods are complicated, so I will only take Russo's fuzzy edge detector [6] as example to explain how these methods work. Russo's method will use some fuzzy function (Fig.14) in the edge judgement and the min and max calculations to eliminate the salt-and-pepper type noise.

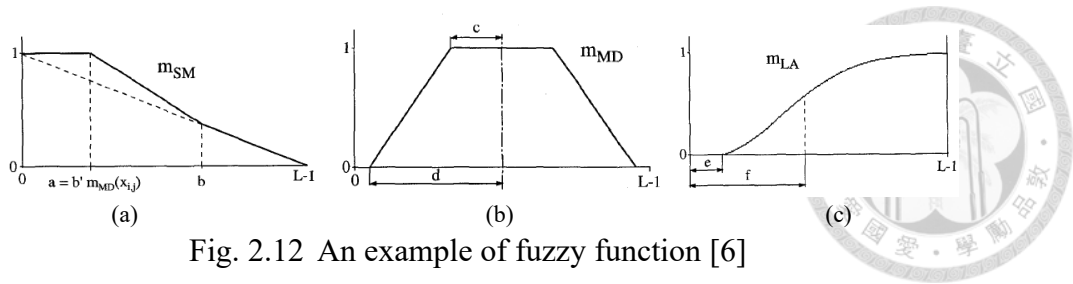


Fig. 2.12 An example of fuzzy function [6]

We can see that the function value will not change while the input value is small, which can be seen as a method to avoid noise from affecting the output.

2.3.3 Texture methods

For most edge detectors, once if the average intensity (or color) of the regions on both sides of an edge are similar, then this edge is hard to be detected, so edge detections about texture are proposed.

In Martin's method[9], the edge map is computed by four different edge score, and one of them is the texture feature edge score. In the computation of texture edge score, first 64 texture patterns(textons) are computed from the 200 training images (Fig. 2.13(b)). After using these textons as convolution kernels and performing convolution, a texture map with texture vector of each pixel will have a length of 64 will be generated (Fig. 2.13(c), (d)), and the texture edge score will be the χ^2 difference() between feature vectors of different pixels.

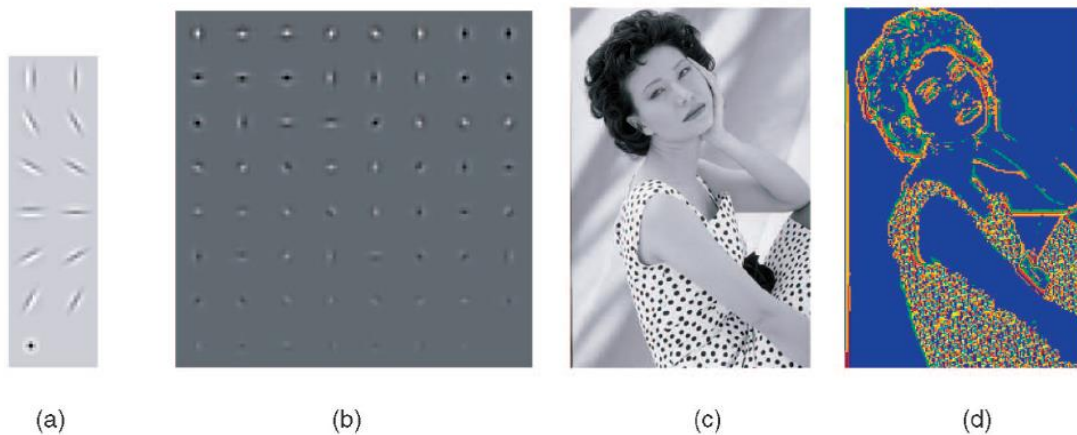


Fig. 2.13 The textons and texture map [9]



2.3.4 Wavelet methods

Wavelet transform is a signal analysis which can detect signal and its frequency. Different from the Fourier transform, it can also detect the location of signal; besides, it can detect the same signal at different scales, so it's also applied for edge detection.

For edge detection, we need to use the discrete wavelet transform. As the Fig shows, it can decompose the signal into high frequency and low frequency components on the two axis.

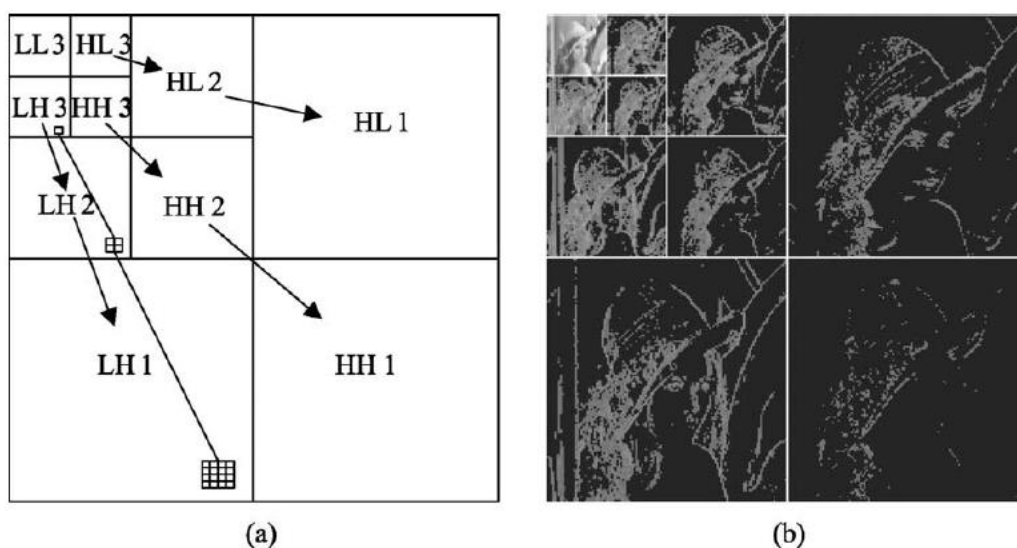


Fig. 2.14 An example of 2D wavelet decomposition. [8]

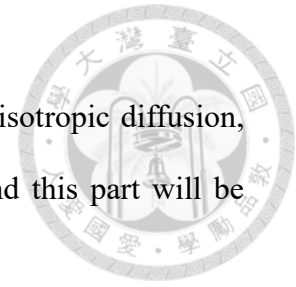
Another method [14] use shearlet transform, which is an extension of wavelet transform, to applied a multi-scale direction detector and get a good performance[2].

2.3.5 Edge-preserving smoothing methods

Gaussian smoothing is a widely used denoising method, but it sometimes becomes a problem in many edge detection methods because it will also erase the edge feature. Another kinds of methods to solve that problem is the edge-preserving smoothing. It's an image smoothing algorithm which could detect edge feature during the smoothing

process and then preserves them afterwards.

There are many edge-preserving smoothing methods, like anisotropic diffusion, bilateral filter, adaptive bilateral filter and guided image filter, and this part will be explained in more detail in the following Chapter 4.



2.4 Learning based methods

2.4.1 Traditional learning-based method

Unlike traditional hand-crafted methods where edge detectors are designed manually, learning-based methods allow the algorithm to automatically find the desired image patterns from the input image data, or to learn how to judge edges by multiple input features through the training data. Learning-based methods are more efficient than human methods at these works, so some learning-based edge detection methods have been proposed and have performed well on edge detection.

For example, Pb[9] using texton learned from training and use logistic regression to detect edge by different image feature maps, and SE[11] use learning-based decision tree to classify the edge patterns and then detect them.

2.4.2 Convolutional Neural Network

The emergence of deep learning has impacted many domains, and edge detection is also included. In the deep learning domain, Convolutional Neural Network(CNN) is the architecture which has the best performance in image processing, and therefore almost all of the best-performing edge detection methods recently are CNNs. CNNs are quite powerful in learning high-level features in images, so they have good ability to extract semantic edges which cannot be easily detected by human designed methods.

The main topic of CNN models is to make the neural network model more efficient,

which is not similar to those traditional methods. For example HED[17] applied the powerful VGG16[16] structure, and RCF[12] and BDCN[18] focus on simplify the parameter number and solving multiscale problem.

There are also some methods focus on the characteristic of edge and training data of edge detection, for example the pidinet[19] change the convolution logic in CNNs to better fit the edge detection, and UAED[20] focus on the uncertainty of groundtruth image of training data and try to fix it.



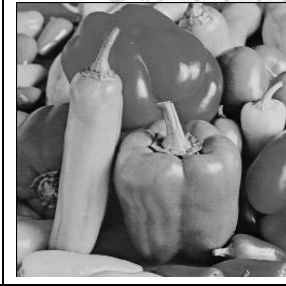

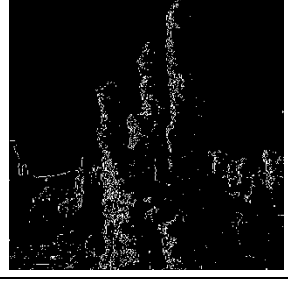


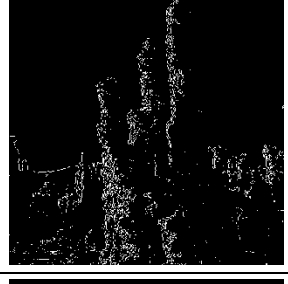


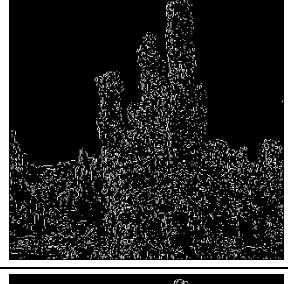


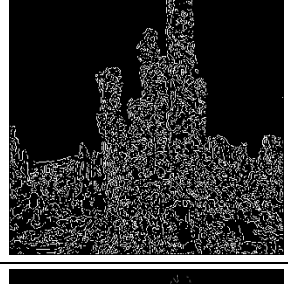
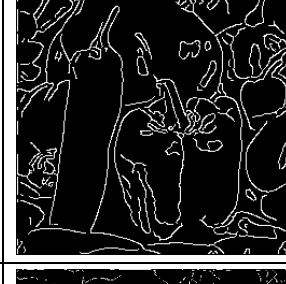

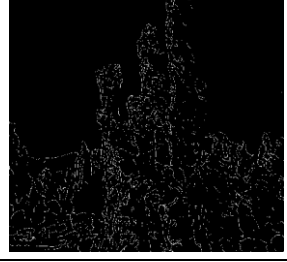

2.4.3 Transformer model


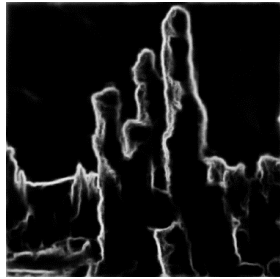
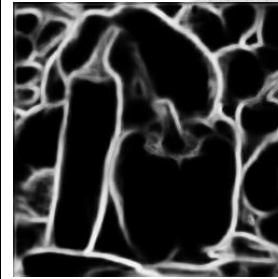
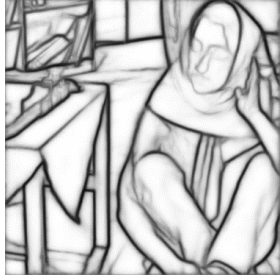
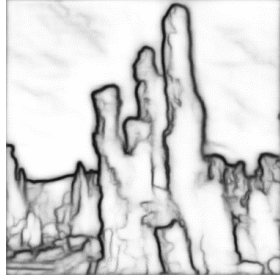


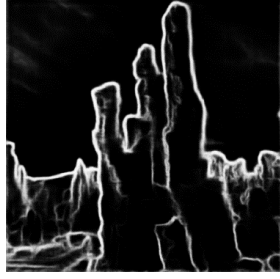
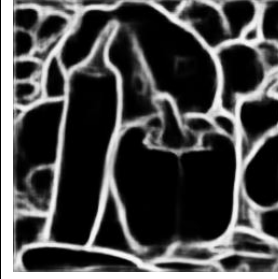

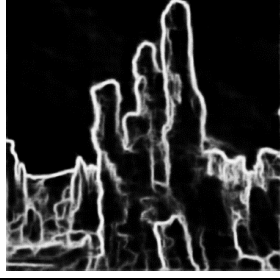
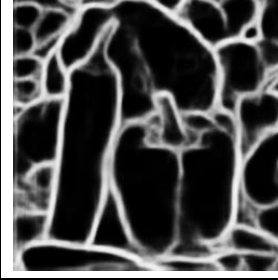

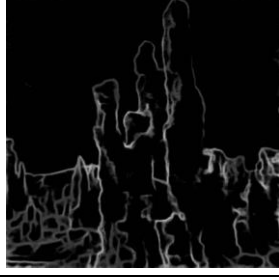
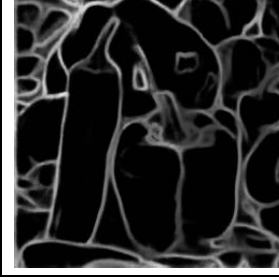
Transformer is a powerful deep learning architecture which is mainly used on natural language processing and computer vision fields. It just likes the convolution with unfixed kernel size and shape. The Vision transformer(ViT) model even got a score better than CNN models while the amount of training data is huge. In the EDTER method[13], the pre-train model of ViT is used in the training, and EDTER is the currently best edge detector in the performance score.

2.5 Demonstration of survey methods

In the process of studying related papers, we have successfully practiced some of the previous edge detection methods either through the programs provided by paper authors or by writing programs based on the algorithms provided in those papers. We hope this simple demonstration shows how edge detection technology is evolving as technology advances.

Table 2.2 Experiment outputs of survey edge detection methods

| | | | |
|--------------------------|---|--|---|
| Original image |  |  |  |
| Prewitt |  |  |  |
| Sober |  |  |  |
| Log |  |  |  |
| Canny[4] |  |  |  |
| Edge- Drawing [59] |  |  |  |

| | | | |
|-----------------|---|--|---|
| HED[17] |  |  |  |
| RCF[12] |  |  |  |
| BDCN [18] |  |  |  |
| Pidinet [19] |  |  |  |
| RINDNet [60] |  |  |  |



Chapter 3 Ridge filter



3.1 Introduce

Ridges and valleys are image features consist of connected linear local maximum or minimum pixels, respectively, and because valleys are ridges with opposing intensity distributions, so detections of both ridges and valleys are usually categorized as ridge detections.

Ridge detection is useful in medical or biological images. Besides, both of ridges and edges are image features that appear on the boundaries of an object or an area, and thus ridges are sometimes regarded as a kind of edges. Because of the similarity between ridges and edges, ridge detection has a lot in common with edge detection. The explanations of the following ridge detection methods are based on my survey and two papers of ridge detection survey[21].

3.2 Undirected filters

Convolution is the most intuitive tool to detect image features, and for ridge features, which often have their direction, there are two kinds of convolution methods: undirected filters and oriented filtered.

The most common undirected filters for ridge detection are the LoG filter, which is also used as an edge detector at Section 2.2.6. In contrast to the usage in edge detection, where the zero-crossing of LoG is used as edge feature, in ridge detection the LoG is just the ridge score. Beside from LoG, Difference of Gaussian (DoG) or difference of Low-pass filter are also used in ridge detection.

3.3 Oriented Filters

Oriented filters are designed to be rotatable and capable to detect feature which has the same angle with filter. An example is the 2nd derivative Gaussian kernel.

$$\frac{d^2G}{dx^2}(x, y) = \frac{1}{2\pi\sigma^6} (x^2 - \sigma^2) \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (18)$$



As the cases of edge detectors, this kernel can be rotated to detect ridge in different directions.

There are mainly two methods of oriented filter methods. The first method is to apply filter with corresponding angle on image, which requires knowledge of the direction of the ridges. A common way to find out the ridge direction of a pixel is usually to calculate its gradient direction. The second method is to apply filters with all angles and fuse the ridge scores of all angles.

3.3.1 Low-Pass filter based methods

Except for the Gaussian-based kernel filters, There are still other Low-Pass filters than can be used to make oriented filters, for example the Ziou Filter Z and Gouton Filter R [21] (Fig. 3.1(a)).

$$Z(t) = \frac{1}{s_z^2} \cdot (1 + s_z \cdot |t|) \cdot e^{-s_z \cdot |t|}, \quad R(t) = (K \cdot \sin(s_r \cdot |t|) + D \cdot \cos(s_r \cdot |t|) + E) \cdot e^{-s_r \cdot |t|}, \quad (19)$$

Their second derivative (Fig. 3.1(b)) can be use as the ridge detector in 1D, and 2d kernel can be derived by taking a Low-Pass filter and its vertically transposing 2nd order differentials.

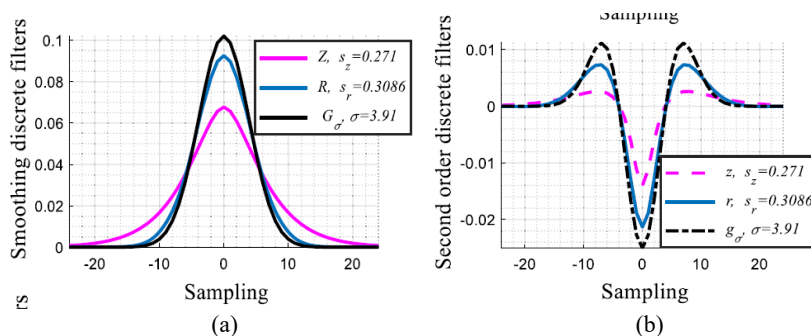


Fig. 3.1 Ziou, Gouton and Gaussian filters [21]



3.3.2 Gaussian based methods

Methods in this part are designed based on the derivatives of the Gaussian kernel. Steerable Filter [32] can derive a series of filter that are odd-symmetric or even-symmetric consisting of derivative with different order, and the design of filter can be changed by adjusting the parameters. For example, two of the steerable filters are given by:

$$SF(M = 2, \mu = 0) = -\sqrt{2/3\pi} \sigma g_{yy} \quad (20)$$

$$SF(M = 4, \mu = 0.25) = -0.392\sigma g_{yy} + 0.113\sigma g_{xx} + 0.034\sigma^3 g_{yyyy} + 0.113\sigma^3 g_{xxyy} + 0.113\sigma^3 g_{xxxx} \quad (21)$$

where g is the Gaussian kernel and $g_{xx} = \partial^2 g / \partial x^2$, $g_{yy} = \partial^2 g / \partial y^2$, $g_{xxxx} = \partial^4 g / \partial x^4$, $g_{xxyy} = \partial^4 g / \partial x^2 \partial y^2$, $g_{yyyy} = \partial^4 g / \partial y^4$.

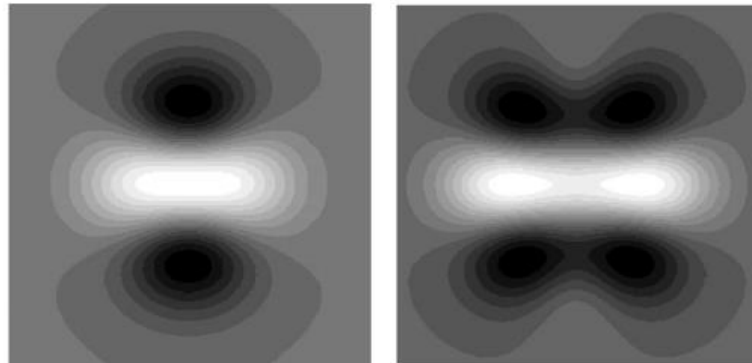
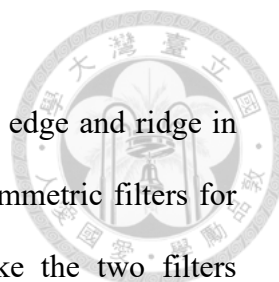


Fig. 3.2 The steerable filter (a) Case($M = 2, \mu = 0$)(b) Case($M = 4, \mu = 0.25$)[32]

SOAGK[22] uses the second order derivative of anisotropic Gaussian kernel, i.e. it uses a Gaussian kernel with adjustable aspect ratio, and then try all the combinations of angles, scales, and aspect ratio. The advantage of it is that longer kernels are more powerful at detecting some ridge lines, for example the crossing lines or straight line situation. SCIRD[24] further makes the curvature of kernels can be adjusted so that the kernels would be curved-support, which is powerful at detecting curved ridge segment.



3.3.3 Hilbert transform

Paper [23] proposed an algorithm which is designed to detect edge and ridge in the same time. For this purpose, it uses a pair of odd and even symmetric filters for detecting edges and ridges, respectively. Then, in order to make the two filters orthogonal to each other, one of the filters will be the Hilbert transformations of another one, which is a new method to generate a ridge filter.

3.4 Non-filter methods

Unlike filter methods that use convolution to measure the magnitude of second derivative to detect ridges, derivative-based methods use derivatives to derive ridges. Many of these methods will still use those Gaussian derivative kernel to calculate the derivatives to minimize the effect of noise.

3.4.1 Hessian matrix

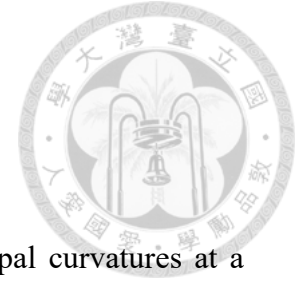
Hessian matrix is a tool to find out the main partial derivatives along all directions of the image function. For a 2D image, its Hessian matrix of any given point will be a 2*2 matrix.

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}, \text{ where } \begin{cases} I_{xx} = \frac{\partial^2 I}{\partial x^2} \\ I_{xy} = \frac{\partial^2 I}{\partial x \partial y} \\ I_{yy} = \frac{\partial^2 I}{\partial y^2} \end{cases} \quad (22)$$

The two eigenvalues of Hessian matrix are the magnitude of the two main curvatures (2nd derivatives) of the intensity function, and the two eigenvectors are the direction of those 2nd derivatives. These two main derivatives will orthogonal to each other, and one of it will be the biggest derivative in all direction.

An early method [25] used Hessian matrix to classify pixels as flat, edge or corner pixels by the two eigenvalue magnitudes, and this method can be used to detect ridge by the same idea of it. The latter methods usually utilize one or two of these eigenvalues,

or use the direction of main derivatives to check ridge direction.



3.4.2 Weingarten

Weingarten map[21] is a method to calculate the two principal curvatures at a given point of the surface. Curvature rather than 2nd order derivative.

For an 1D signal I , the second derivative of it will be, and the curvature of it will be as below.

$$K_{xx} = -\frac{I_{xx}}{(1 + I_x)^{3/2}} \quad (23)$$

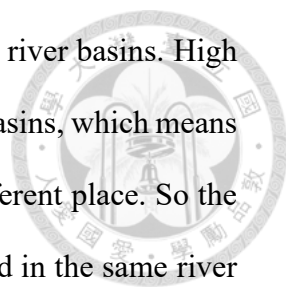
Where I_x means the 1st x derivative of signal I and I_{xx} means the 2nd x derivative. In a nutshell it is like the 2nd order derivative which perpendiculars to the signal I . And for a 2D intensity function, Weingarten is like the 2D version of formula and the curvature version of Hessian matrix.

$$W(x, y) = \frac{1}{(1 + I_x^2 + I_y^2)^{\frac{3}{2}}} \cdot \begin{pmatrix} 1 + I_y^2 & -I_x I_y \\ -I_x I_y & 1 + I_x^2 \end{pmatrix} \cdot \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix}. \quad (24)$$

Similar to Hessian matrix, the matrix here is a matrix representation of curvature. The two eigenvalues and eigenvectors are the two principal curvature of intensity function and the two directions of them. Therefore, these eigenvalues and eigenvectors can also be used in ridge detection.

3.4.3 Separatrices and Drainage patterns

These two types of methods[29] are designed for real ridges and valleys on digital elevation models(DEMs) image, i.e. map of terrain height, and because of this, both of these two kinds of methods utilize the rain water simulation simulated by the slope of DEMs images.



Separatrices method is based on the concept of watersheds and river basins. High mountain ridge usually acts as watershed that divide different river basins, which means that rainwater falling on either side of it will eventually flow to different place. So the pixels whose slopelines go to the same minimum will be categorized in the same river basins, and the boundaries between different districts will be ridge.

Drainage Patterns method uses the fact that rainwater that falls to the surface will eventually flow into the valley in real world. It simulates the water flow which flows to the lower place on DEM by the slope direction, and then measures the flow of water as valley scores.

3.4.4 Divergence

A paper [31] found that in some ridge points, for example points around a saddle point, the ridge will be discontinuous. In order to solve this problem, method proposed by paper uses the divergence of normalized gradient vector as ridge score and then use structure tensor to replace gradient vector in further method.

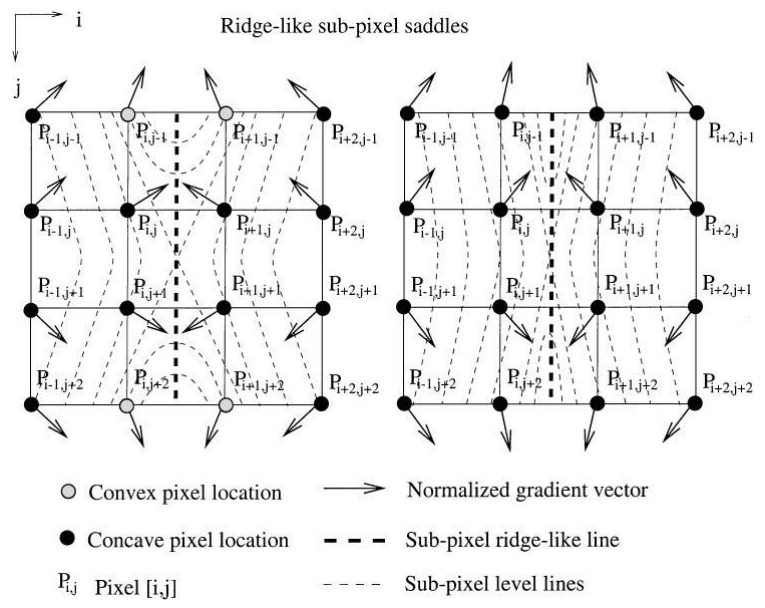


Fig. 3.3 A demonstration of finding ridge by divergence [31]

3.4.5 Polynomial-fitting methods

The Gaussian derivative kernels can be regarded as ways to estimate the shape of local area around a pixel point, and Savitzky-Golay (S-G) filter is something similar, which uses a LS polynomial fit to estimate the local area, so that the image derivatives of different orders and scalar can be determined. The polynomial-fitting process also has a smoothing effect, so Gaussian is not needed after this calculation.

Method in [27] uses this polynomial to find the image derivatives of different order, and then use the derivatives to estimate the ridge direction, which will be used in the fusion of ridge maps detected by different oriented filters. [6] uses the polynomial to find out the Hessian matrix and check the variations of intensity along four directions of Hessian principal curvature, and that determine whether a point is plane, edge, peak, corner or a ridge by the combination of variations.

Chapter 4 Bilateral filter



4.1 Introduce of bilateral filter

Bilateral filter [33] is an edge-preserving smoothing filter for images or signal. Although Gaussian Blur is a very common and useful tool in many image processing, but one of the problem with it is that smoothing process also causes the low-level feature such as edges in the image to disappear. To solve this problem, edge-preserving smoothing methods have been invented and bilateral filter is one of them. Formula of bilateral filter is:

$$I^{BF}(x) = \frac{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r}(I(x) - I(y)) \cdot I(y)]}{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r}(I(x) - I(y))]} \quad (25)$$

Where the function G_{σ_s} and G_{σ_r} mean the Gaussian function whose scale parameter are σ_s and σ_r . In contrast, the formula of a Gaussian smoothing filter is:

$$I^G(x) = \frac{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot I(y)]}{\sum_{y \in W_x} G_{\sigma_s}(\|x - y\|)} \quad (26)$$

We can see that the weight values of bilateral filter obtained by multiplying the following two components: the spatial kernel $G_{\sigma_s}(\|x - y\|)$ and the range kernel $G_{\sigma_r}(I(x) - I(y))$, while the weight values of Gaussian filter only contain the spatial kernel. The effect of the spatial kernel is to reduce the effect of more distant pixels, and the effect of the range kernel is to reduce the weight of pixels in the window whose brightness values are far from the center pixel. In this way, the pixels close to the edge will not be affected by pixels on the other side of the edge, so that bilateral filter can prevent the edge from disappearing.

The following papers [34][35] are referenced in this chapter.



4.2 Improvement of bilateral filter

4.2.1 Speed Improvement

Many of improvements on bilateral filter are about the computation speed. Because unlike the Gaussian blur, bilateral filter is non-linear and unable to use convolution to speed up. By the help of convolution, the complexity of Gaussian filter is $O(n \log n)$ where n is the number of pixels, and the complexity of bilateral filter is $O(nm)$ where m is the kernel size, and the complexity of it is obviously much larger. In order to solve this problem, different types of speeding up methods are proposed and following are some examples [49].

Durand and Dorsey [42] propose a method to approximate the output of bilateral filter. The formula of bilateral filter can be considered as a convolution between $G_{\sigma_s}(\|x - y\|)$ and $G_{\sigma_r}(I(x) - I(y))I(y)$ as long as pixel value $I(x)$ is a fixed value for all pixel position x , so if to replace the pixel value $I(x)$ with a fixed intensity value i to replace, then the whole computation can be done by convolution. According to this idea, as long as we make convolutions between $G_{\sigma_s}(\|x - y\|)$ and $G_{\sigma_r}(i - I(y))I(y)$ for all possible intensity value i in image, then we can get the output of bilateral filter too. But it's still slow*, so two things are done to further speed up it: the first thing is to calculate some of the intensity i , and then use interpolator to approximate the other intensity values. The second thing is to downsample the image in the convolutions, and then upsample them to original resolution after convolution.

J. Chen, S. Paris and F. Durand [44] proposed another method using a new idea: Bilateral Grid, which convert the 2D image to a 3D grid where the first and second axis are pixel position and third axis are the pixel intensity $I(x, y)$, and then use a 3D Gaussian kernel to do the convolution.



4.2.2 Adaptive bilateral filter

Some researches change the structure or the kernel of bilateral filter to improve the performance of bilateral filter, for example adaptive bilateral filter can change range kernel based on different locations in the image. Paper [39] proposed a bilateral filter (equation (27)):

$$I^{ABF}(x) = \frac{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r(x)}(I(x) - \zeta(x) - I(y)) \cdot I(y)]}{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r(x)}(I(x) - I(y))]} \quad (27)$$

The value of $(\sigma_r(x), \zeta(x))$ for each pixel will be decided by LoG response, so it can automatically adjust whether the smoothing effect should be enhanced or reduced. Another method[36] uses polynomial fitting on the intensity histogram as range kernel.

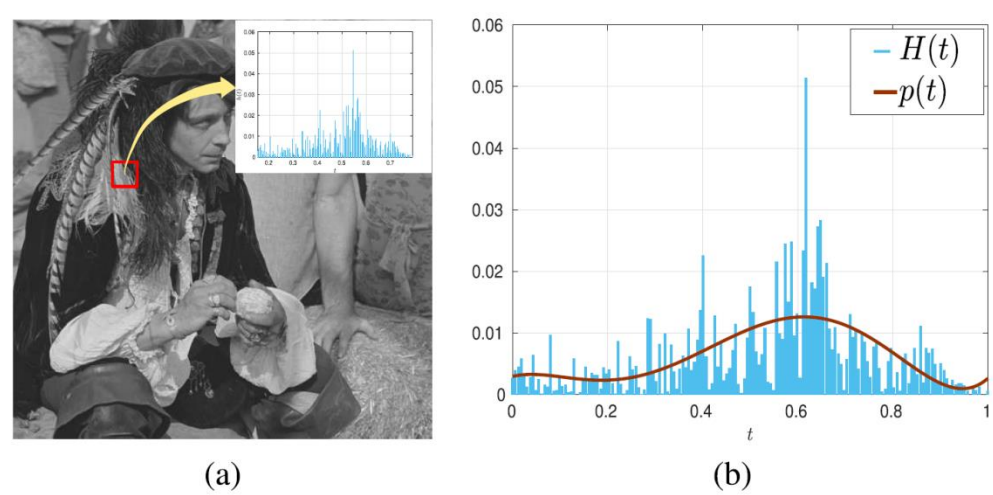
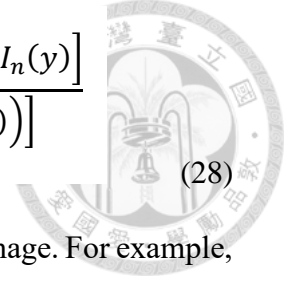


Fig. 4.1 Adaptive bilateral filter based on polynomial fitting [36]

4.2.3 Joint/Cross Bilateral Filter

Joint/cross bilateral filter [45][46] is a special kind of bilateral filter, which use two images in the filtering, the original image I_n and the guidance image I_g . The idea is to use a noise-free guidance image to derive better range kernel weight, i.e. replace the image intensity in range kernel with guidance image intensity.

$$I^{JBF}(x) = \frac{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r}(I_g(x) - I_g(y)) \cdot I_n(y)]}{\sum_{y \in W_x} [G_{\sigma_s}(\|x - y\|) \cdot G_{\sigma_r}(I_g(x) - I_g(y))]} \quad (28)$$



The guidance image should be highly associated with original image. For example, photo taken without flash and the same photo but taken with flash and less noise, or the original image and its image-processed version. If the noise-free guidance image does not exist, then it can be derived from a denoising filter, such as Gaussian filter [41] or median filter.

4.3 Other methods

4.3.1 Anisotropic diffusion

Anisotropic diffusion is a recursive method[38]. This method uses a diffusion formula to flatten the image, and the speed of diffusion is determined by its gradient while keep the pixels with high gradient (below).

$$\frac{dI}{dt} = \text{div}[g(\|\nabla I\|)\nabla I] \quad (29)$$

Where ∇I is gradient of image intensity, and g is usually a monotonically decreasing function which get smaller while input increases. This decreasing function ensures the diffusion speed will slow down near regions with high gradient value, thus preserving the image features.

Anisotropic diffusion and bilateral filter are not only similar in terms of edge-preserving; one paper [47] demonstrated that the original bilateral filter can also be implemented by a diffusion computation, but with a different penalty function for optimization than for anisotropic diffusion.

There are many researches about anisotropic diffusion, and here is a common model of anisotropic diffusion methods which use a Gaussian blur in the function g .

The idea of it is similar to the guidance image mentioned at Section 4.2.3.

$$\frac{dI}{dt} = \text{div}[g(\|\nabla(G_\sigma(I))\|)\nabla I] \quad (30)$$



4.3.2 Guided image filter

Guided image filter[37] is another edge-preserving smoothing method. Just like the joint bilateral filter, the input of this guided image filter are an input image and a guided image, and guided image can be the same image as the input image. This algorithm will approximate the input image by linearly converting the guided image within a window and then the output image intensity of target point will be the result of the linear transformation for all the windows that cover that point.

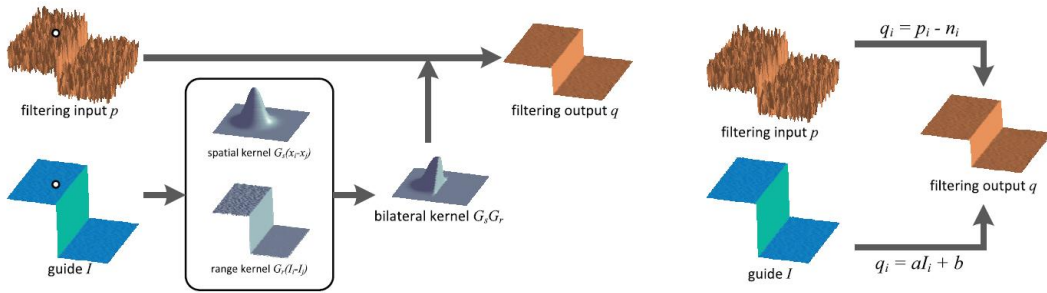


Fig. 4.2 A contrast between bilateral filter and Guided image filter. [37]

The formula of linear transformation, which transforms guided image I to output q in the window ω_k center at pixel k , and an error function to approximate this linear transformation to the input image p .

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (31)$$

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((q_i - p_i)^2 + \epsilon a_k^2) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2) \quad (32)$$

In which i and k mean the pixel indexes, and a_k and b_k are parameters of linear transform. The error function is the sum of two terms, the first term $(q_i - p_i)$ is the difference between the input and output, and the second term ϵa_k^2 is a regularization

parameter that penalizing large a_k because the gradient of output will be proportional to a_k , i.e. $\nabla q = a\nabla I$. Then the solution to this minimization problem is:

$$a_k = \frac{\text{cov}(I_{\omega_k}, p_{\omega_k})}{\text{var}(I_{\omega_k}) + \epsilon}, b_k = \text{mean}(p_{\omega_k}) - a_k \cdot \text{mean}(I_{\omega_k}) \quad (33)$$

where I_{ω_k} and p_{ω_k} mean the intersection of guided image and input image with the window ω_k . The final output is the average of the linear transformations of all windows containing pixel i . And because of the symmetry of the box windows ω_k , the equation can be simplified as follows.

$$q_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} (a_k \cdot I_i + b_k) = \bar{a}_i I_i + \bar{b}_i \quad (34)$$

where $\bar{a}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} a_k$ and $\bar{b}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} b_k$.

If the guided image is equal to input image ($I = p$), then the two linear approximation parameter will be $a_k = \text{var}(p_{\omega_k}) / (\text{var}(p_{\omega_k}) + \epsilon)$ and $b_k = (1 - a_k) \cdot \text{mean}(p_{\omega_k})$, therefore it can be deduced that:

$$\text{if } \text{var}(p_{\omega_k}) \gg \epsilon, a_k \approx 1, b_k \approx 0, (a_k \cdot I_i + b_k) \approx I_i \quad (35)$$

$$\text{if } \text{var}(p_{\omega_k}) \ll \epsilon, a_k \approx 0, b_k \approx \text{mean}(p_{\omega_k}), (a_k \cdot I_i + b_k) \approx \text{mean}(p_{\omega_k}) \quad (36)$$

Which means that if a window's variance value is large, then its average value doesn't take into consideration in the output calculation, thus reducing the smoothing effect around the edges.

Experiment in paper [37] shows that the performance of guided image filter is close to bilateral filter, and although guided image filter is non-linear, but it can use convolution in its calculations, which avoids the speed problem of the conventional bilateral filter.

Chapter 5 Proposed Bilateral filter



5.1 Algorithm

As previously introduced, bilateral filter can preserve edge features while eliminating noise. However, if the bilateral filter needs to eliminate noise with higher amplitude, the performance of the bilateral filter will become close to that of a normal Gaussian smoother, resulting in the edge part being smoothed out, especially those ridge-type and valley-type edges. In other words: formula of the range kernel cannot distinguish between noise and edge.

To solve this problem, the proposed method uses the ideas of joint bilateral filter and a weighted combination of original image and smoothed image. In the first part, we use a Gaussian smoothed image g_1 as guidance image of joint bilateral filter, and then we get a smoothed bilateral filter output y_1 . Below is a 1d case formula of y_1 .

$$y_1[n] = \frac{\sum_m [G_{\sigma_s}(|n-m|) \cdot G_{\sigma_r}(g_1[n] - g_1[m]) \cdot x[m]]}{\sum_m [G_{\sigma_s}(|n-m|) \cdot G_{\sigma_r}(g_1[n] - g_1[m])]} \quad (37)$$

In the last part, we set the output as the weighted combination of the bilateral filter output $y_1[n]$ and the input signal $x[n]$, and using the weight to make output dependent with difference between the noisy signal and another Gaussian smoothed image y_2 . The scale parameter of y_2 in Gaussian smoothing is 3/4 times of that in g_1 .

$$y[n] = w[n]x[n] + (1 - w[n])y_1[n] \quad (38)$$

$$w[n] = f(x[n] - y_2[n]), \quad f(x) = \frac{1}{2} + \frac{\tanh(\tau(|x| - x_0))}{2} \quad (39)$$

And that weight is decided by a threshold function f . If position n is in the smooth region, then $x[n]$ will be near to $y_2[n]$. In this case $w[n] \cong 0$, and the output $y[n]$ will be close to $y_2[n]$; If position n is in the edge, ridge, or valley region, then $x[n]$ will be

much different from $y_2[n]$. In this case $w[n] \cong 1$, and $y[n]$ is close to the original signal $x[n]$.



5.2 Experiment

Experiment of 1d case on two signals:

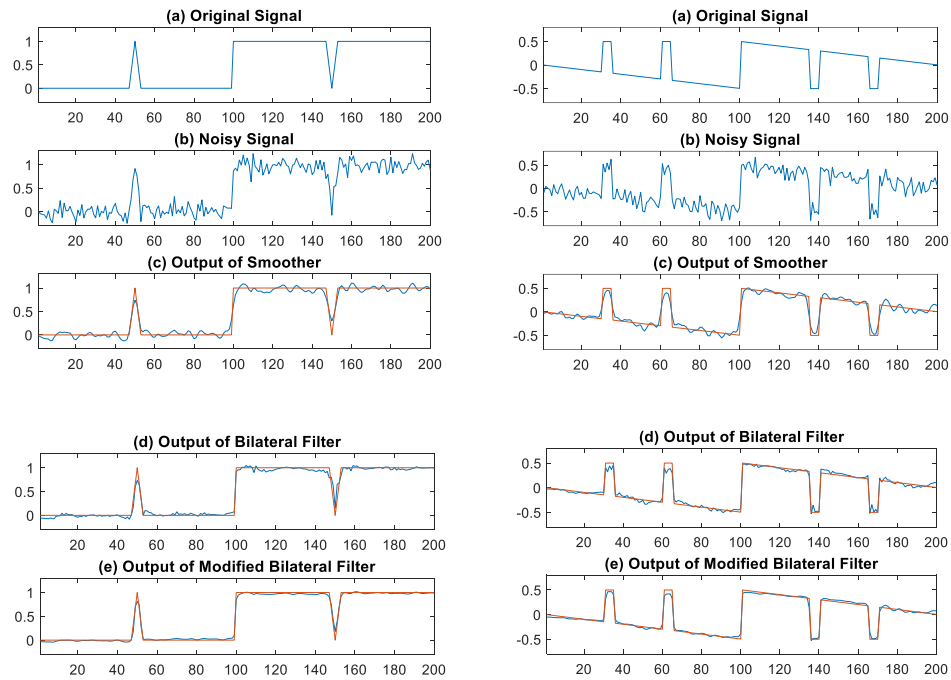


Fig. 5.1 Experiment of two 1D signals

And here are a simple 2D experiment, the image is from GUFU dataset[55].

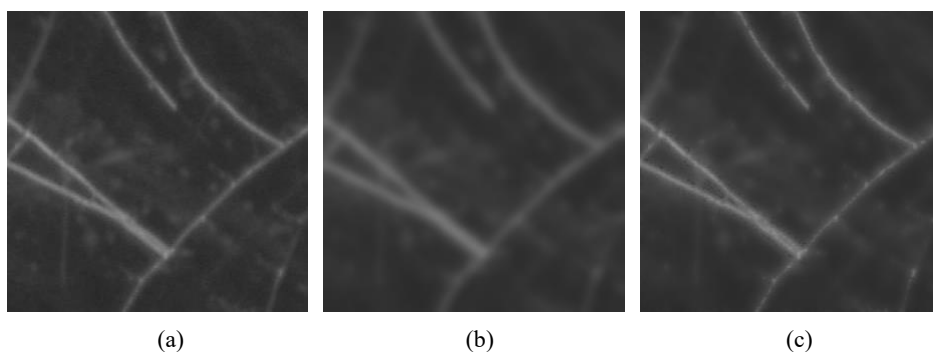


Fig. 5.2 Experiment of 2D image

(a) Original image (b) output of original bilateral filter (c) output of proposed method

As we can see, the proposed method works better than original bilateral filter.

Chapter 6 Proposed Ridge detector



6.1 Algorithm

6.1.1 Introduction

The proposed method is based on a previous thesis, “Improved Harris' Algorithm for Corner and Edge Detections”[26], which was based on a different approach than LoG to detect ridge. This method computes the variations in the directions horizontal and vertical to gradient, and then utilizes the variations as conditions for determining if a pixel is corner, edge, peak or ridge. The following figure Fig. 6.1 is a simple demonstration of how it works.

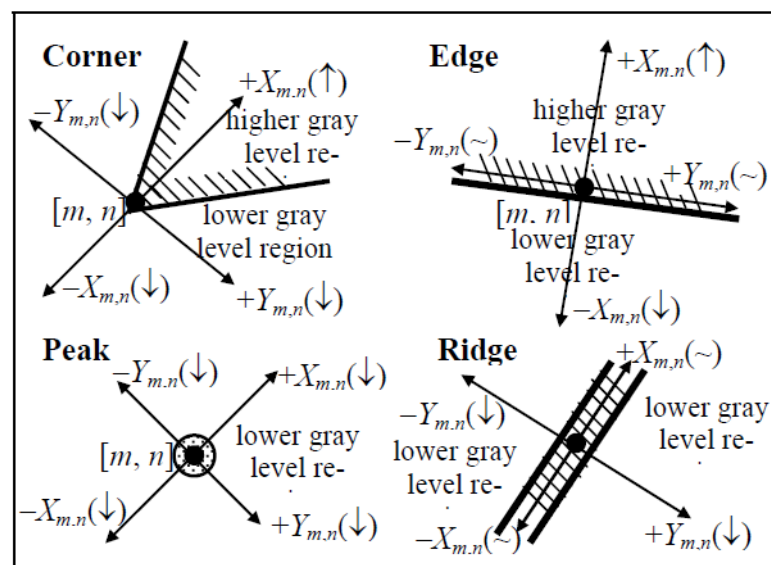


Fig. 6.1 Pixels classification in Improved Harris' Algorithm [26]

Problem of this method is that it doesn't automatically determine the width of ridge, Proposed method uses an adaptive filter to get the scale and direction information of ridges, then adopts the old method to detect ridge scores, and finally applying some post-processing methods to optimize the ridge map.

The general flow of the proposed method is listed below (next page).

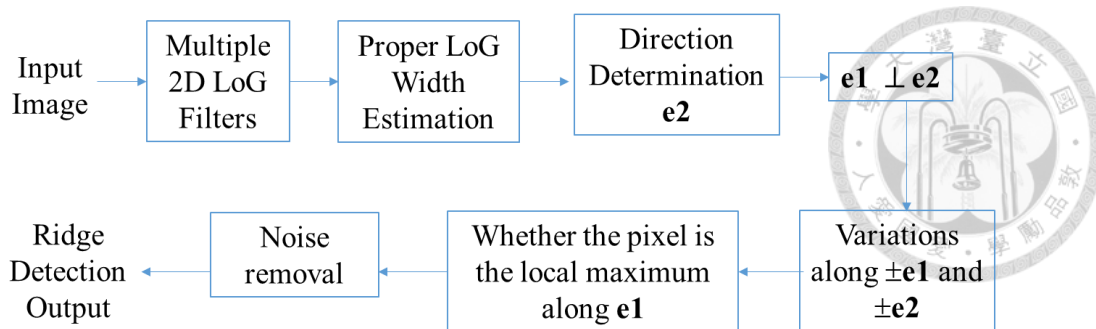


Fig. 6.2 Flow chart of proposed ridge detection method

6.1.2 Multiscale directional filter

In order to find the best scale size of ridge at target pixel, proposed method use an automatic scale selection by multiscale LoG filter. And Fig. 6.3 is the experiment results of ideal rectangular ridge(Fig. 6.3 (a), Fig. 6.4(a)) and triangle ridge(Fig. 6.3 (b), Fig. 6.4(b)) with width $w=10$ in 1d case, respectively.

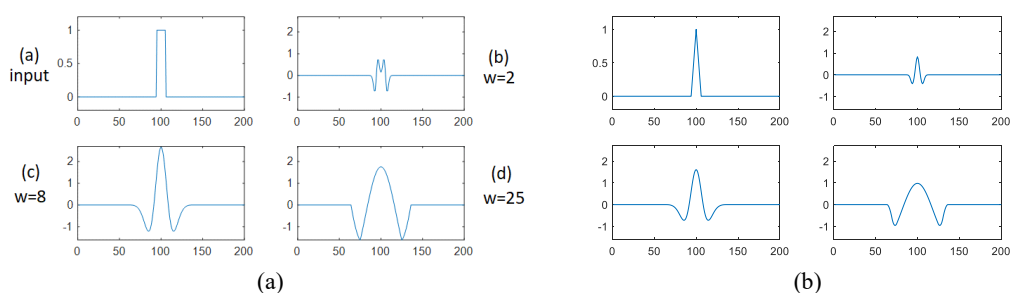


Fig. 6.3 The output on ridge with different LoG widths.

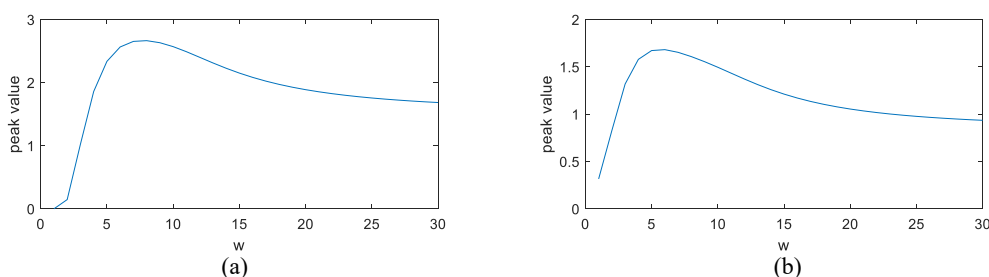


Fig. 6.4 Relation between LoG widths and center response

By applied LoG filters with different scale parameters, we found that the LoG filter will have a maximum on the center ridge if the scale parameter is proper.

The first part is automatic scale selection. We use four LoG filter with different widths as follow: $(w_1, w_2, w_3, w_4) = (3, 5, 8, 13)$, select the width with highest

intensity, and then decide the ridge direction and size in the following part by this width.

$$k[m, n] = \underset{k}{\operatorname{argmax}} \operatorname{abs}(L_k[m, n]), \quad L_{[m, n]} = \operatorname{round}(1.5w_{k[m, n]}) \quad (40)$$

and also, this ridge size parameter will be used in ridge direction measure:

$$\Phi[m, n] = \frac{m + jn}{\sqrt{m^2 + n^2}} \text{ if } -L \leq m, n \leq L \text{ and } [m, n] \neq [0, 0]$$

$$\Phi[m, n] = 0 \text{ otherwise} \quad (41)$$

$$\theta[m, n] = \angle(\Phi[m, n] * I[m, n]) \quad (42)$$

Where the * means convolution calculation and \angle means the argument of that complex number.

6.1.3 Ridge scoring

After knowing the angle of ridge direction, we can find out the four directions of variations:

$$\pm \mathbf{e}_1[m, n] = \pm(\sin(\theta[m, n]), -\cos(\theta[m, n])) \quad (43)$$

$$\pm \mathbf{e}_2[m, n] = \pm(\cos(\theta[m, n]), \sin(\theta[m, n])) \quad (44)$$

where $\pm \mathbf{e}_1$ are parallel to the ridge, and $\pm \mathbf{e}_2$ are perpendicular to the ridge. The next step is to calculate the four variations. The variation orthogonal to gradient direction ($\pm \mathbf{e}_1$) need to consider shorter distance.

$$v_{\pm 1}[m, n] = \frac{4}{L[m, n]} \sum_{c=1}^{L[m, n]/4} I[m, n] - I[[m, n] \pm \operatorname{round}(c\mathbf{e}_1)] \quad (45)$$

$$v_{\pm 2}[m, n] = \frac{1}{L[m, n]} \sum_{c=1}^{L[m, n]} I[m, n] - I[[m, n] \pm \operatorname{round}(c\mathbf{e}_2)] \quad (46)$$

If pixel $[m, n]$ is on a ridge-like region, then its variations should satisfy the next three conditions:

$$(i) \quad |v_2[m, n] + v_{-2}[m, n]| \gg |v_1[m, n] + v_{-1}[m, n]| \quad (47)$$

$$(ii) \quad \operatorname{sign}(v_2[m, n]) = \operatorname{sign}(v_{-2}[m, n]), \quad (48)$$

$$(iii) \quad |v_2[m, n] + v_{-2}[m, n]| \text{ should be sufficient large.} \quad (49)$$

So according to these requirements, the score algorithm will be as follows:

$$sc_0[m, n] = \begin{cases} 1 & \text{if } sign(v_2[m, n]) = sign(v_{-2}[m, n]) \\ 0 & \text{otherwise} \end{cases} \quad (50)$$

$$sc_1[m, n] = |v_2[m, n] + v_{-2}[m, n]| \quad (51)$$

$$sc_2[m, n] = \frac{|v_2[m, n] + v_{-2}[m, n]|}{10 + |v_1[m, n] + v_{-1}[m, n]|} \quad (52)$$

$$sc[m, n] = sc_0[m, n] \cdot sc_1^{0.8}[m, n] \cdot sc_2^{0.2}[m, n] \quad (53)$$

6.1.4 Local maximum and refinement

After determining the ridge score, we will perform local maximum along $\pm e_2$, which is done by the following function

$$(i) \quad sc[m, n] > sc[[m, n] + round(de_2)] \quad (54)$$

where $d = \pm 1, \pm 2, \dots, \pm 13$. The second is that $sc[m, n]$ should be large enough:

$$(ii) \quad sc[m, n] > 4mean(sc[m, n]) \quad (55)$$

Furthermore, we also remove the isolated segment. That is, if $count[m, n]$ is the number of pixels in the neighbor regions $\Omega: \{|m + a, n + b| - 4 \leq a, b \leq 4\}$ which satisfy the two constraints (i) and (ii), then we need to make sure $count[m, n]$ is also large enough so that it will not be isolated segment, so the third condition is:

$$(iii) \quad count[m, n] \geq 16 \quad (56)$$

The final ridge pixels will be those that satisfy (i), (ii), and (iii).

6.2 Experiment

6.2.1 Scoring method

In many papers[9], F-measure is used as an evaluation method for the edge score. For an edge detection result, if we set the number of ground truth pixels to N_{gt} , the number of detected edge pixels to N_d , and the number of true detected edge pixels to

N_{td} , then the accuracy of this detection will be [2]:

$$\text{Precision} = \frac{N_{td}}{N_d} \quad (57)$$

$$\text{Recall} = \frac{N_{td}}{N_{gt}} \quad (58)$$



In a nutshell, precision is the accuracy among the edge pixels detected by the target method, and recall is the accuracy among the real edge pixels provided by ground truths.

And F-measure is the harmonic mean of them.

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (59)$$

However, this method is more suitable for higher precision detection, and we found that detected ridge pixels usually have small positional deviations from ground truth, so we propose a measurement method that allows for positional deviations.

The original F-measure depends only on the true detected number N_{td} and does not take into account the distance between detected ridge pixels and ground truth. Therefore, we replace N_{td} with summation of a distance-based functions, $e^{-\lambda \cdot \text{dist}(p, M)}$, where $\text{dist}(p, M)$ means the minimum distance between pixel p and map M , which will be one if detected ridge pixels is on ground truth and decreases with distance from ground truth. Based on this function, new precision and recall function are proposed.

$$\text{precision}(D, R) = \frac{1}{|D|} \sum_{d \in D} e^{-\lambda \cdot \text{dist}(d, R)}$$

$$\text{recall}(D, R) = \frac{1}{|R|} \sum_{r \in R} e^{-\lambda \cdot \text{dist}(r, D)}$$

And then we can calculate new F-measure score base on this method.

6.2.2 Dataset

Three datasets, GUFU[55], RITE[57] and Sketch[54], are used for this part of the

experiments. They are datasets of fungi, retinal blood vessels, and human sketch images, respectively.

GUFI (Ghent University Fungal Images) is a dataset of fungi images, and because the mycelium in fungi images (Fig. 6.5 (a)) is similar to ridges, it was used as a database for ridge detection, and the ridge groundtruths (Fig. 6.5(b)) were also provided in this dataset.

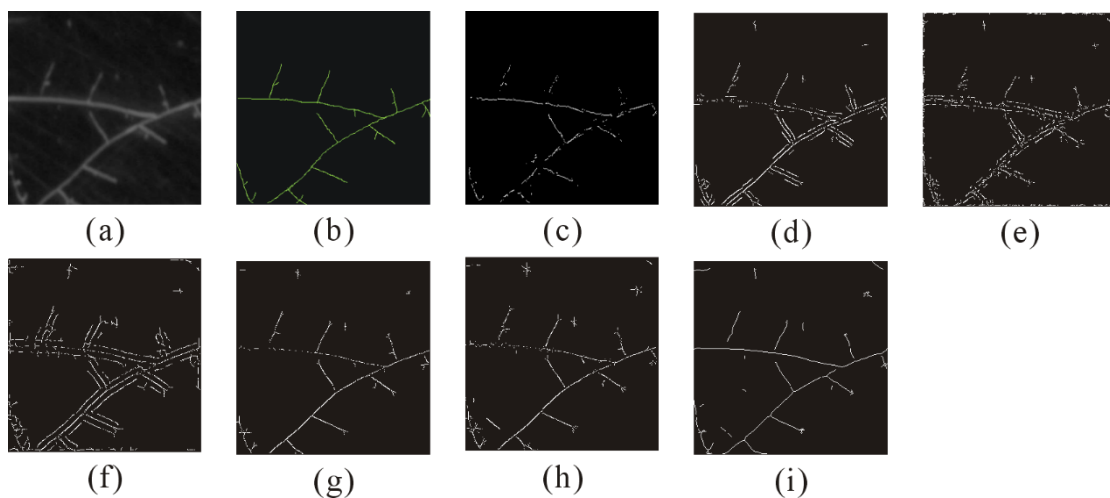


Fig. 6.5 Example of GUFI. (a) fungi images (b) groundtruths (c)-(i) ridge outputs.

RITE (Retinal Images vessel Tree Extraction) is a dataset of retinal image with vessels groundtruths inherited from DRIVE dataset[56], and then since the retinal vessel are not 1-pixel width, we used a thinning algorithm [58] to thinner them.

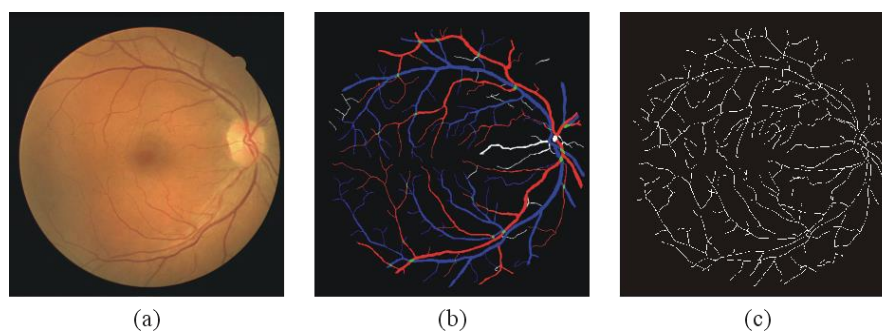


Fig. 6.6 Example of RITE. (a) the retinal vessels image (b) the vessels groundtruth (c) thinner ridge ground truth.

Sketch dataset is composed of human sketch of different items. We take out a part

of the images in dataset, and use them as experimental input images after adding noise and as groundtruths after thinning by the same algorithm [58], respectively. And since the dataset is already images with simple thin lines, I added noise to images and experimented twice with different noise intensities.

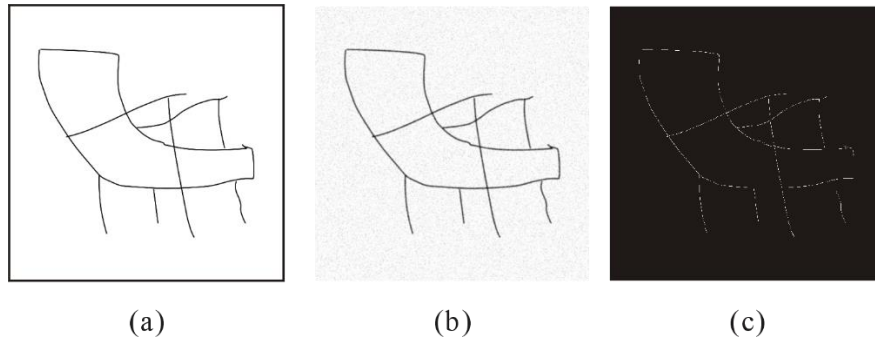


Fig. 6.7 Example for Sketch (a) the original sketch image (b) the noisy image (c) thinned image(as groundtruth).

6.2.3 Experiment Result

Table 6.1 Comparison of evaluation scores

| Method | GUFI[55] | RITE[57] | Sketch[54] $n_i = 0.2$ | Sketch $n_i = 0.4$ |
|-----------|----------|----------|---------------------------|-----------------------|
| Proposed | 0.5153 | 0.1813 | 0.6567 | 0.6009 |
| Hessian | 0.4556 | 0.3942 | 0.4595 | 0.4360 |
| SGSF[32] | 0.3883 | 0.3145 | 0.2529 | 0.2657 |
| SF[27] | 0.3820 | 0.3342 | 0.4065 | 0.4276 |
| SOAGK[22] | 0.4951 | 0.4036 | 0.5044 | 0.5173 |
| SCIRD[24] | 0.5563 | 0.4433 | 0.6348 | 0.6424 |
| SymFD[23] | 0.5504 | 0.4063 | 0.7830 | 0.7345 |

where n_i in Sketch dataset means the noise intensity.

We can also see that the performance of the proposed method on datasets GUFI and Sketch is close to that of SCIRD and SymFD, which is better among all the methods. SCIRD and SymFD are two complete programs provided by the author of their papers, not my own programs that I have implemented according to other authors' papers, so they contain complete ridge optimization like multi-scale and post-processing.

Therefore, it is good to be close to these two methods in terms of performance. Besides, we can see that running time of proposed method is much faster than SCIRD and SymFD by Table 6.2.

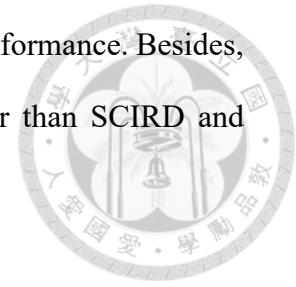


Table 6.2 Average running time in dataset GUF1

| Method | Running time (seconds) |
|-----------|------------------------|
| Proposed | 0.1648 |
| Hessian | 0.0304 |
| SGSF[32] | 0.0459 |
| SF[27] | 0.0705 |
| SOAGK[22] | 3.0292 |
| SCIRD[24] | 0.5934 |
| SymFD[23] | 0.4976 |

However, the performance of proposed method on RITE dataset is the worst. Observing the ridge maps, we can find that proposed method is very poor at detecting the thinner and lighter ridge, while the images in RITE distributes a large number of vessels that satisfy that condition. In terms of precision and recall score, we can also find that the precision of the proposed method is very high while the recall is low, which means that although the ridges are detected with a high degree of conformity, many pixels of the ridges are missed.

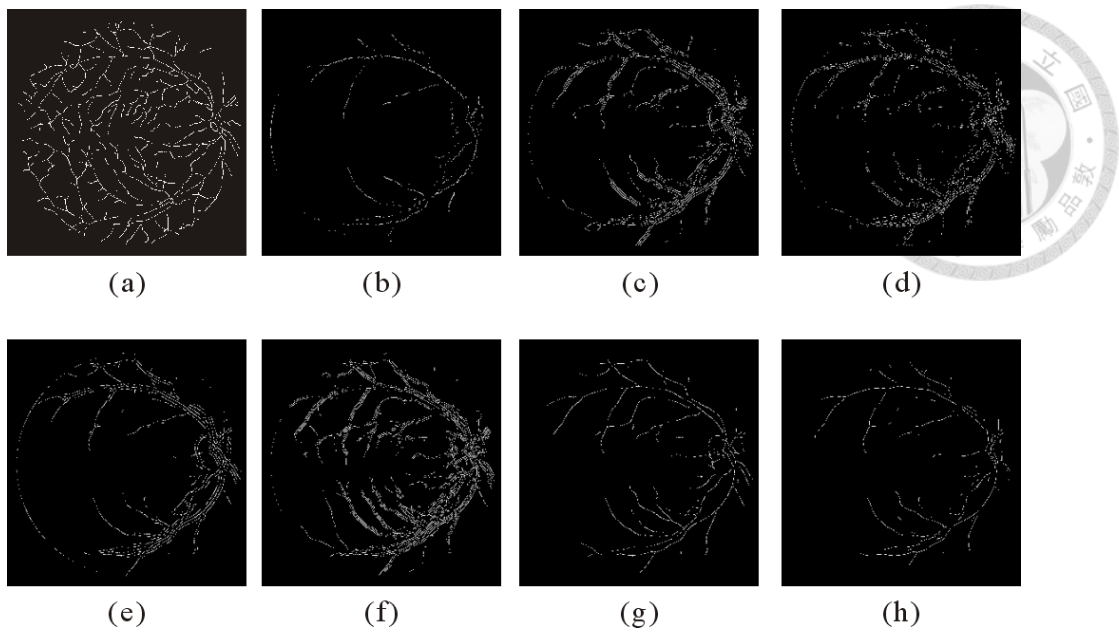


Fig. 6.8 Example of RITE. (a) groundtruth (b) proposed method (c)-(h) other methods

Table 6.3 Comparison of evaluation scores on RITE

| Method | F-measure | Precision | Recall |
|-----------|-----------|-----------|--------|
| Proposed | 0.1813 | 0.8833 | 0.1045 |
| Hessian | 0.3942 | 0.5083 | 0.3412 |
| [32] | 0.3145 | 0.514 | 0.2414 |
| [27] | 0.3342 | 0.5841 | 0.2453 |
| SOAGK[22] | 0.4036 | 0.461 | 0.4122 |
| SCIRD[24] | 0.4433 | 0.884 | 0.3045 |
| SymFD[23] | 0.4063 | 0.8415 | 0.2773 |

Chapter 7 Conclusion and future work



7.1 Conclusion

In our first proposed method, we applied smoother image as guidance image in range kernel of bilateral filter to fix the noise problem, and then use a weight combination to solve the ridge over-smoothed problem. We also further demonstrated its effectiveness on 1D experiment and 2D demonstration.

In another proposed ridge method, we applied multiple 2D LoG filters to find the optimal filter width, so that we can adaptively determine ridge width and direction of all pixels, and then we refine the results by post processing methods. We also proposed a new ridge scoring method to calculate our ridge maps, and then prove that proposed method can approach those good multiscale ridge detectors with a faster speed by experiment. Although experiments also show our method will not work as well as other methods that in the dataset of complex ridge images.

7.2 Future work

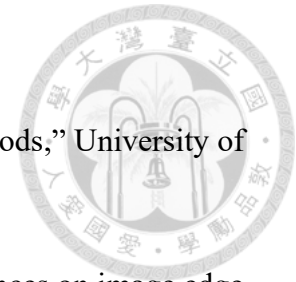
The proposed bilateral filter is only a modification of the joint bilateral filter based on the ridge feature, so it may be possible to improve it based on other features.

Experimental results on our ridge method may indicate that there is room for improvement of our multiscale directional filter, especially on test data with large width variations of ridges. How to make improvements in this area and to propose a better method for ridge scale measurement will be the focus in the future. In addition, the experiment did not consider deep learning methods as I was unable to find relevant papers, so it would have been better to have deep learning methods as a counterpart.

In addition to our proposed methods, our proposed ridge scoring method also has

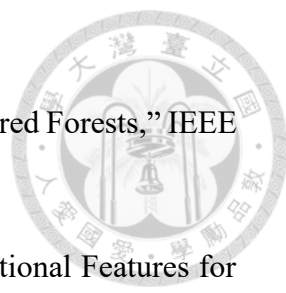
some problems. Although it can allow for positional shifts, because pixels are scored as long as there are neighboring pixels, it is difficult to penalize the discontinuity of ridge lines and false detections of multiple neighboring parallel ridge lines. While the former may be solved by complex morphological methods based on ridge line direction and distance between endpoints, the latter may have to be approached from the mathematical side, for example Canny talked about the one-to-one problem of edges in his paper on edges[4]. Changing the scoring method to a one-to-one method might also be a solution, so that unmatched pixels do not add to the total score.

REFERENCE



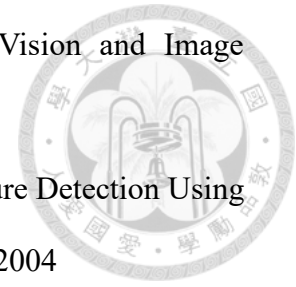
- [1] M. A. Oskoei and H. Hu, “A Survey on Edge Detection Methods,” University of Essex, UK, 2010
- [2] J. Jing, S. Liu, G. Wang, W. Zhang and C. Sun, “Recent advances on image edge detection: A comprehensive review,” *Neurocomputing*, Volume 503, Pages 259-271, September 2022
- [3] D. Ziou and S. Tabbone, “Edge detection techniques: An overview,” *Pattern Recognition and Image Analysis C ...*, 1998
- [4] J. Canny, “A Computational Approach to Edge Detection,” *IEEE PAMI*, pp 679 - 698, November 1986
- [5] F. A. Pellegrino, W. Vanzella and V. Torre, “Edge detection revisited,” *IEEE Transactions on Cybernetics*, pp 1500-1518, 2004
- [6] F. Russo, “Edge detection in noisy images using fuzzy reasoning,” *IEEE Instrumentation and Measurement Technology Conference*, May 1998
- [7] D. Marr and E. Hildreth, “Theory of edge detection,” *Proceedings of the Royal Society of London. Series B, Biological Sciences*, Vol. 207, No.1167, pp.187-217, Feb. 29, 1980
- [8] M.Y. Shih and D.C. Tseng, “A wavelet-based multiresolution edge detection and tracking,” *Image and Vision Computing*, Vol 23, Issue 4, pp. 441-451, April 2005
- [9] D.R. Martin, C.C. Fowlkes and J. Malik, “Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues,” *IEEE PAMI*, Volume 26, Issue 5, pp. 530 – 549, May 15, 2004
- [10] P. Isola, D. Zoran, D. Krishnan and E.H. Adelson, “Crisp Boundary Detection Using Pointwise Mutual Information,” *Computer Vision – ECCV 2014*, pp 799–

814, 2014

- 
- [11] P. Dollár and C.L. Zitnick, “Fast Edge Detection Using Structured Forests,” IEEE PAMI, Volume: 37, pp 1558 – 1570, Aug 2015
- [12] Y. Liu, M. Cheng, X. Hu, K. Wang, X. Bai; “Richer Convolutional Features for Edge Detection,” CVPR, pp. 3000-3009, 2017
- [13] M. Pu, Y. Huang, Y. Liu, Q. Guan and H. Ling, “EDTER: Edge Detection With Transformer,” CVPR, pp. 1402-1412, 2022
- [14] S. Yi, D. Labate, G. R. Easley and H. Krim, “A Shearlet Approach to Edge Analysis and Detection,” IEEE transactions on image processing, VOL. 18, NO. 5, MAY 2009
- [15] Tzu-Heng Henry Lee, “Edge Detection Analysis,” NTU DISP Lab, 2007, <https://disp.ee.ntu.edu.tw/research.php> (file not available now)
- [16] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv preprint arXiv:1409.1556, 2014
- [17] S. Xie and Z. Tu, “Holistically-nested edge detection,” CVPR, pp. 1395–1403, 2015
- [18] J. He, S. Zhang, M. Yang, Y. Shan and T. Huang, “Bi-directional cascade network for perceptual edge detection,” CVPR, pp. 3828–3837, 2019
- [19] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen and L. Liu, “Pixel Difference Networks for Efficient Edge Detection,” International Conference on Computer Vision (ICCV), pp. 5117-5127, 2021
- [20] C. Zhou, Y. Huang, M. Pu, Q. Guan, L. Huang, H. Ling, “The Treasure Beneath Multiple Annotations: An Uncertainty-aware Edge Detector,” CVPR, pp. 15507-15517, 2023
- [21] G. S. Shokouh, B. Magnier, B. Xu and P. Montesinos, “An Objective Comparison of Ridge/Valley Detectors by Image Filtering,” Pattern Recognition. ICPR

- International Workshops and Challenges, pp 182–197, February 2021
- [22] C. Lopez-Molina, G. D. Ulzurrun and J. M. Baetens, “Unsupervised ridge detection using second order anisotropic Gaussian kernels,” *Signal Processing*, Volume 116, pp 55-67, November 2015
- [23] R. Reisenhofer and E.J. King, “Edge, Ridge, and Blob Detection with Symmetric Molecules” *SIAM Journal on Imaging Sciences*, Vol 12, Issue 4, 2019
- [24] R. Annunziata, A. Kheirkhah, P. Hamrah and E. Trucco “Scale and Curvature Invariant Ridge Detector for Tortuous and Fragmented Structures,” *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp 588–595, November 2015
- [25] C. Harris and M. Stephens, “A combined corner and edge detector,” *Alvey vision conference*, 1988
- [26] S. C. Pei and J. J. Ding, “Improved Harris' Algorithm for Corner and Edge Detections”, *IEEE International Conference on Image Processing*, 2007
- [27] A. Jose, S. R. Krishnan and C. S. Seelamantula, “Ridge detection using Savitzky-Golay filtering and steerable second-order Gaussian derivatives,” *2013 IEEE International Conference on Image Processing*, 2013
- [28] G. Norgard and P.T. Bremer, “Ridge–Valley graphs: Combinatorial ridge detection using Jacobi sets,” *Computer Aided Geometric Design*, 2013
- [29] A.M. Lopez, F. Lumbreras, J. Serrat and J. J. Villanueva, “Evaluation of Methods for Ridge and Valley Detection,” *IEEE PAMI*, Volume 21, Issue 4, pp 327 – 335, April 1999
- [30] Z. Qiu, L. Yang, W. Lu, “A new feature-preserving nonlinear anisotropic diffusion for denoising images containing blobs and ridges,” *Pattern Recognition Letters*, Vol 33, Issue 3, pp 319-330, 2012
- [31] A. M. López, D. Lloret, J. Serrat and J. J. Villanueva “Multilocal Creaseness

Based on the Level-Set Extrinsic Curvature” Computer Vision and Image Understanding, Vol 77, Issue 2, pp 111-144, February 2000



- [32] M. Jacob and M. Unser, “Design of Steerable Filters for Feature Detection Using Canny-Like Criteria,” IEEE PAMI, VOL. 26, NO. 8, August 2004
- [33] C. Tomasi and R. Manduchi, “Bilateral Filtering for Gray and Color Images,” Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), January 1998
- [34] Paris, S., Kornprobst, P., Tumblin, J., Durand, F., “Bilateral filtering: Theory and applications,” Foundations and Trends in Computer Graphics and Vision, 2009, 4, pp. 1-73
- [35] C. Pal, A. Chakrabarti and R Ghosh, “A Brief Survey of Recent Edge-Preserving Smoothing Algorithms on Digital Images,” Computer Vision and Pattern Recognition, 2015
- [36] R.G. Gavaskar, K.N. Chaudhury; “Fast Adaptive Bilateral Filtering,” IEEE transactions on image processing, VOL. 28, NO. 2, FEBRUARY 2019
- [37] K. He, J. Sun, X. Tang; “Guided Image Filtering,” IEEE PAMI, VOL. 35, NO. 6, JUNE 2013
- [38] M. J. Black, D. H. Marimont; “Robust Anisotropic Diffusion,” IEEE transactions on image processing, VOL. 7, NO. 3, MARCH 1998
- [39] Zhang, B., Allebach, J., ‘Adaptive bilateral filter for sharpness enhancement and noise removal’, IEEE Trans. Image Processing, 2008, 17, pp. 664-678
- [40] Bhonsle, D., Chandra, V., Sinha, G., ‘Medical image denoising using bilateral filter’, Int. J. Image, Graphics and Signal Processing, 4(6), article 36, 2012
- [41] Chen, B., Tseng, Y., & Yin, J., ‘Gaussian-adaptive bilateral filter’, IEEE Signal Processing Lett., 2020, 27, pp. 1670-1674.
- [42] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-

range images,” ACM Siggraph, vol. 21, pp. 257 – 266, 2002.

- [43] J. Chen, S. Paris and F. Durand, “Real-time Edge-Aware Image Processing with the Bilateral Grid,” ACM Transactions on Graphics, Vol. 26, Issue. 3, pp. 103–es, 2007
- [44] S. Paris and F. Durand, “A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach,” Computer Vision–ECCV 2006, pp. 568–580, 2006
- [45] E. Eisemann and F. Durand, “Flash photography enhancement via intrinsic relighting,” ACM Transactions on Graphics, vol. 23, no. 3, pp. 673 – 678, July 2004.
- [46] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, “Digital photography with flash and no-flash image pairs,” ACM Transactions on Graphics, vol. 23, no. 3, pp. 664 – 672, 2004.
- [47] M. Elad, “On the Origin of the Bilateral Filter and Ways to Improve It,” IEEE transactions on image processing, Vol. 11, No. 10, October 2002
- [48] R. C. Gonzalez and R. E. Woods, “Digital Image Processing,” 4th ed., Pearson, 2018.
- [49] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," Int. J. Computer Vision, vol. 30, pp. 117-156, 1998.
- [50] G. S. Shokouh, B. Magnier, B. Xu, and P. Montesinos, “Ridge detection by image filtering techniques: A review and an objective analysis,” Pattern Recognition and Image Analysis, vol. 31, pp. 551-570, 2021.
- [51] S. Koka, K. Anada, K. Nomaki, K. Sugita, K. Tsuchida, and T. Yaku, “Ridge detection with the steepest ascent method,” Procedia Computer Science, vol. 4, pp. 216-221, 2011.
- [52] H. F. Alhasson, C. G. Willcocks, S. S. Alharbi, A. Kasim, and B. Obara, “The relationship between curvilinear structure enhancement and ridge detection

- methods,” *The Visual Computer*, vol. 37, pp. 2263-2283, 2021.
- [53] S. Y. Pen, Y. C. Lee, I. W. Wu, C. C. Lee, C. C. Sun, J. J. Ding, C. F. Liu, and L. Yeung, “Impact of blood pressure control on retinal microvasculature in patients with chronic kidney disease,” *Scientific Reports*, vol. 10, article 14275, pp. 1-10, 2020.
- [54] M. Eitz, J. Hays, M. Alexa, “How do humans sketch objects?” *ACM Transactions on graphics (TOG)*, Volume 31, Issue 4, Article No. 44, pp 1–10, 2012
- [55] Baetens, Jan, 2015, “Ghent_University_Fungal_Images,” <https://doi.org/10.13140/RG.2.1.4441.1607>.
- [56] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, B. van Ginneken, “Ridge-based vessel segmentation in color images of the retina,” *IEEE transactions on medical imaging*, Vol 23, Issue 4, pp 501-509, April 2004, <https://doi.org/10.1109/TMI.2004.825627>
- [57] Q. Hu, M. D. Abramoff, M. K. Garvin, “Automated Separation of Binary Overlapping Trees in Low-Contrast Color Retinal Images,” *MICCAI 2013*, pp 436–443, 2013, https://doi.org/10.1007/978-3-642-40763-5_54
- [58] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Comm. ACM*, vol. 27, no. 3, pp. 236-239, 1984
- [59] C. Topal and C. Akinlar, “Edge Drawing: A combined real-time edge and segment detector,” *Journal of Visual Communication and Image Representation*, Volume 23, Issue 6, pp 862-872, August 2012
- [60] M. Pu, Y. Huang, Q. Guan and H. Ling, “RINDNet: Edge Detection for Discontinuity in Reflectance, Illumination, Normal and Depth”, *ICCV*, pp. 6879-6888, 2021
- [61] T. Huang, G. Yang and G. Tang, “A fast two-dimensional median filtering algorithm,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*,

Volume 27, Issue 1, February 1979

