# EDGE DETECTION OPERATORS FOR X-RAY IMAGES BASED ON HESSIAN MATRICES

by

Wanru Jia

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
December 2020

*To my daughter, my husband and my grandmother.*

# Table of Contents

# List of Figures

## Abstract

This thesis developed two sophisticated statistical methods for edge detection in X-ray images. Both of the two methods are developed on the basis of the Hessian matrices of brightness. The first method evaluates the possibility of each pair of pixels in the X-ray images being on the intended edges by assigning a calculated goodness score to each pair. A comparison shows that this method outperforms the state-of-the-art edge detection methods for X-ray images. To further improve the quality of edge detection, another method was developed to obtain higher accuracy and precision by getting thinner and more continuous edges. The second method carries out edge detection by tracing the progression of edges from certain starting points, in this way, the edges detected are clearer, thinner, and more continuous curves. For edge detection for the hip joint part that we mostly focused on, the second method generates better results than the first method. Although the methods are mainly applied to X-ray images in this thesis, the methods are generally applicable to all other images as well.

## Acknowledgements

I would like to thank my supervisor, Dr. Hong Gu, and my co-supervisor, Dr. Toby Kenney, for the extensive support, patience, and guidance I have received from them during my Master's study. I would also like to thank my committee members, Dr. Lam Ho and Dr. Andrew Irwin for their continuous help and advice.

I am also very grateful to my group members, who are not only my colleagues at work generously giving me support and help, but also lovely friends of me and my family, always being there for me and bringing happiness to my life.

I would like to extend my sincere thank to Dr. Ivan Wong and his group for providing us with data and constructive guidance. I am also thankful to Compute Canada for the technical support.

# Chapter 1

# Introduction

## 1.1 Research Background

Medical imaging is becoming a powerful tool for diagnosis and treatment of numerous medical conditions. Common medical images include computed tomography, fluoroscopy, and radiography. As one type of radiography, X-ray imaging has been utilized for more than 100 years and remains valuable. X-ray image analysis is conventionally carried out by specialists and thus requires great human resources. To improve the precision and efficiency of X-ray image analysis, a lot of effort has been made to automate the image processing.

There is a trend to introduce general image analysis techniques, such as edge detection and image segmentation, into X-ray image analysis. Figure 1.1 shows two state-of-the-art X-ray image analysis algorithms. The first algorithm [24] aims to detect fractures from X-ray images. An important step of this method is edge detection and it uses the Canny edge detector. The second algorithm [22] is designed to detect defects and the Sobel edge detection method plays a crucial role.

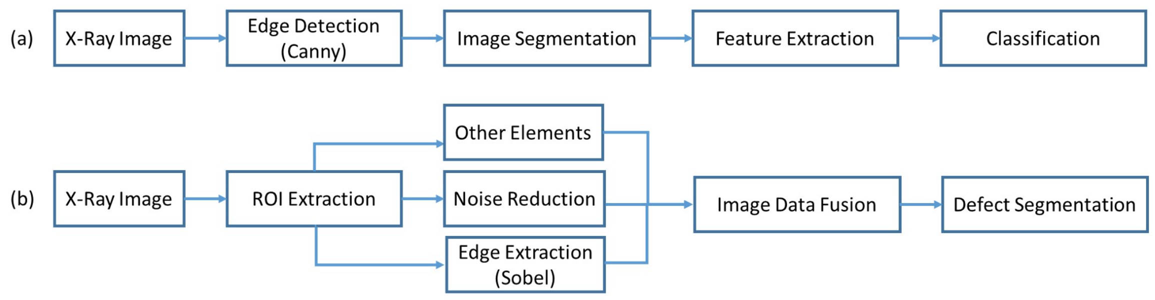

Figure 1.1: The general algorithms of X-ray image analysis

The X-ray images of this work are for osteoarthritis diagnosis. Osteoarthritis can cause severe joint pain and stiffness and often begins to affect people in middle age or older. Early treatment and detection of osteoarthritis is important because osteoarthritis worsens over time and early diagnosis may have the potential to slow its progression. This research will begin with a study on osteoarthritis of the hip. Osteoarthritis of the hip can be classified into grade 0 to 4 based on severity. Classification based on X-ray imaging prior to surgical intervention is important as it leads to better treatment decisions. Currently, classification is based on physicians judgement. Physicians classify the grades by measuring the distances between joint spaces and defining how clear the boundaries of the bones are. Unfortunately, it is time consuming and arbitrary since different physicians have different opinions on how clear or vague the boundaries are. So, we aim to develop a carefully designed method which can offer an accurate edge detector for X-Ray images.

## 1.2 Edge Detection

Edge detection is a traditional topic in image processing, used for identifying the contours of objects in images. Edges include the most crucial structures in the images [14]. Edge detection is very useful for problems such as image segmentation, reconstruction, interpretation, tracking [10] and data extraction. For numerous fields of study, edge detection is an important step to unveil the features underneath the original images. An accurate and robust edge detection methodology can be applied in many interdisciplinary scenarios. For example, in medicine, edge detection is useful for medical condition diagnosis. The applications shown in this work are mainly medical.

Edge detection has an extremely rich history. There are many algorithms for edge detection such as the Canny [2], the Sobel operator [20], the Marr-Hildreth Algorithm [23], and some deep learning methods. These methods can be classified into two groups as traditional statistical algorithms and deep learning based algorithms. Chapter 2 of this thesis is a relatively comprehensive review on edge detection methods.

It is difficult to find a general edge detection method that suits well for many contexts or different requirements. We initially tried the state-of-the-art computer vision method - the Canny edge detector on our X-ray images. But it could not catch the edges very well. In order to make an accurate edge detection with less noise, we develop two edge detection algorithms. More specifically, we use eigenvalues and eigenvectors of the Hessian matrix as an important indicator of the change of brightness. The first algorithm considers the second derivatives of the brightness with two more elements, angle and distance. The second algorithm, on the other hand, aims to trace the edge using an objective function based on the Hessian matrix. Both of them show a better result than Canny [2] and HED [25] on the images we have tested.

## 1.3 Thesis Structure

We review the existing edge detection algorithms in Chapter 2. Then, the first edge detection algorithm is proposed and tested in Chapter 3. Chapter 4 discusses the second edge detector and shows a comparison between the state-of-the-art methods and our method. Finally, Chapter 5 concludes the work and discusses the implications of the edge detection methods in other interdisciplinary fields and suggests future works.

# Chapter 2

# A Review of Edge Detection Methods

## 2.1 Edge Detection Problem

The raw data of our edge detection problem is the brightness of the pixels on X-Ray images, here $b(x, y)$ is used as the notation of the brightness at the corresponding coordinates $(x, y)$ of each pixel. An edge detector is intrinsically designed to find the sharp change of the brightness. Derivative is a suitable tool to depict the change. Figure 2.1 is an example of how brightness and derivatives change at the edge area (the area between the dotted lines) in one dimension. The brightness changes from dark to bright, the first-order derivative forms a bell shape and reaches a peak in the middle. The second-order derivative has two bumps within the edge area. These features of derivatives are good clues to detect the edge. The Canny edge detector is based on the first-order derivative of the brightness. The edges detected are not clear enough for X-ray images. The methods we develop are based on the second derivative.

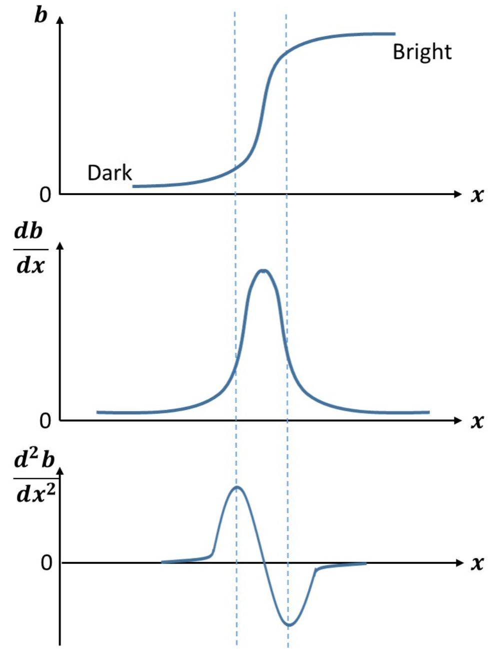

Figure 2.1: Illustration of changes of brightness, derivatives of brightness at the edge area in one dimension.

This figure is a simplified one dimensional example with a known direction of brightness change. However, for our two dimensional image, the brightness change could be in any direction. So, we need a vector to capture the direction of the brightness change. Here, we introduce the eigenvectors and eigenvalues of the Hessian matrix, which is a matrix of second-order derivatives, of brightness to detect the sharp change. The Hessian matrix can be written as

$$\boldsymbol{H}(x,y) = \begin{bmatrix} \frac{\partial^2 b}{\partial x^2} & \frac{\partial^2 b}{\partial x \partial y} \\ \frac{\partial^2 b}{\partial y \partial x} & \frac{\partial^2 b}{\partial y^2} \end{bmatrix}$$

Since our X-Ray images are converted to greyscale images, $b$ of each pixel ranges from 0 to 255. And the coordinates of the pixels on an image vary from 0 to a positive integer, so the brightness is a function $b : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}_0$.

## 2.2 Algorithms for Edge Detection

Since edge detection can be used in many different fields for different purposes, it is hard to design a general method that works well in multiple contexts and define a fixed set of parameters needed for the detector. As a result, there are lots of variations of edge detectors. We group them into two categories: traditional statistical algorithms and deep learning based algorithms.

### 2.2.1 Traditional Statistical Algorithms

The methods of traditional statistical algorithms can be separated into first-order derivative based and second-order derivative based algorithms. The most popular first-order derivative based algorithm is the Canny edge detector which was developed by John F. Canny in 1986 [2]. First of all, the Canny edge detector smooths the images to reduce noise by applying a Gaussian filter kernel with size $(2k+1)\times(2k+1)$ namely:

$$F_{ij} = \frac{1}{2\pi\sigma^2} exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \leq i,j \leq (2k+1)$$

Then, it detects the directions and gradients on the edge using edge detection operators such as Roberts [17], Sobel [21], or Prewitt [16] as shown below.

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\Theta = atan\left(\frac{G_y}{G_x}\right)$$

where $G_x, G_y$ are the first derivatives of the brightness on the horizontal and vertical directions respectively. The following steps are thinning the edge using non-maximum suppression, filtering out points with weak gradients, and tracking the edge by hysteresis.

Although the Canny edge detector was created more than forty years ago, it is widely used nowadays since it is a simple and accurate method. And in order to fulfill more requirements of edge detection, there are some variations of Canny edge detectors. For example, replacing the Gaussian filter by an adaptive filter can improve the accuracy [7], Otsu's method can help to calculate a more robust high threshold [18], Mallat and Zhong improved the edge thinning step through a reconstruction algorithm [11], and the Curvelet technology was used to reduce the noise in the first two steps of the Canny algorithm [4].

The Marr-Hildreth algorithm [12] is the first use of second-order derivatives for edge detection. The accuracy of this method is not competitive on overall quality of results, but a number of refinements added since the original method was introduced have made it competitive. In 1998, Lindeberg developed an edge detector [9] using scale selection [8] and using a Gaussian filter for smoothing. For any pixel $(x_0, y_0)$, they set a local coordinate $(u, v)$ where $v$ is parallel to the gradient direction. Then, they introduce a scale $t$ to form a scale space representation:

$$L(x, y; t) = g(x, y; t) \times b(x, y)$$

where $g(x, y; t)$ is a Gaussian kernel, $b(x, y)$ is the brightness of the pixel, and the scale parameter $t$ is selected from local maxima over scales of $\gamma$-normalized derivatives [9].

The points on the edge are where the gradient magnitudes are the maximum in the gradient direction. That is, the second-order directional derivative in the $v$-direction is zero and the third-order derivative is negative:

$$L_{vv} = 0,$$

$$L_{vvv} < 0.$$

Next, they restated the above two conditions under the location coordinates $(x, y)$ as

$$\tilde{L}_{vv} = L_v^2 L_{vv} = L_x^2 L_{xx} + 2L_x L - yL_{xy} + L_y^2 L_{yy} = 0,$$

$$\tilde{L}_{vvv} = L_v^3 L_{vvv} = L_x^3 L_{xxx} + 3L_x^2 L_y L_{xxy} + 3L_x L_y^2 L_{xyy} + L_y^3 L_{yyy} < 0.$$

These two conditions define the edge at scale $t$. Along with the Gaussian filter, the edges can be detected automatically. Following this work, several more second-order derivative based algorithms, also based on the Gaussian filter, have been developed, such as Statistical Edge Detection [6] and Pb [13].

### 2.2.2 Deep Learning Based Algorithms

In recent years, there is a wave of developing deep learning models for edge detection. For instance $N^4$-Fields [3], Deepedge [1], and Holistically-Nested Edge Detection (HED) [25]. Some of them use carefully learned features [13] and others rely on automatic feature learning, in other words, unsupervised learning. Compared to the supervised learning methods, the automatic feature learning methods have lower prediction efficiency [5].

Among these deep learning based algorithms, HED is the most popular algorithm in the application field. HED is a supervised learning model combination of a single-stream network with multiple side outputs. The input of the training set is structured as $(X_n, Y_n), n = 1, ..., N$. Here, $X_n$ are the original images and $Y_n$ are the manually annotated ground truth binary edge maps of the corresponding images. The HED model aims to learn the features that are helpful for predicting the edge maps. Instead of supervised learning on one output image, HED uses the deep supervision method, which makes learning the features more transparent. As shown in Figure 2.2, the model produces side outputs for each layer and performs supervised learning on them all for feature extraction for edge detection. Each side-output layer is associate with a classifier based on a loss function:

$$l_{side}(\boldsymbol{W}, \boldsymbol{w}) = \sum_{m=1}^{M} \alpha_m l_{side}^{(m)}(\boldsymbol{W}, \boldsymbol{w}^{(m)})$$

where $\boldsymbol{W}$ are parameters of all standard network layer, $\boldsymbol{w}$ are the weights of the classifier of the side-output layer, $M$ is the number of side-output layers, and $l_{side}^{(m)}$ is the image-level loss function for side output m. This method improves the prediction efficiency and gets better results. The average precision (AP) of HED is 0.833 on BSDS500 dataset, which outperforms thirteen popular edge detection methods [25]. We also compare our method with HED results.

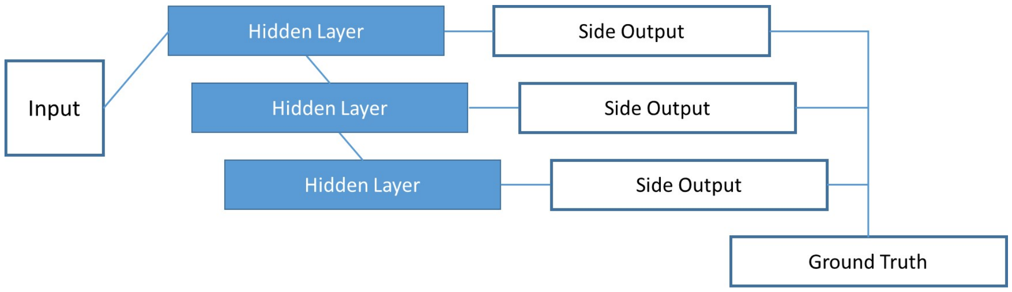

Figure 2.2: The structure of HED supervised deep learning network

# Chapter 3

# Edge Detection by Linking Points

The edges that are of interest are typically the boundaries between different bulks, such as the gap between bones in a joint, and have relative sharp contrast in brightness with respect to the bulk areas due to imaging mechanisms. Our first method is based on identifying pairs of points that are likely to be on an edge, based on the second derivative of brightness. We base the decision about whether pairs of points are likely to be on an edge on three factors: the points should be near to each other; the second derivative of brightness at the points should have a large negative eigenvalue; and the line between the points should be close to orthogonal to the eigenvector corresponding to the large negative eigenvalue. Since we only use a one-dimensional brightness value for each pixel, our method is designed for greyscale images.

## 3.1 Algorithm for Detecting Pairs of Points on Edges

Sections 3.1.1 and 3.1.2 present our method for the calculation of change of the brightness for each pixel. Then, Section 3.1.3 traverses all the pixels to find pairs of points that are most likely to be on the edge and links them.

### 3.1.1 Hessian Matrices

With the brightness of each pixel digitized to greyscale, using the derivatives of greyscale to capture the brightness change and in turn detect edges is a reasonable approach as discussed in Chapter 2. The second derivatives are expressed using the Hessian matrix $\boldsymbol{H}(x, y)$, which has entries given by the second-order partial derivatives of $b(x, y)$.

Since the brightness is only observed at discrete points and is subject to measurement error, we develop a robust method to estimate the second-order partial derivatives. The brightness as a function of $(x, y)$ can be approximated near a point $(x, y)$ by its second order Taylor expansion, which can be written as a binary quadratic function:

$$b(x+\delta, y+\epsilon) \approx \alpha(x, y)\delta^2 + \beta(x, y)\delta\epsilon + \gamma(x, y)\epsilon^2 + d(x, y)\delta + e(x, y)\epsilon + f(x, y) \quad (3.1)$$

The parameters $\alpha(x, y), \beta(x, y)$, and $\gamma(x, y)$ are the elements of the Hessian matrix $\boldsymbol{H}$(x,y).

$$\boldsymbol{H}(x, y) = \begin{bmatrix} 2\alpha(x, y) & \beta(x, y) \\ \beta(x, y) & 2\gamma(x, y) \end{bmatrix} \quad (3.2)$$

We want to estimate $\alpha(x, y), \beta(x, y)$ and $\gamma(x, y)$ from the observed values of $b(x + \delta, y + \epsilon)$. However, since the observed $b(x + \delta, y + \epsilon)$ values are subject to noise, we base our estimates on regression. Equation (3.1) is only a good approximation for small values of $\delta$ and $\epsilon$, so we perform the regression on a $k \times k$ grid centred at (0, 0). To illustrate the calculation clearly, we use the following equation instead.

$$\begin{aligned}\boldsymbol{B}(x,y) &= \alpha(x,y)(\tilde{\boldsymbol{X^2}}) + \beta(x,y)(\tilde{\boldsymbol{XY}}) + \gamma(x,y)(\tilde{\boldsymbol{Y^2}}) \\ &\quad + d(x,y)\tilde{\boldsymbol{X}} + e(x,y)\tilde{\boldsymbol{Y}} + f(x,y)\end{aligned}$$

where $\boldsymbol{B}(x,y)$ is a vector of brightness for a local area where pixel $(x,y)$ is the centre of a $k \times k$ grid with k an odd number. We use $\boldsymbol{X}$ and $\boldsymbol{Y}$ to denote the matrices of the horizontal and vertical coordinates of the $k \times k$ pixels. Without loss of generality, the matrix of $X$ and $Y$ coordinates is centralized. For instance, based on a five by five pixel block,

$$\boldsymbol{X} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix},$$

and

$$\boldsymbol{Y} = \begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}.$$

$(\tilde{\boldsymbol{X^2}})$ is used to represent a vector generated from $\boldsymbol{X}$ coordinate matrix with each element squared. For example, using the five by five $\boldsymbol{X}$ above, $(\tilde{\boldsymbol{X^2}})$ is a $25 \times 1$ vector $[4, ..., 4, 1, ..., 1, 0, ..., 0, 1, ...1, 4, ..., 4]^T$. Similarly,

$$(\tilde{\boldsymbol{Y^2}}) = [4, 1, 0, 1, 4, ..., 4, 1, 0, 1, 4]^T,$$

$$\tilde{\boldsymbol{X}} = [-2, ..., -2, -1, ..., -1, 0, ..., 0, 1, ...1, 2, ..., 2]^T,$$

$$\tilde{\boldsymbol{Y}} = [-2, -1, 0, 1, 2, ..., -2, -1, 0, 1, 2]^T,$$

and $(\tilde{\boldsymbol{XY}})$ is the multiplication of $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$ element wise. All of the vectors are of dimension $25 \times 1$.

In essence, the problem here is a linear regression problem. To simplify the formula, we use $\boldsymbol{Z}$ as a matrix of $[(\tilde{\boldsymbol{X^2}}), (\tilde{\boldsymbol{XY}}), (\tilde{\boldsymbol{Y^2}}), \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}, \boldsymbol{1}]$. Then, the parameters $\boldsymbol{\theta}$, which is the vector of parameters $\{\alpha, \beta, \gamma, d, e, f\}$, can be calculated using the following formula.

$$\boldsymbol{\theta} = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{B}$$

where $\boldsymbol{B}(x, y) = b(x + \tilde{\boldsymbol{X}}, y + \tilde{\boldsymbol{Y}})$.

For each pixel $(x, y)$, for a chosen odd number $k$, the Hessian $\boldsymbol{H}(x, y)$ is calculated using the following kernel:

$$kernel = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T$$

This kernel only changes with kernel size $k$ - the dimension of the original coordinate matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. For a fixed kernel size, the kernel is applied to every pixel.

### 3.1.2 Eigenvectors and Eigenvalues of Hessian Matrices

For a quadratic surface, the eigenvector which corresponds to the largest absolute eigenvalue points to the direction that the brightness function has the sharpest change in its value. The other eigenvector is orthogonal to the direction of sharpest change of the brightness.

Figure 3.1 shows an example of a pair of eigenvectors and eigenvalues of a Hessian matrix of one point on the edge. Eigenvector 1 follows the edge and with a small eigenvalue. Eigenvector 2 is orthogonal to the edge with a large eigenvalue. The visualization of the eigenvectors and eigenvalues of all pixels is shown in Figure 3.7.

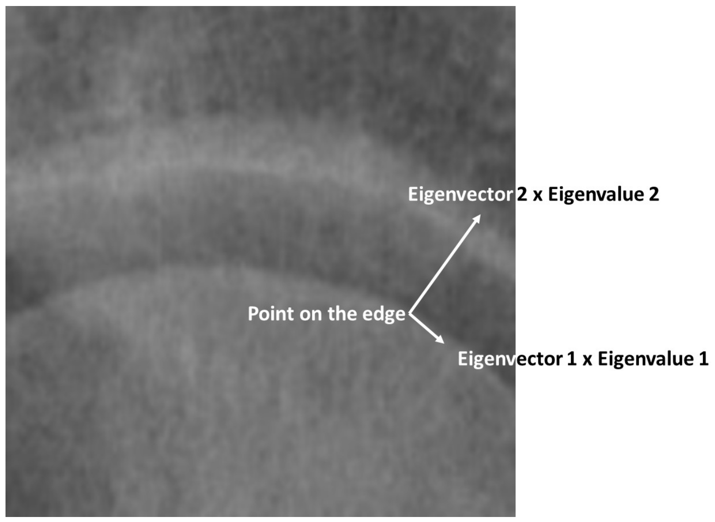

Figure 3.1: An example of eigenvectors weighted by eigenvalues of the Hessian matrix at a single pixel on the edge. The two eigenvectors are orthogonal and one of them is on the direction of the edge's path.

The points on the edges typically have a feature of strong contrast compared to other points that are not on the edges. Thus, the absolute value of one of the eigenvalues of the points on the edge is expected to be higher than other points. As illustrated in Figure 2.1, there are two sides of the edges - dark side and bright side. The point with large positive second-order derivative is the point on the dark side of the edge. Similarly, in our two dimensional image, the point with large positive eigenvalue of Hessian matrix is the point on the dark side of the edge, and the point with large negative eigenvalue of Hessian matrix is the point on the bright side of the edge, as shown in Figure 3.2.

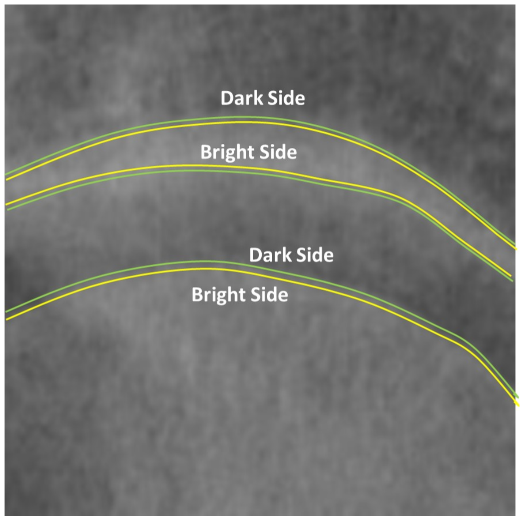

Figure 3.2: Illustration of the dark sides and the bright sides of the edges. The points on the green curves are the points on the dark sides, and the points on the yellow curves are the points on the bright sides. At the green points, the Hessian matrix has a large positive eigenvector orthogonal to the edge, while at yellow points, the Hessian matrix has a large negative eigenvalue orthogonal to the edge.

The points on the dark side of the edge will have the first eigenvectors $\boldsymbol{v}_1$ pointing to the bright side and with positive eigenvalues $\lambda_1$. And the points on the bright side will be opposite to the points on the dark side. The Hessian matrices have large (in absolute value) negative eigenvalues $\lambda_2$ associated to the direction pointing to the bright side of $\boldsymbol{v_2}$. Since we only want one edge, we choose the points on the edge on the bright side. We will therefore look for points with large negative eigenvalues.

### 3.1.3 Function for Linking The Points

Since the edge is a continuous line, identifying points where sharp brightness contrast exists is the first step. The following step is to find the pairs of points on the edge and connect them correctly to carve out the continuous curve. In this step, three factors are included in the algorithm to detect and connect points: (1) the change of brightness along the eigenvector; (2) the angle between the eigenvector and the line between the two points (near orthogonality implies greater likelihood of this pair being on the edge); (3) The distance between the two points (shorter distance implies greater likelihood of this pair being on the edge). The idea here is that we only want to join close points, and only if both have an eigenvector where brightness rapidly changes, and if the line between those points is perpendicular to these eigenvectors.

As described above, a function that incorporates the above-mentioned three factors is used to assign a goodness score to each pair of pixels to evaluate the chance of each pair being part of the edges. With the goodness score, most of the noise can be filtered out.

In the context of greyscale image processing, a positive eigenvalue represents the dark side of the edge and a negative eigenvalue represents the bright side of the edge. The pair of points of interest on the edge are on the same side of the edge, and thus their corresponding eigenvalues should have the same signs, either positive or negative. With this property, the product of eigenvalues is introduced as one term in the goodness score calculation. A segment between two points needs to satisfy three conditions to be on the edge:

(1) The eigenvalues of two nearby points on the edge have large absolute values with the same sign, we evaluate

$$contrast = \lambda_{2,\boldsymbol{p_1}}\lambda_{2,\boldsymbol{p_2}}$$

where $\boldsymbol{p_1, p_2}$ are the coordinates of the pair of points and $\lambda_{2,\cdot}$ is the second eigenvalue of the Hessian matrix of point $\cdot$.

(2) The line segment between the two points is almost orthogonal to the corresponding eigenvectors of the Hessian matrices, so we want to minimize the cosine of the angle:

$$\cos(angle) = [|(\boldsymbol{p_2} - \boldsymbol{p_1})\boldsymbol{v_{2,p_1}}| + |(\boldsymbol{p_1} - \boldsymbol{p_2})\boldsymbol{v_{2,p_2}}|]||\boldsymbol{p_2} - \boldsymbol{p_1}||^{-\frac{1}{2}}$$

where $\boldsymbol{v_{2,\cdot}}$ is the second eigenvector of the Hessian matrix of point $\cdot$.

(3) The points are near each other, so we want to minimize:

$$distance = ||\boldsymbol{p_2} - \boldsymbol{p_1}||$$

So, the score of each pair of points which measures the likelihood of whether it's on the edge or not is defined by

$$S = \lambda_{2,p_1}\lambda_{2,p_2} - w_1[|(\boldsymbol{p_2} - \boldsymbol{p_1})\boldsymbol{v_{2,p_1}}| + |(\boldsymbol{p_1} - \boldsymbol{p_2})\boldsymbol{v_{2,p_2}}|]||\boldsymbol{p_2} - \boldsymbol{p_1}||^{-\frac{1}{2}} - w_2||\boldsymbol{p_2} - \boldsymbol{p_1}||$$

where $w_1, w_2$ are the tuning parameters used to control the weights of the angle and the distance part. Based on some empirical experiments, we found that $w_1 = 2$ and

$w_2 = 0.2$ gave good results for the images studied. This makes the three parts of the formula similar in magnitude, so represents a balance between the terms. Then, we link the pairs of points that can reach a high score.

## 3.2 Edge Detection for X-Ray Images and Other Pictures

In this section we apply the above method to a number of X-ray images, and to a photograph, and we compare the results with state-of-the-art methods.

### 3.2.1 X-Ray Image Data

X-Ray images are widely used for diseases such as bone fractures, tumors, abnormal masses, calcifications, dental problems, and osteoarthritis. In almost all cases, early diagnosis and timely treatment are important and have a large impact on patients' life quality.

However, diagnosing these conditions from X-Ray images is often challenging and subjective. It would be useful to develop machine-learning techniques that would take in all the data and learn the features needed for diagnosis. This is extremely challenging because of low image quality.

Often the most crucial information in the image is the bone shape; allowing relevant measurements to be taken from it. Therefore, the first stage of this project is to identify the edges of the bones in the image. There are already methods for detecting edges in images: for example, the widely used Canny method. However, these methods are designed for use on photographs, which tend to show better image quality and clearer edges.

In this section, the method from Section 3.1.3 is applied to hip joint X-ray images shown in Figure 3.3. Extracting clear edges accurately from the X-ray images could improve the measurement of the joint space size, and further brings about an improved classification of the pre-operative X-Ray data.

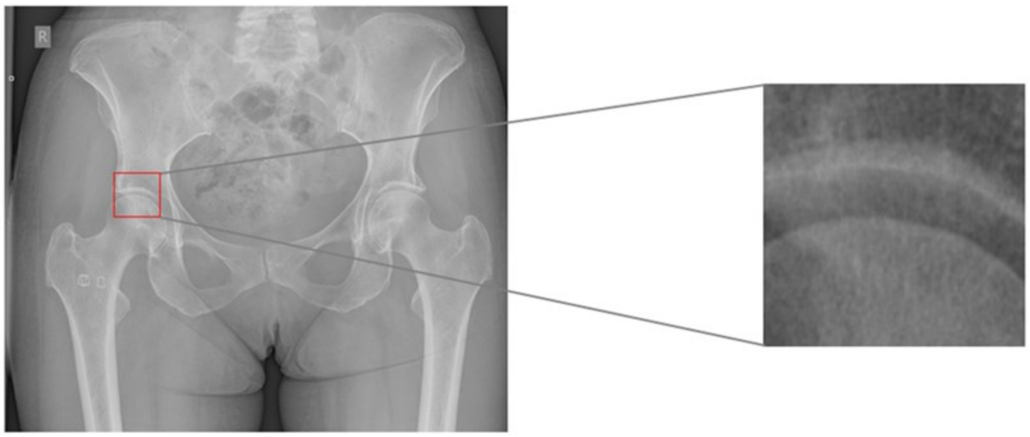

Figure 3.3: Illustration of the extracted part of the X-Ray image. The part extracted are the hip bone joints of the patients. Here we focus on the side with the risk of osteoarthritis.

The default X-ray images we use are formatted as RGB color images although the images shown are black and white. Prior to edge detection, a pre-processing step is needed to convert the original RGB color images to greyscale images. We use the default method from the OpenCV python library: [15]:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

To show how the eigenvalues and eigenvectors reflect the curve of the edges clearly, we visualize the data using heatmaps and vector plots in Figures 3.4 – 3.6.

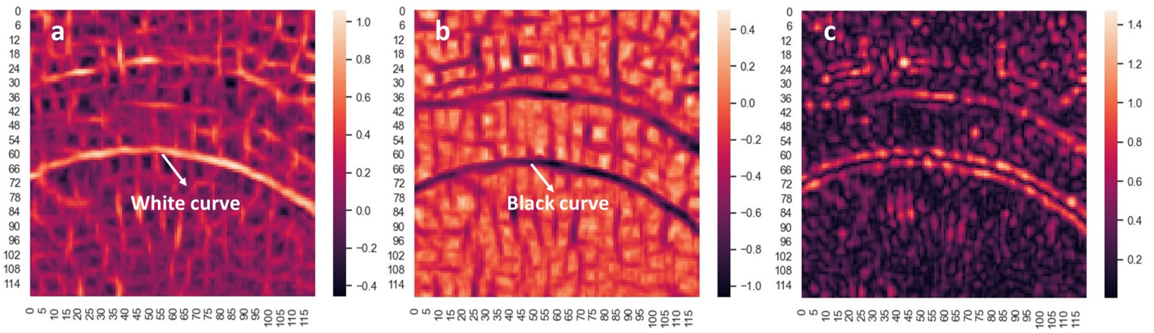

Figure 3.4: Heatmaps for (a) the first eigenvalue $\lambda_1$; (b)the second eigenvalue $\lambda_2$; and (c)the difference between the first and second eigenvalues $\lambda_1 - \lambda_2$. Larger values correspond to brighter pixels.

Figure 3.4 shows a set of heatmaps that help to visualize the distribution of eigenvalues. The edges in the original X-ray images are the boundaries between bright areas and dark areas, as discussed in the previous section. The use of these eigenvalues captures the edge by detecting steep grey scale changes. Large positive eigenvalues represent the dark side of the edge, shown as a bright pixel on the heatmap, while large (in absolute value) negative eigenvalues represent the bright side of the edge, shown as a dark pixel on the heat map.

Figure 3.4(a) is the heatmap of the first eigenvalue $\lambda_1$, where eigenvalues are ordered in decreasing order, from large positive eigenvalues to large negative eigenvalues. The white curve marks a clear contour of the dark side of one edge with relatively large positive eigenvalues. Figure 3.4(b), the heatmap of the second eigenvalue $\lambda_2$, shows clear and continuous contours of the edges with large (far from 0) negative eigenvalues. Consistent with the discussion above, each detected edge shown on the heatmap is made up with a black curve and a white curve representing the bright side and the dark side of the edge, respectively. Figure 3.4(c) is a heatmap of the difference between the first and the second eigenvalues $\lambda_1 - \lambda_2$. The edges are intermittent because of the noise in the X-ray image. The bright side of the edge shows a clearer pattern than the dark side because the noise is additive for X-ray images. The noise is generated because something, other than bones, stops the X-ray occasionally. So, the noise area is brighter than it should be and it influences the dark side of the edge but not the bright side. Since the size of the joint space of the hip bones is an important measurement to diagnose osteoarthritis, we consider the second eigenvalue $\lambda_2$, which shows clear patterns of the bright side of edges, as an appropriate statistic for edge detection of the X-ray images.

As discussed in Section 2.2.1, some edge detection methods are based on the first derivative. So, we also draw a heatmap for the gradient using first derivatives as a comparison. The calculation of the gradient is the same as the Canny method. As shown in Figure 3.5, the bright and dark curves at the edge areas are thicker than the curves for second derivatives shown in Figure 3.4. Some parts of the curves are darker than other parts. If we use the first derivative and apply a threshold on the gradient as Canny does, it might lose the darker parts on the curves.

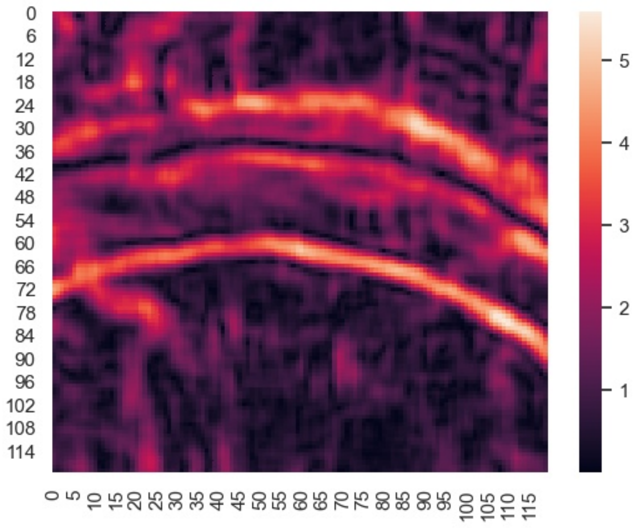

Figure 3.5: Heatmap for the gradient using first derivative of brightness. Larger values correspond to brighter pixels.

On the other hand, eigenvectors show the directions of the change of brightness. It is another metric that can be used to detect the edges. We visualize the eigenvectors in Figure 3.6.

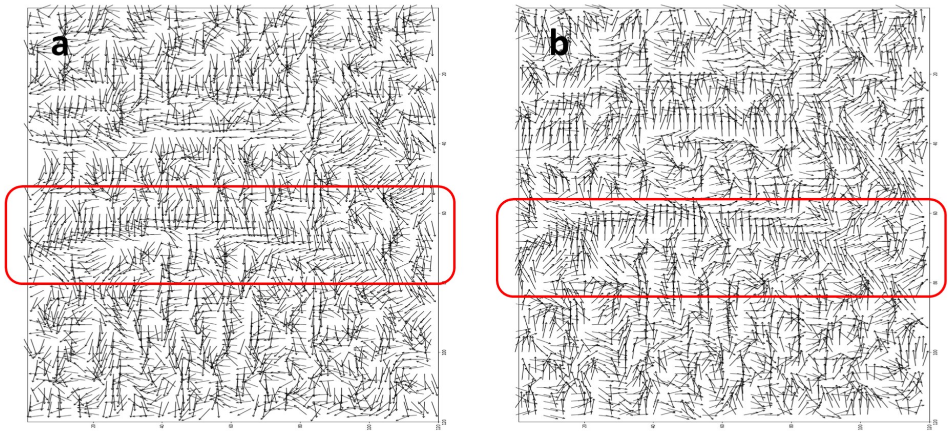

Figure 3.6: Arrow plots for (a) the first eigenvectors $\boldsymbol{v_1}$ and (b) the second eigenvectors $\boldsymbol{v_2}$. To observe them clearly, all of the eigenvectors are five times the original unit length.

From the vector plots in Figure 3.6, it can be observed that the vectors falling on the edges follow a more aligned direction, while other vectors are more randomly oriented. Figure 3.6(a) is the arrow plot of the first eigenvectors $\boldsymbol{v_1}$. As shown in the red inset of Figure 3.6(a), there are consecutive eigenvectors, that are almost vertical, follow a clear path along the edge. These features together indicate an edge. Similarly, in Figure 3.6(b), which is the arrow plot of the second eigenvectors $\boldsymbol{v_2}$, the eigenvectors along the edge are also almost vertical. These are also consistent with the theory discussed in Section 3.1.2.

From the discussion above, eigenvalues and eigenvectors are both important for edge detection. By integrating them together, we gain a visualization on both eigenvectors and eigenvalues. By drawing arrow plots of weighted eigenvetors by the corresponding eigenvalues, a synergy of both eigenvectors and eigenvalues can be observed in Figure 3.7.

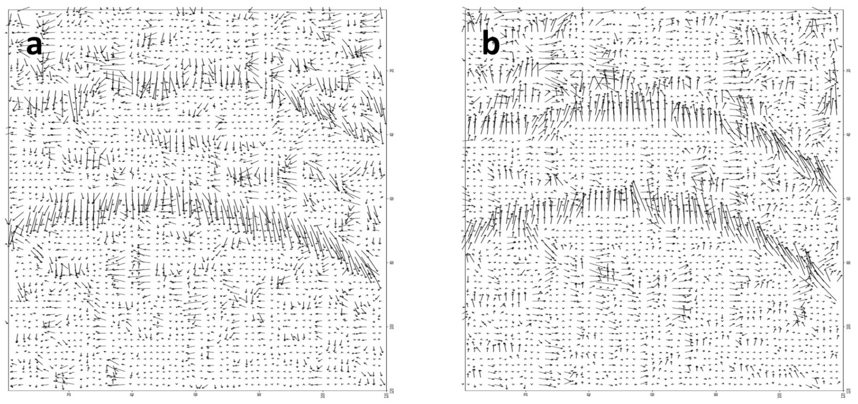

Figure 3.7: Plots of the eigenvectors weighted by the corresponding eigenvalues. Part (a) is for the first eigenvalue $\lambda_1$ and eigenvectors $\boldsymbol{v_1}$; (b) is for the second eigenvalue $\lambda_2$ and eigenvectors $\boldsymbol{v_2}$.

Figure 3.7 shows the arrow plots of weighted eigenvectors. Figure 3.7(a) is the vector plot of the first eigenvectors weighted by the absolute value of the corresponding eigenvalues. There are two curves with clear paths and they are the dark sides of the edges of the original image. Similarly, Figure 3.7(b) is the vector plot of the second eigenvectors weighted by the absolute value of the corresponding eigenvalues. The curves in Figure 3.7(b) are the boundaries of the bright sides of the edges in the original image.

Same as the heatmap, we also draw the vector plot for the first derivatives as a comparison. As shown in Figure 3.8, we plot the vectors using the first derivatives on x and y directions $\left[\frac{db}{dx}, \frac{db}{dy}\right]$. The long vector means a strong contrast around that pixel. Though the long vertical vectors form clear patterns of the edges, there are some vectors pointing to random directions around the edge area. And some long vectors are not on the edges. So, using these vectors to detect the edge may keep more noise than using eigenvectors of Hessian matrices.

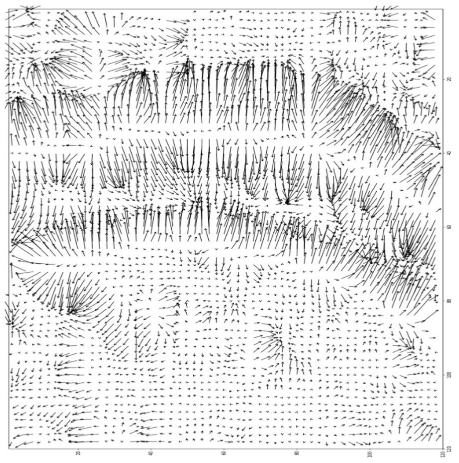

Figure 3.8: Plots of the vectors formed by first derivatives on x and y directions.

Since the eigenvectors on the edges are on the directions of sharp birghtness changes, that are orthogonal to the direction of the edge, we draw arrow plots using the orthogonal directions $\boldsymbol{v}._{orthogonal} = \boldsymbol{Jv}.$ of eigenvectors to further illustrate the edges, where $J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is used to rotate a vector to its orthogonal direction.

Figure 3.9(a) is the vector plot of the orthogonal vectors of the first eigenvectors weighted by the absolute value of the first eigenvalues. The vectors are following clear paths of the dark sides of the edges of the original image. Similarly, Figure 3.9(b) is the vector plot of the orthogonal vectors of the second eigenvectors weighted by the absolute value of the second eigenvalues. The curves in Figure 3.9(b) show the boundaries of the bright sides of the edges.

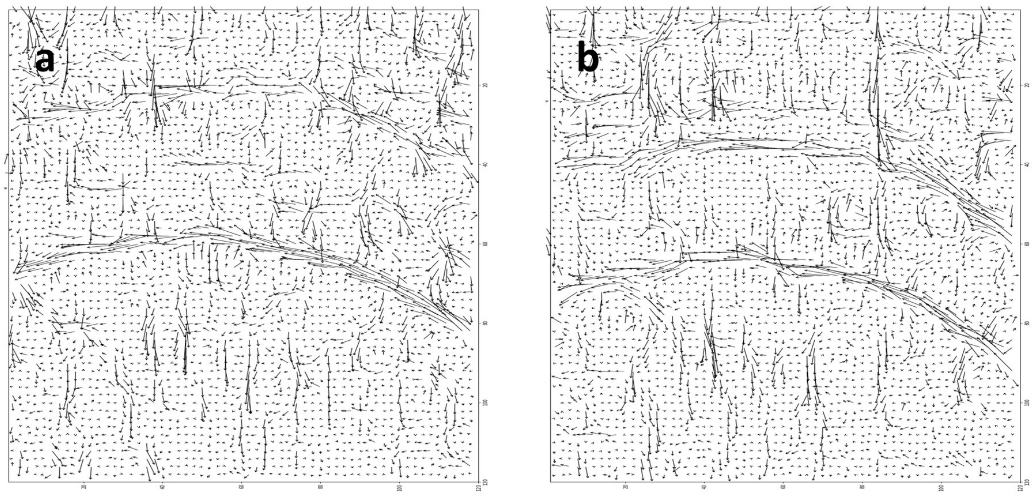

Figure 3.9: Plots of the orthogonal vectors of eigenvectors weighted by the corresponding eigenvalues. Part (a) is for the first eigenvalues $\lambda_1$ and eigenvectors $\boldsymbol{v_1}$; (b) is for the second eigenvalues $\lambda_2$ and eigenvectors $\boldsymbol{v_2}$.

Compared with the first eigenvalue/eigenvector, the second eigenvalue/eigenvetor is more capable of capturing the joint space. Thus, we use the second eigenvalues to calculate the goodness scores in this chapter and use both the second eigenvectors and eigenvalues to trace the path of the edge in Chapter 4.

### 3.2.2 Results of Edge Detection for X-Ray Images

The X-ray images used in this study are 128 pixels $\times$ 128 pixels in size. The kernel sizes of $5 \times 5$, $7 \times 7$, $9 \times 9$, and $11 \times 11$ are reasonable sizes. Figure 3.10 shows a comparison of the results using these kernel sizes.

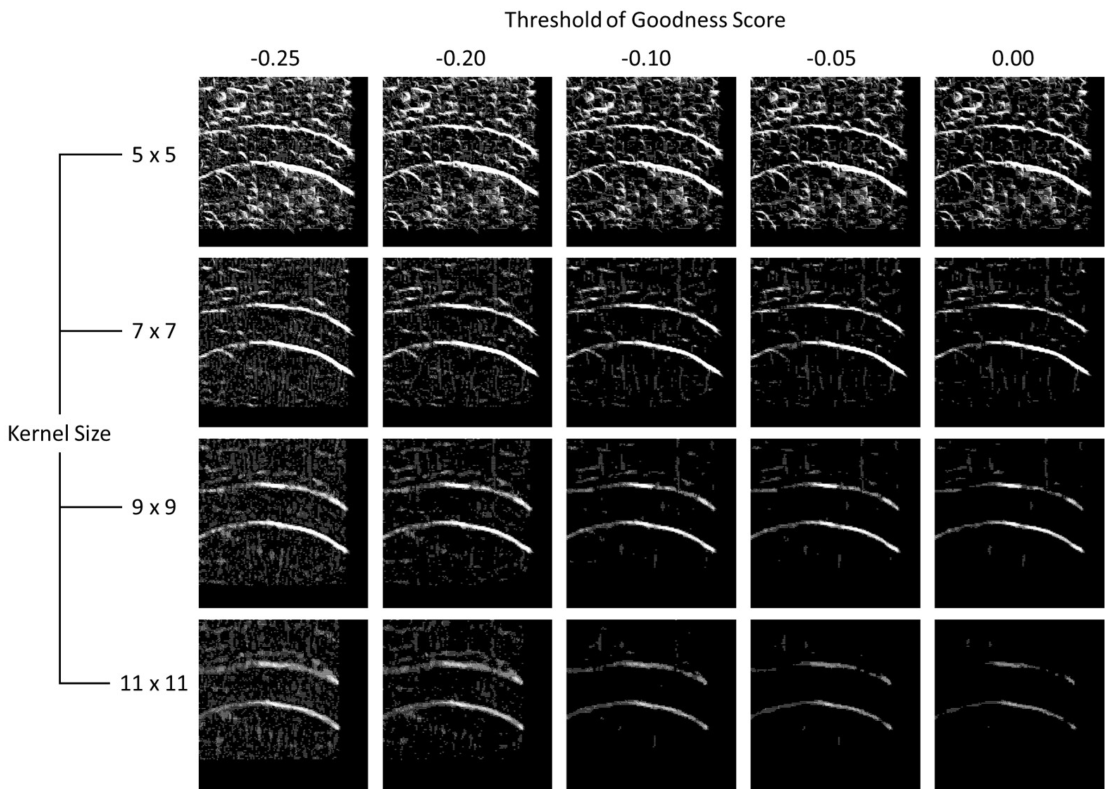

Figure 3.10: Edge detection results with kernel sizes of $5 \times 5, 7 \times 7$, $9 \times 9$, and $11 \times 11$ from the top to the bottom and de-noised by the numbers above each column.

This figure is a set of results with kernel sizes of $5 \times 5$, $7 \times 7$, $9 \times 9$, and $11 \times 11$. Each column uses the same threshold of the goodness score $S$ by assigning value 1 to the points with S larger than the threshold and 0 otherwise. All of them detect clear edges as expected .

As shown in Figure 3.10, smaller kernel size keeps more pairs of points that are considered to be on an edge, so it connects the points better and forms more continuous edges. Take the first column of Figure 3.10 as an example, the larger the kernel size, the less likely two points are detected as a pair of points that form the edge. So, the edges detected under kernel size of $11 \times 11$ have less noise compared to kernel size $5 \times 5$. Larger kernel size helps remove the noise well, but larger kernel size also causes the discreteness of the edges, which affects the precise measurement of joint space. There is a trade-off between edge continuity and the noise level. For instance, the above figure shows that the kernel size of $9 \times 9$ plots the edges more continuously, and maintains the clearness of the edges for the 128 by 128 pixels X-ray image. Kernel size $11 \times 11$ removes too many points and makes the edges discrete.

The threshold of goodness score is another important metric of edge detection. Lower threshold keeps more pairs of points and noise. Higher threshold filters more points forming a less continuous edge. For example, for kernel size $9 \times 9$ in Figure 3.10, the results using threshold -0.25 and -0.20 have more noise than using -0.10. Threshold value is another trade-off between edge continuity and noise level.

For this set of 128 by 128 pixels X-ray images, the results with kernel size $9 \times 9$ and threshold value -0.10 outperform others, and is considered to be a sweet spot. So, we compare our method under this setting with two popular methods, Canny and HED, in Figure 3.11.

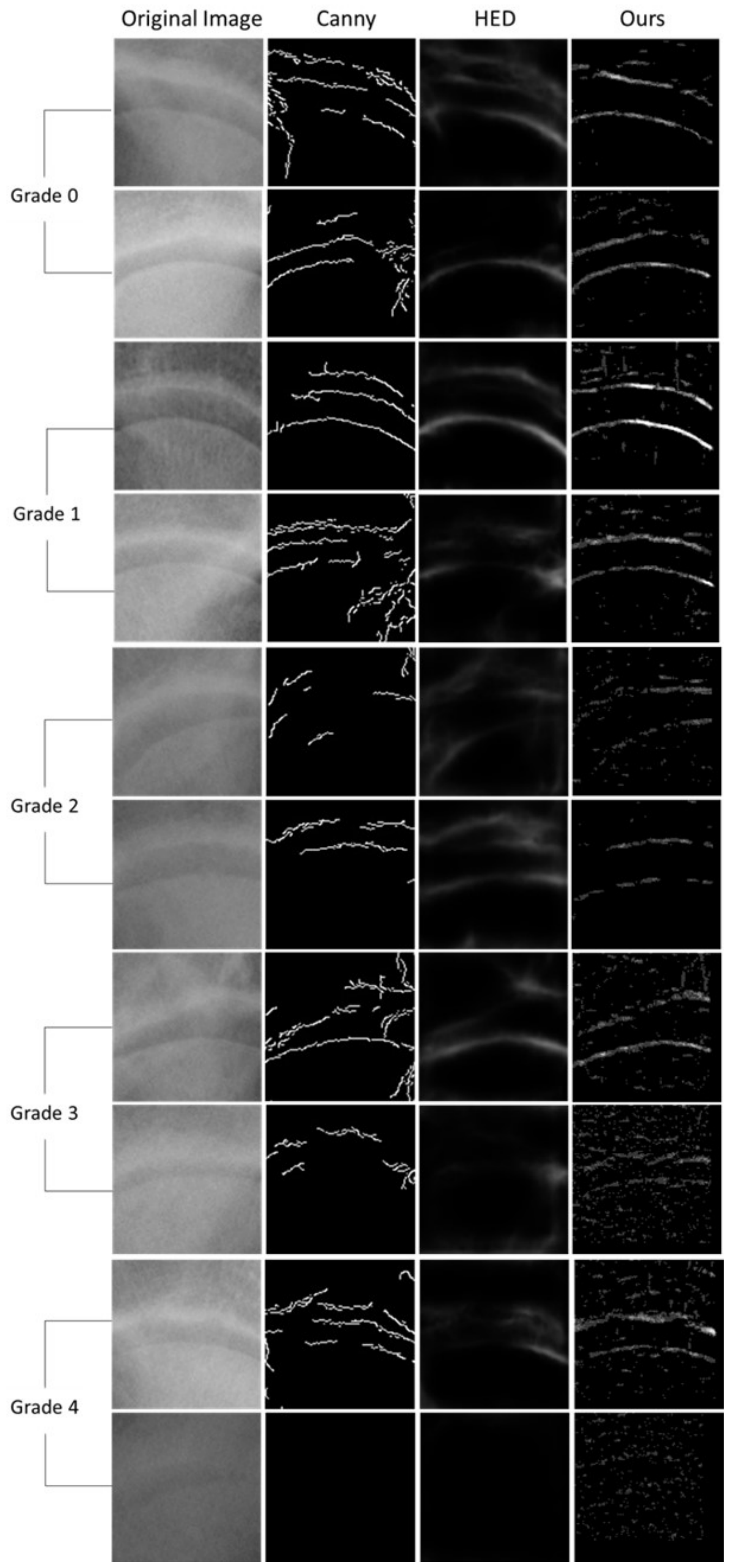

Figure 3.11: A comparison between the method of this chapter and two current state-of-the-art methods: Canny and HED. The first column shows the original images, and the middle two columns are results using Canny and HED respectively. The last column shows the edge detection results using our method.

We test our method on different grades of osteoarthritis from Grade 0 to Grade 4. The first column of Figure 3.11 shows the original X-ray images. The corresponding results using Canny [1] and HED[2] are shown in the second and third columns, respectively. The results obtained with our method discussed in this chapter are shown in the last column. The method developed by this work removes more noise and keeps more complete edges than the Canny. When compared with HED, this new method keeps more information about the edges and is less time-consuming. It doesn't need a pre-trained deep learning model.

[1]The Canny method is based on Canny Edge Detection Tutorial by Bill Green, 2002. The source code is from the Python package - OpenCV $https : //opencv - python - tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html$.

[2]The source code of HED is from $https : //github.com/tensorpack/tensorpack/tree/master/examples/HED$.

### 3.2.3 Edge Detection for Other Images

#### For Whole X-Ray Images

Though doctors need to focus on only one part of the X-ray image to diagnose the diseases, it is worthwhile to try this method on the whole X-ray images. Figure 3.12 is a comparison of the results on a complete X-ray image. The top left of Figure 3.12 is the original image, top right is the result using Canny edge detector, bottom left is the result under HED, and bottom right is the result of our method under the same setting - kernel size $9 \times 9$ and threshold number -0.10. All of these three methods show most of the edges clearly. Canny keeps more information but the edges are not connected well. HED removes the noise perfectly and draws the shape of the big bones clearly. But HED also removes some important parts of the bones, such as the relatively small joint part. HED generates several side outputs of the hidden layers and fuses them. Some of the side output misses the small joint parts, and that influences the fused output. In contrast, our method keeps the edges of the small part completely and most of the edges are continuous.

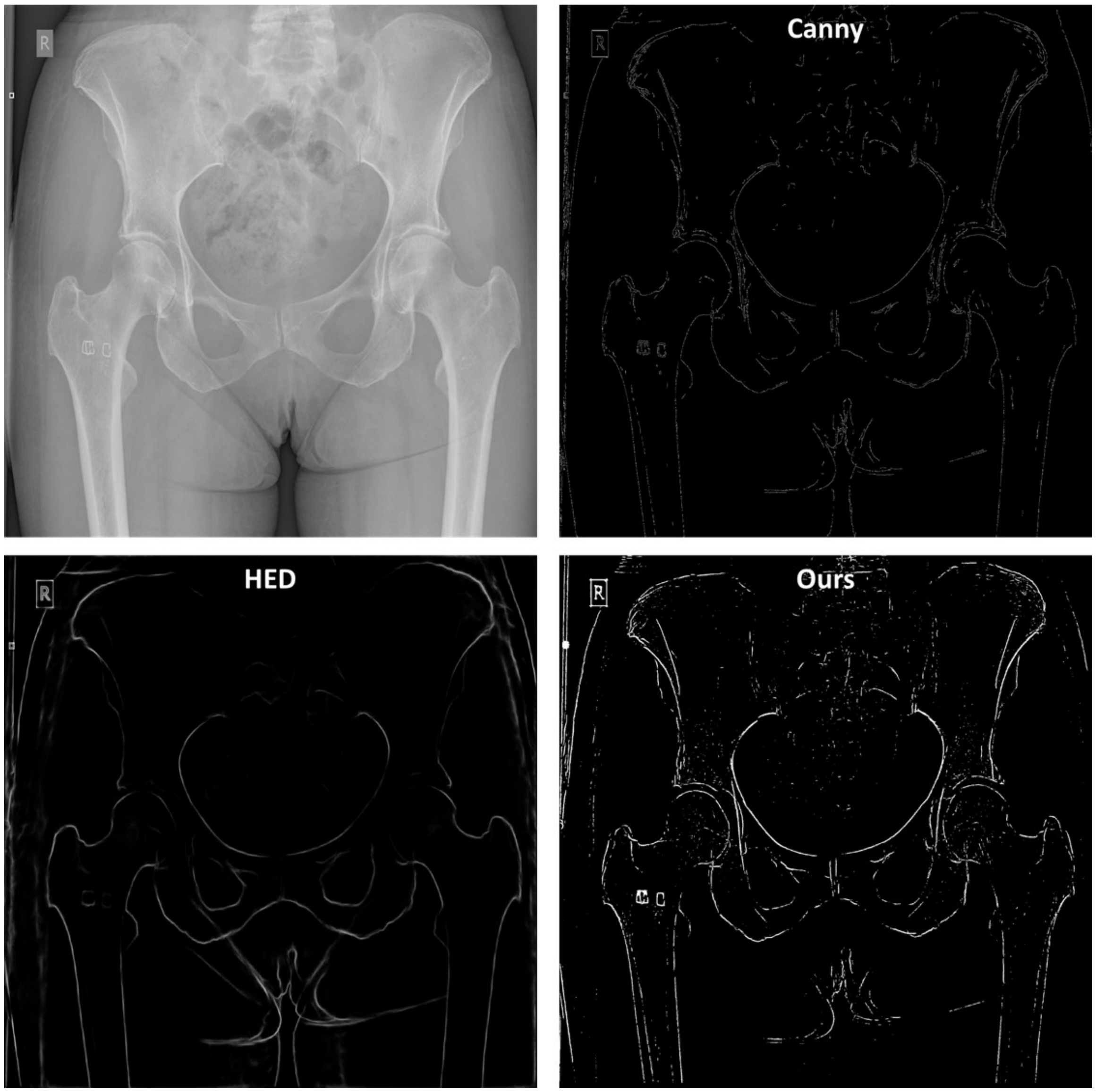

Figure 3.12: Illustration of an application of our method on a whole X-ray image. Top-left is the original X-ray image, and top-right is the result using Canny edge detector. Bottom-left is the result under HED, and bottom-right is our result.

### For RGB Pictures

In addition to the above X-ray image analysis, we also try our method on some regular colour photographs. As illustrated in Figure 3.13, an image of a leaf is used as an example for edge detection in regular colour photographs. The leaf veins can be detected clearly under the same tuning parameters as used for the X-ray images, as shown in part (b). In the case that main veins are of more interest, simply by changing the threshold numbers we get the result as shown in part (c) which mostly identifies just the main veins of the leaf. And this result shows that our method works well when two edges cross.

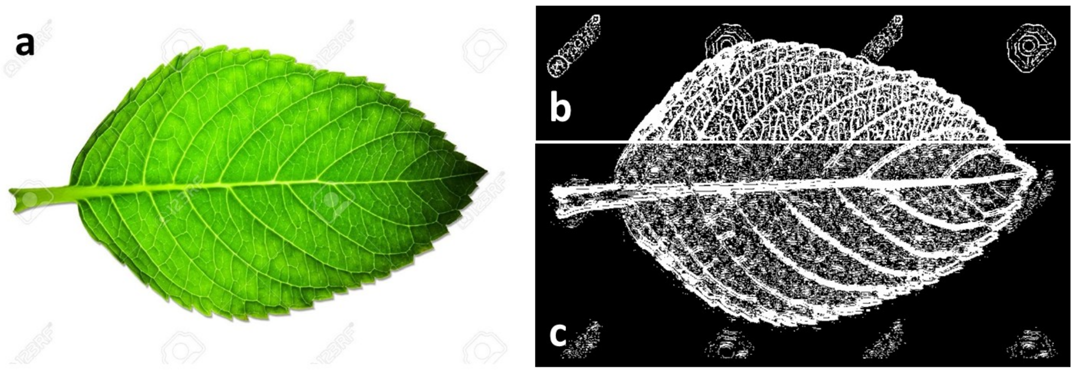

Figure 3.13: An application of our method on a photograph of a leaf. (b) Result using a relatively lower threshold number. (c) Result using a higher threshold number.

It is quite straight-forward to further develop our method to three channel RGB images by either adding the goodness scores from three channels or detecting edges using each channel and then overlaying the pictures afterwards. Such development will be left as possible future work.

# Chapter 4

# Edge Detection by Tracing Paths

As shown in the previous chapter, graphical data, especially medical X-ray images, contains a large amount of intrinsic noise. From the work shown in Chapter 3, it is clear that the quality of edge detection was greatly compromised by noise. By detecting points that are of interest and linking them to generate edges, most of the noise could be excluded, however the detected edges consist of intermittent segments, between points with integer coordinates, rather than smooth and consecutive lines. Figure 4.1 is a zoomed-in plot from the method in Chapter 3.

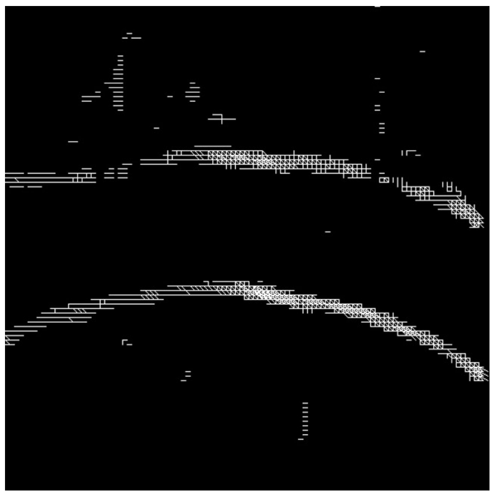

Figure 4.1: A zoomed in plot from the method in Chapter 3. The edges are formed by lines connecting pair of points. The edges detected are not smooth and discrete at some points.

In this chapter, we want to solve the problem of discreteness and generate smooth curves of the edges under the real coordinates (non-integer coordinates) in our X-ray images. As discussed in Section 3.2.1, the orthogonal vector of the second eigenvector of the Hessian matrix follows the direction of the edge perfectly. And the corresponding eigenvalue has a relatively large absolute value. Based on these features, we develop a method that generates a continuous and smooth curve of the edge by tracing points along the orthogonal vector of the second eigenvectors progressively from a starting point.

## 4.1 A Rotation of the Hessian Matrix

Figure 3.9 visualized a set of vector plots using the orthogonal vectors of the eigenvectores and weighted by the absolute value of corresponding eigenvalues. Figure 3.9(b) shows a clear pattern on the bright side of the edges. The weighted orthogonal vector of the second eigenvector $\boldsymbol{v_{2orthogonal}} = \boldsymbol{J}\boldsymbol{v_2}$ traces the edge perfectly. So, the idea of this method is to trace the bright side of the edge along the direction of the orthogonal vector $\boldsymbol{v_{2orthogonal}}$ on the edge. Since the two eigenvectors are orthogonal, $\boldsymbol{v_{2orthogonal}}$ is $\boldsymbol{v_1}$, so, in order to generalize our method, we create a new matrix with the eigenvectors reversed - $\boldsymbol{K}$ matrix.

$$\begin{aligned}
K(x,y) &= -JH(x,y)J \\
&= -\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial^2 b}{\partial x^2} & \frac{\partial^2 b}{\partial x \partial y} \\ \frac{\partial^2 b}{\partial y \partial x} & \frac{\partial^2 b}{\partial y^2} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \\
&= -\begin{bmatrix} -\frac{\partial^2 b}{\partial y \partial x} & -\frac{\partial^2 b}{\partial y^2} \\ \frac{\partial^2 b}{\partial x^2} & \frac{\partial^2 b}{\partial x \partial y} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} -\frac{\partial^2 b}{\partial y^2} & \frac{\partial^2 b}{\partial y \partial x} \\ \frac{\partial^2 b}{\partial x \partial y} & -\frac{\partial^2 b}{\partial x^2} \end{bmatrix}
\end{aligned}$$

Suppose Hessian matrix $\boldsymbol{H}$ has spectral decomposition: $\boldsymbol{H} = \boldsymbol{v_1}\lambda_1\boldsymbol{v_1}^T + \boldsymbol{v_2}\lambda_2\boldsymbol{v_2}^T$, then

$$\begin{aligned}
K(x,y) &= -JH(x,y)J \\
&= -J\boldsymbol{v_1}\lambda_1\boldsymbol{v_1}^T J - J\boldsymbol{v_2}\lambda_2\boldsymbol{v_2}^T J \\
&= -\boldsymbol{v_{1orthogonal}}\lambda_1\boldsymbol{v_{1orthogonal}}^T - \boldsymbol{v_{2orthogonal}}\lambda_2\boldsymbol{v_{2orthogonal}}^T \\
&= -(\boldsymbol{v_2}\lambda_1\boldsymbol{v_2}^T + \boldsymbol{v_1}\lambda_2\boldsymbol{v_1}^T)
\end{aligned}$$

It is proved above that matrices K and H have the same pair of eigenvectors, but the eigenvalues are negated and switched. The eigenvector of the K matrix which is parallel to the tangent vector of the edge curve is associated with a large positive eigenvalue (thus will be termed as the first eigenvector $\boldsymbol{v_{K1}}$ from now on) and the other eigenvector $\boldsymbol{v_{K2}}$ is associated with a smaller eigenvalue $\lambda_{K2}$.

## 4.2 Tracing Edges by Local Greedy Search

As discussed in Section 4.1, the first eigenvalues and eigenvectors of $\boldsymbol{K}$ are more favorable to our edge detector. To identify the edges accurately, this edge detector takes both of the eigenvalues and eigenvectors into account. The following steps were taken progressively.

1. Define a starting point $S_0$ of the edge.

2. Set a window size $w$ and step size $n$.

3. From the starting point, perform a local greedy search through all the points within the window at left side and identify the point $(x_d, y_d)$ maximizing the first eigenvalue $\lambda_{K1}$ of $\boldsymbol{K}$.

4. Move along the direction of the corresponding first eigenvector $\boldsymbol{v_{K1}}$, which has positive inner product with the previous direction, by step size $n$ to reach the new starting point $S_0$.

5. Repeat steps 3 and 4 till it comes to the boundary of the image or back to the initial point on the edge.

6. Repeat steps 3 to 5 for the right side window of the original starting point. Here, we need to change the first direction to the opposite of the first eigenvector $-\boldsymbol{v_{K1}}$.

7. Link the points found in steps 3 to 6.

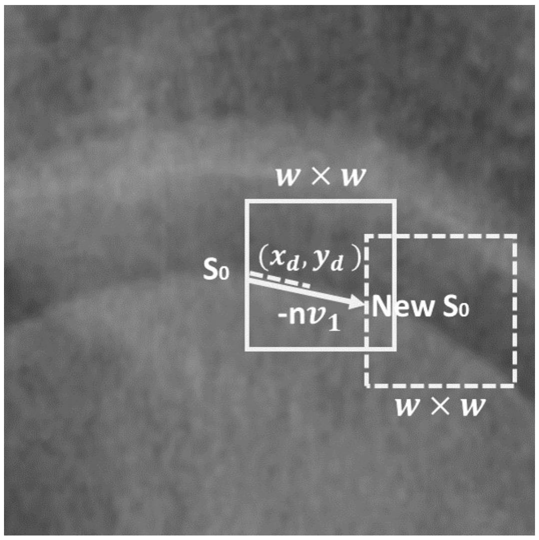

Figure 4.2: Illustration of how we trace the edge by maximizing the first eigenvalue within an window with size $w$.

As shown in Figure 4.2, this method has a window size $w$ and a step size $n$. Within a $w \times w$ window at the left (or right) side of the starting point, this method selects the point with the largest first eigenvalue $\lambda_{K1}$, then moves in the direction that has positive inner product with the previous direction, $\boldsymbol{v_{K1}}$, or $-\boldsymbol{v_{K1}}$, by step size $n$ to locate the new starting point. By iterating steps 3 to 6, the points on the edge will be detected. Then, link the points together to trace the edge out. Different from the method in Chapter 3, we now allow our edge tracing to pass through points with non-integer coordinates.

## 4.3 Identify Starting Points Automatically

In this edge detection algorithm, a critical part is setting the starting points. Instead of selecting it manually, we want to develop a method to detect the starting points automatically.

### 4.3.1 Algorithm for Starting Points Selection

First of all, we need a range of candidate starting points. As shown in Figure 4.3, we use the points on the vertical dotted line in the middle of the zoomed in X-ray image as the candidate starting points. We use the method from Section 4.2 to trace a small segment of the edge path from each candidate starting point, and assess all the edge segments based on the extent to which the direction of the edge segment at point (x,y) aligns with the largest eigenvector of K(x,y), and the magnitude of the corresponding eigenvalue. We demonstrate this method in Figure 4.3.

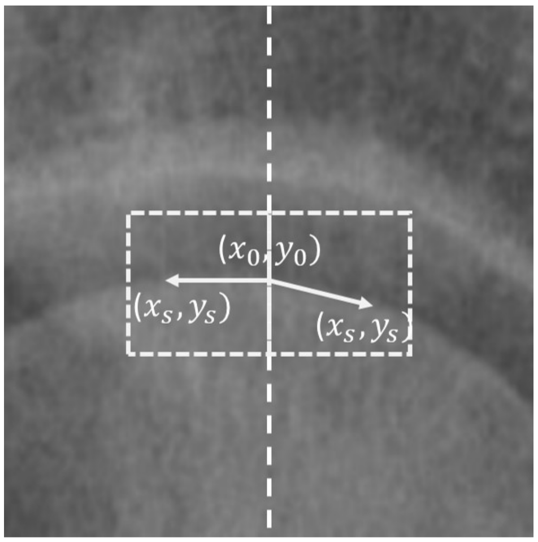

Figure 4.3: The candidate starting points of the zoomed-in hip joint part of the X-ray image. And the windows to calculate the objective function.

We use the function:

$$L(x, y, t) = v(t)^T \boldsymbol{K}(x(t), y(t))v(t)$$

where $v$ is the vector $[\Delta x, \Delta y]^T$ from the starting point $(x_0, y_0)$ to the selected point $(x_s, y_s)$ within the window and $t$ is the coordinate of the point projected on vector

$[\Delta x, \Delta y]$. So, $(x, y)$ is a function of $t$ and

$$v(t) = [\Delta x, \Delta y]^T = [x'(t), y'(t)]^T.$$

If $v(t)$ follows the edge, the contrast along $v(t)$ will be large and $L(x(t), y(t), t)$ of each point on the edge will be large. So, we maximize the objective function

$$Z = \int_{u_1}^{u_2} L(x(t), y(t), t)dt$$

over all the candidates of starting points. Here, $u_1$ and $u_2$ are the coordinates of $(x_s, y_s)$ on the vector $[\Delta x, \Delta y]$ of the left side and right side windows of the starting point. The objective function $Z(x_s, y_s)$ measures how well the curve follows an edge. We select starting points $(x_s, y_s)$ which are local maxima of Z, and which correspond to large values of Z. As there may be multiple edges in the image, we may select multiple starting points.

To calculate the integral in the objective function:

$$\begin{aligned} v^T K(x,y)v &= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} -\frac{\partial^2 b}{\partial y^2} & \frac{\partial^2 b}{\partial y \partial x} \\ \frac{\partial^2 b}{\partial x \partial y} & -\frac{\partial^2 b}{\partial x^2} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ &= \begin{bmatrix} -\Delta x \frac{\partial^2 b}{\partial y^2} + \Delta y \frac{\partial^2 b}{\partial x \partial y} & \Delta x \frac{\partial^2 b}{\partial y \partial x} - \Delta y \frac{\partial^2 b}{\partial x^2} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ &= -\Delta x^2 \frac{\partial^2 b}{\partial y^2} + 2\Delta x \Delta y \frac{\partial^2 b}{\partial y \partial x} - \Delta y^2 \frac{\partial^2 b}{\partial x^2} \end{aligned}$$

Since $\boldsymbol{H}(x,y) = \begin{bmatrix} \frac{\partial^2 b}{\partial x^2} & \frac{\partial^2 b}{\partial x \partial y} \\ \frac{\partial^2 b}{\partial y \partial x} & \frac{\partial^2 b}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2\alpha(x,y) & \beta(x,y) \\ \beta(x,y) & 2\gamma(x,y) \end{bmatrix}$, the above equation can be written as

$$v^T K(x,y)v = -2\gamma(x,y)\Delta x^2 + 2\beta(x,y)\Delta x \Delta y - 2\alpha(x,y)\Delta y^2.$$

We evaluate the integral numerically as a sum using 20 internal points (10 for each side). Because we are only comparing the integral at different points, this is sufficiently accurate for our purposes.

### 4.3.2 Interpolation of K Matrix

To calculate the integral, we need $\boldsymbol{K}$ matrices with real coordinates, but the $\boldsymbol{K}$ matrices calculated in Section 4.1 are only for the points with integer coordinates. So, we interpolate the $\boldsymbol{K}$ matrix.

Since $\boldsymbol{K}(x,y) = -J\boldsymbol{H}(x,y)J$, we can interpolate the Hessian matrix first and then left multiply $-J$ and right multiply $J$. As illustrated in Figure 4.4, we extract one grid of the image with four known Hessian matrices, $H_1, H_2, H_3$ and $H_4$, at four corners. And the Hessian matrix of point $(x,y)$ in the square is the Hessian matrix we want to calculate.

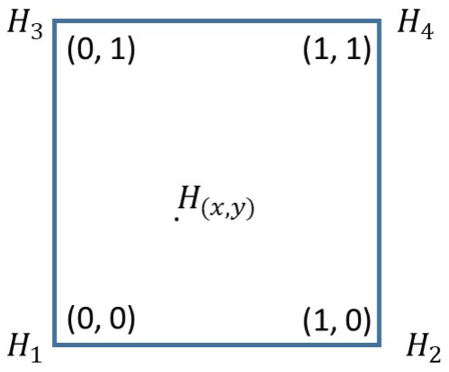

Figure 4.4: Illustration for interpolating the Hessian matrix.

We use the following function to interpolate the Hessian matrix:

$$H_{(x,y)} = (1-x)(1-y)H_1 + x(1-y)H_2 + (1-x)yH_3 + xyH_4$$

The closer the point to one of the corners, the higher the weight of the Hessian matrix of that corner. So, we use the multiplication of the distances between the point and the corner on the opposite direction on the vertical and horizontal coordinates as the weight to interpolate the Hessian matrix. For example, if the point $(x, y)$ is close to corner (0, 0), $(1-x)$ and $(1-y)$ will be relatively large, $H_1$ will have a larger weight. This way, we calculate the Hessian matrix and $\boldsymbol{K}$ matrix for every point on the image and the objective function in Section 4.3.1.

## 4.4 Tracing The Edge Curve on Hip Joint X-Ray Images

Figure 4.5 demonstrates our automatic method for identifying starting points. The left side of Figure 4.5 is an X-ray image which shows the candidate starting points. The line chart on the right side of Figure 4.5 shows the corresponding values of the objective function $Z$ for the candidate starting points. Since this method aims to find the points that maximize the objective function $Z$, Figure 4.5 shows the peaks of the line chart to illustrate the points that meet the requirement. For this specific X-ray image, there are three peaks and their corresponding pixel positions in matrix style coordinates, where the first coordinate is the row (starting at the top) and the second is the column (starting at the left), are (38, 60), (66, 60), and (87, 60) on the image, as indicated by the three horizontal lines in Figure 4.5. Two of the candidate starting points, namely (38,60) and (66,60), have sharp peaks of $Z$. These two starting points are the points we want to detect. The other peak is more broad and corresponds to a weak edge. For this research, where only the strong edges of the joint bones are needed, the broad peak will be ruled out. But in other cases, we may be interested in finding these weak edges.

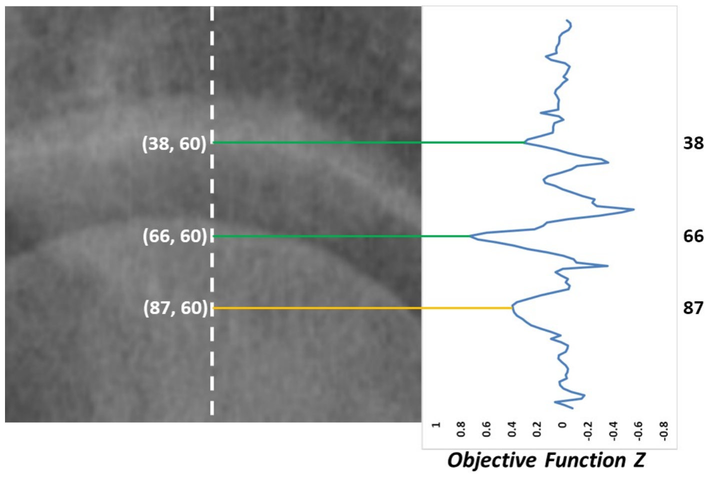

Figure 4.5: Illustration of detecting the starting points using our algorithm. The horizontal coordinate of the line chart is the value of $Z$ and the vertical coordinate is the position of the pixel. The three local peaks of $Z$ are connected to the corresponding candidate starting points. The sharp peaks (green lines) are the ones we are looking for. The broader peak (orange line) is not of interest in this image.

The results of applying our method from Section 4.2 with these starting points and window size w=1 and step sizes n=5 and n=7 are shown in Figure 4.6.

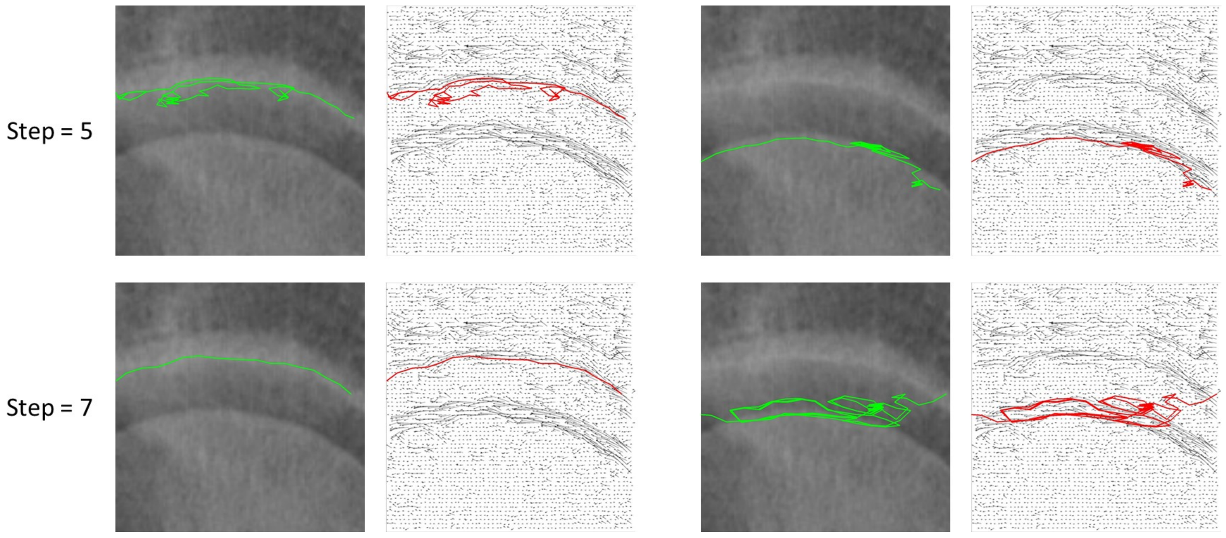

Figure 4.6: Results with window size 1 for starting points (38, 60) (left) and (66, 60) (right). Edges detected are shown as green and red curves in the superimposed images.

When step size is 5, both the upper edge and the lower edge detected form loops along the curve. Step size 7 makes the upper edge detected clear and smooth. But the lower edge with step size 7 also forms loops.

If the window size is too small, it might not include the area with sharp brightness change. To continue the tracing process, the detector will find a point not exactly on the edge and deviate using the eigenvector of that point which will point in a random direction. After this, it may be further from the edge and will move in random directions until it returns to the edge, finds another edge, or reaches the boundary of the image. To prevent this problem, we increase the window size to include enough pixels in our window. Reducing the step size also reduces the risk of leaving the edge. Figure 4.7 shows the results with step size n=1 and window size w=9 and w=11, using the same starting points, (38,60) and (66,60).

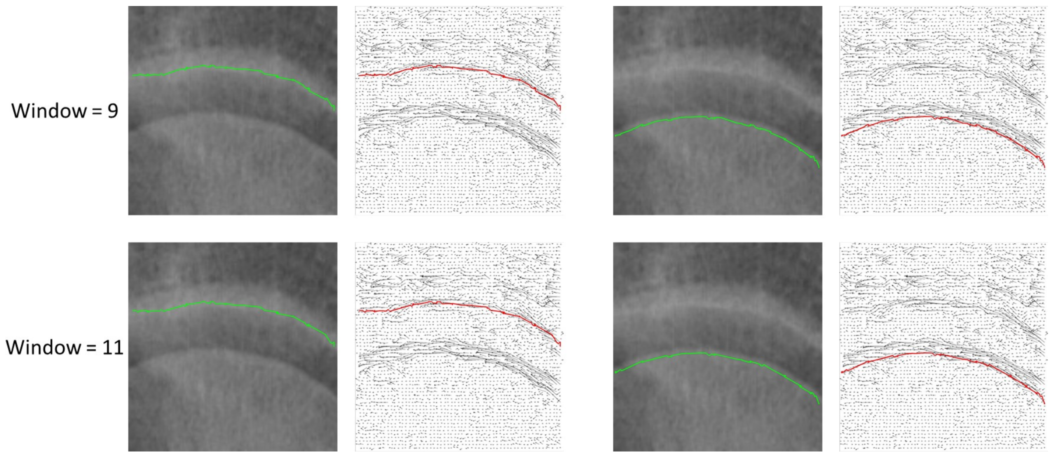

Figure 4.7: Results for step size $n = 1$ for starting points (38, 60) (left) and (66, 60) (right). Edges detected are shown as green and red curves in the superimposed images.

The edges shown in Figure 4.7 are slightly jagged, but follow the true edge much more closely. The edges detected under window size 11 are slightly smoother than window size 9. Increasing the step size makes the detected edges less jagged. Figures 4.8 and 4.9 show results using larger step sizes.

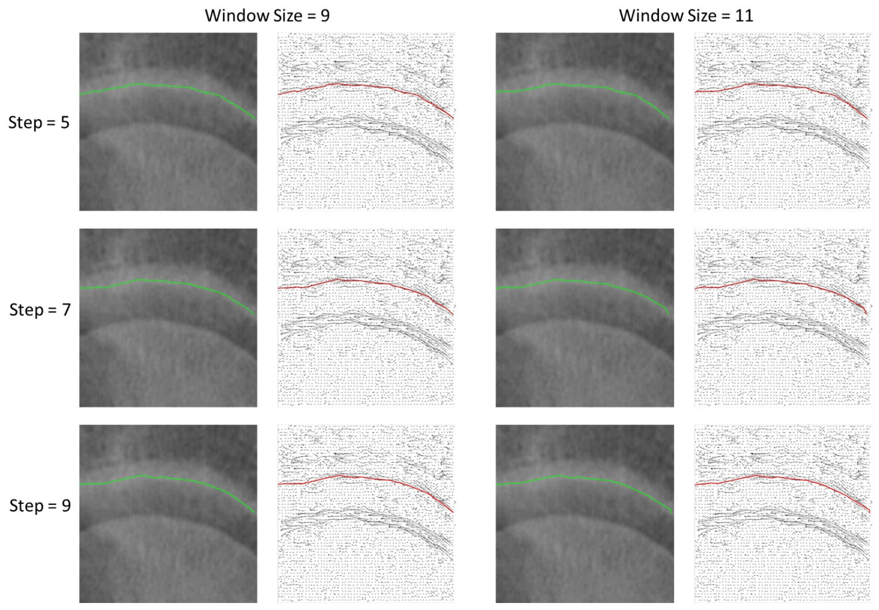

Figure 4.8: Results of upper edges with window sizes 9 and 11, and step sizes 5, 7 and 9.

Figure 4.8 illustrates the results from starting point (38, 60). The differences between these results are tiny, but results with window size 11 are slightly smoother.

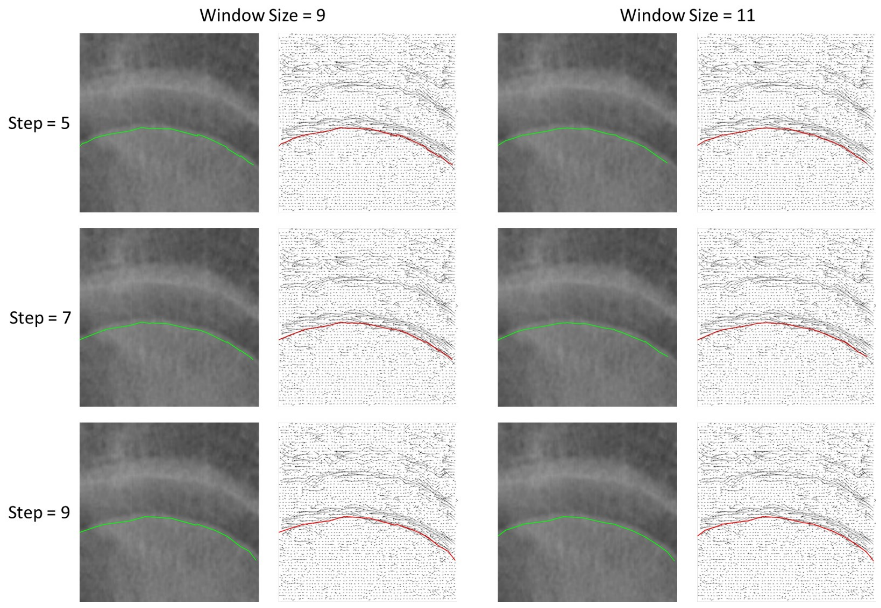

Figure 4.9: Results of lower edges with window sizes 9 and 11, and step sizes 5, 7 and 9.

Figure 4.9 illustrates the results from starting point (66, 60). The edge detected using window size 11 and step size 7 slightly outperforms other settings. All of the edges detected in Figures 4.8 and 4.9 are smooth and continuous. Since window size 11 with step size 7 generates slightly smoother edge curves, we use this setting for other X-ray images.

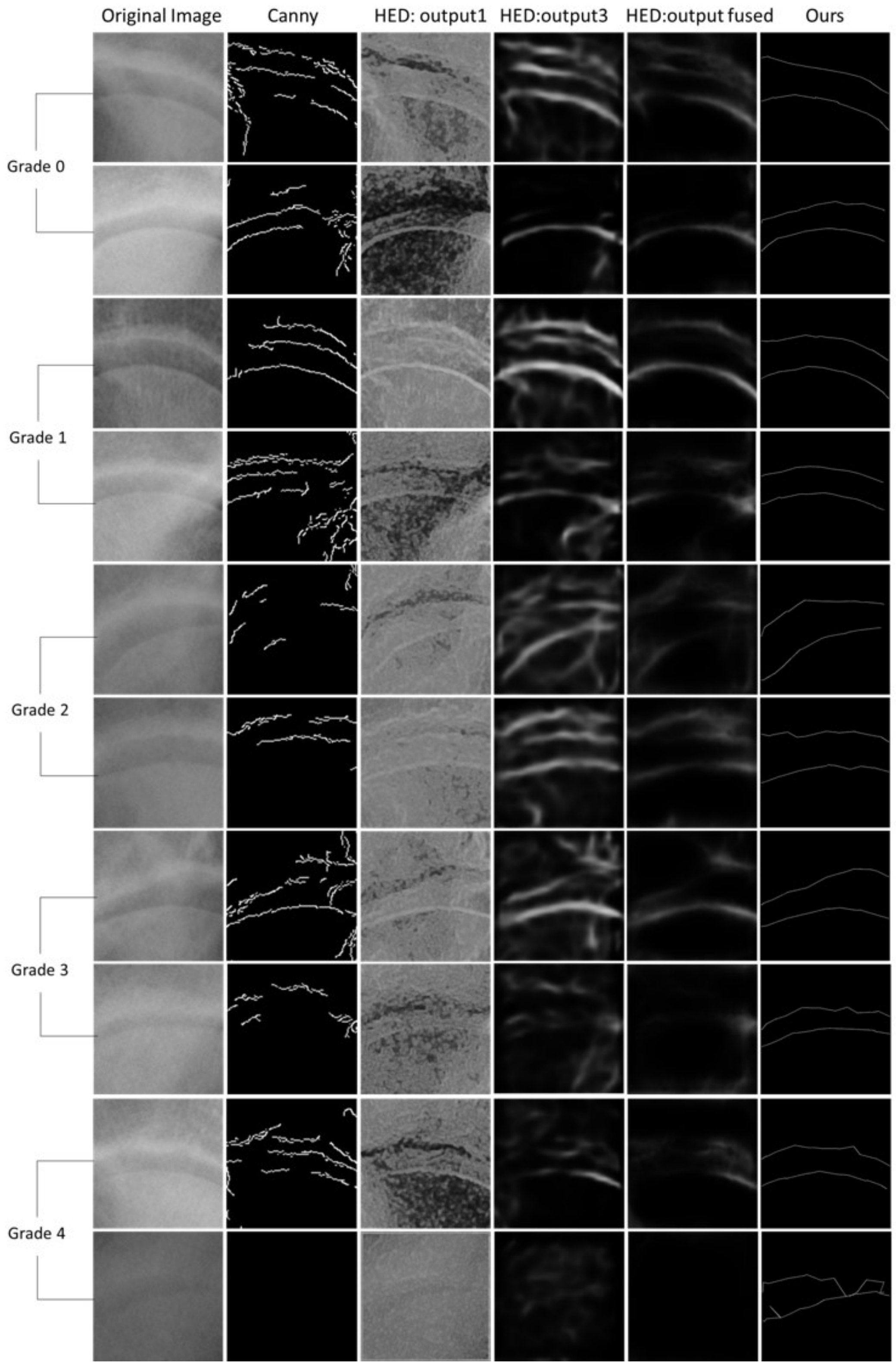

Figure 4.10: Illustration of our results compared with Canny and HED algorithms. The first column are the original X-ray images. The second column are the results of the Canny edge detector. The third to fifth columns are three different outputs of HED. The final column is our method.

Figure 4.10 is a comparison between our method and the state-of-the-art methods, Canny [2] and HED [25]. We compare the methods on two images for each grade of osteoarthritis, from 0 to 4. For most of the images, Canny could not detect the whole edge and also included some noise. HED generates five side outputs with the hidden layers and one fused output of its deep learning network. We compare the three clearest outputs from the six HED results. HED successfully detected the whole edge, but also included a lot of noise. Additionally, HED is computationally expensive. Our method clearly detects the real edges as a thin curve and removes the noise of the original X-ray images. It outperforms Canny and HED.

# Chapter 5

# Conclusion And Future Work

## Conclusion

For osteoarthritis diagnosis, joint space is one of the most critical metrics, typically examined by X-ray imaging. To accelerate and automate the joint space measurement in X-ray imaging analysis, a precise and accurate joint edge detection is imperative. Motivated by this need, we developed two statistical methods for edge detection in X-ray images. Both of the methods are based on the Hessian matrices of the brightness function, and their eigenvectors and eigenvalues.

The first method traverses all pairs of pixels in the X-ray images to calculate the goodness scores, which reflects the possibility of this pair of pixels falling on the edges. As shown in Chapter 3, the first method outperforms the state-of-the-art edge detection approach on X-ray images. To further improve the quality of edge detection, the second method presented in Chapter 4 showed improved accuracy and precision with thinner and more continuous edges detected. The second method performs edge detection by tracing the progression of edges from certain starting points, in this way, the edges detected are clearer, thinner, and more continuous curves. For edge detection for the hip joint, the second method generates better results than the first method.

## Future Work

On the basis of this thesis, there are three aspects of future works that would be of great value to explore. The first is to develop an algorithm to detect the edges for more complex images using the second method. As discussed above, the second method is capable of generating thinner, clearer, and more continuous contour in edge detection when it is performed on images containing limited distracting features, e.g. cropped images containing only the hip joint without other bones or organs. However, for the full X-ray images of the whole abdomen, the fitted edges deviate from the long real edges which generates some noise as shown in Figure 5.1. This is because our method works by locally tracing the curve, which means that when the current point is not near an edge, it still proceeds in some direction, producing a noisy path. With more edges or edge like features in the image, the edge detection quality is compromised. Therefore, future work aimed at tackling this challenge will be of interest. A natural approach to this is to remove detected edge segments which do not correspond to smooth edges with large eigenvalues of the Hessian in the orthogonal direction.

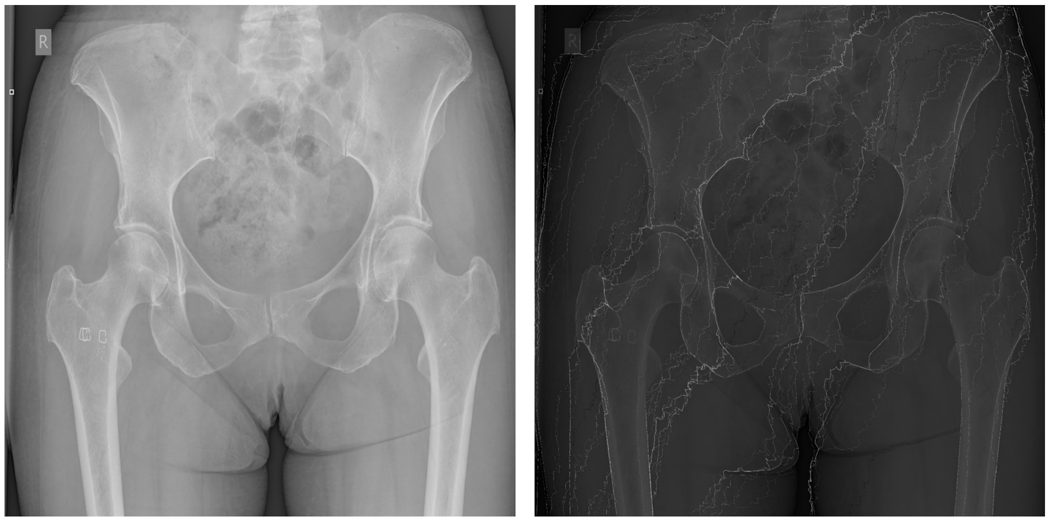

Figure 5.1: Illustration of directly using the algorithm in Chapter 4 to a whole X-ray image. The image on the left is the original image and the image on the right is the superimposed edge detection result.

Besides, developing some metrics for comparing methods for edge detectors especially for edges on medical images is an important future direction. For edge detection results, the accuracy is hard to judge. Subjectivity is not avoidable. Currently, the most widely used method to compare the accuracy of edge detectors is to use the standard database BSDS500, which has 500 images with manually annotated ground truth contours, and three standard measures ODS, OIS, and AP [1] [19] [25]. However, the manually annotated ground truth itself is subjective. An optimization on a loss function regarding the brightness on the edge is a reasonable start. Once we develop a robust method to calculate the accuracy, we can automate the tuning process using a statistical framework, including the kernel size, window size, step size and other tuning parameters, based on the accuracy index. Then, we can pack the computer code and apply our method on new classes of images easily. This is a big project that is worth a lot of efforts.

Another future direction is to apply our edge-detection method to automate osteoarthritis diagnosis. This requires an interdisciplinary effort from both orthopaedics for professional medical input as well as training data, and a fine-tuned supervised learning model to carry out the automatic diagnosis based on the edges detected using this method.

# Bibliography

[1] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4380–4389, 2015.

[2] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[3] Yaroslav Ganin and Victor Lempitsky. $n$4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision*, pages 536–551. Springer, 2014.

[4] Tobias Gebäck and Petros Koumoutsakos. Edge detection in microscopy images using curvelets. *BMC bioinformatics*, 10(1):75, 2009.

[5] Jyh-Jing Hwang and Tyng-Luh Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015.

[6] Scott Konishi, Alan L. Yuille, James M. Coughlan, and Song Chun Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):57–74, 2003.

[7] Q Li, B Wang, and S Fan. Browse conference publications computer science and engineer. help working with abstracts an improved canny edge detection algorithm. In *2009 Second International Workshop on Computer Science and Engineering proceedings: WCSE*, volume 2009, pages 28–30, 2009.

[8] Tony Lindeberg. Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4):349–376, 1993.

[9] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156, 1998.

[10] Baptiste Magnier, Frédéric Comby, Olivier Strauss, Jean Triboulet, and Cédric Demonceaux. Highly specific pose estimation with a catadioptric omnidirectional camera. In *2010 IEEE International Conference on Imaging Systems and Techniques*, pages 229–233. IEEE, 2010.

[11] Stephane Mallat and Sifen Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):710–732, 1992.

[12] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[13] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[14] Giuseppe Papari and Nicolai Petkov. Edge and line oriented contour detection: State of the art. *Image and Vision Computing*, 29(2-3):79–103, 2011.

[15] Charles A Poynton. *A technical introduction to digital video*. John Wiley & Sons, Inc., 1996.

[16] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[17] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[18] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–166, 2004.

[19] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3982–3991, 2015.

[20] Irwin Sobel. History and definition of the so-called sobel operator, more appropriately named the sobel-feldman operator. *Sobel, I., Feldman, G., A 3x3 isotropic gradient operator for image processing, presented at the stanford artificial intelligence project (SAIL) in*, 1968, 2015.

[21] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[22] Yuan Tian, Dong Du, Guorui Cai, Li Wang, and Hua Zhang. Automatic defect detection in x-ray images using image data fusion. *Tsinghua Science and Technology*, 11(6):720–724, 2006.

[23] Scott E Umbaugh. *Digital image processing and analysis: human and computer vision applications with CVIPtools*. CRC press, 2010.

[24] Nitish Premanand Wadker and Amita Dessai. Pubic bone fracture and displacement detection using x-ray images. 2017.

[25] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.