UPPSALA
UNIVERSITET

IT 12 057
Examensarbete 30 hp
November 2012

Stitching of X-ray Images

Krishna Paudel

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Stitching of X-ray Images

Krishna Paudel

Image processing and analysis algorithms are widely used in medical systems to analyze medical images to help diagnose the disease of a patient. This thesis covers one of the demanding problems of a medical system: Stitching of X-ray Images. The flat panel of an X-ray system cannot cover all part of a body, so image stitching is incorporated in the medical system to combine two or more X-ray images and get a single high resolution image. The output of this thesis work is to develop a real-time and user interactive stitching application which works for all X-ray images with different intensity and orientation.

The stitching of X-ray images is carried out by employing two basic steps: registration and blending. The classical registration methods search for all the pixels to get the best registration. These methods are slow and cannot perform well for high resolution X-ray images. The feature based registration methods are faster and always gives the best registration. This thesis evaluates three popular feature based registration methods: HARRIS, SIFT and SURF. The exhaustive nearest neighborhood method has been modified to get faster matching of key points.

The overlapping areas of the composite image are blended to remove the seams and discontinuities. This thesis evaluates some faster blending techniques and incorporates an advanced blending method using blending masks to blend complexly aligned images.

Handledare: Felix Rutscher
Ämnesgranskare: Cris Luengo
Examinator: Lisa Kaati
IT 12 057
Tryckt av: Reprocentralen ITC

Acknowledgements

First of all, I would like to express my gratitude to *Protec GmbH Co. & KG* for providing me the opportunity to work as my thesis work. I would like to thank my supervisor *Felix Rutscher* from Protec for his invaluable suggestion and support during my thesis work. Many thanks to *Frank Baisch*, managing director of Protec, for providing good working environment for me in the company. I would like to thank all the co-workers in the company, I got a lot of support from them; special thanks to *Erkan Basata* & *Michael Müller*.

I am thankful to the *Center for Image Analysis*, Uppsala University for accepting my thesis proposal. I would like to thank my examiner *Cris Luengo* for his invaluable suggestions and providing his invaluable time to correct my report. Thanks to *Xuan Tuan Trinh* for his suggestion regarding format and content of the report. Many many thanks to my friends who helped me directly or indirectly to successfully complete this project. Special thanks to *Mesfin Mamuye*, *Sujan Kishor Nath*, *Ansar Rafique*, *Shirin Pourmoshir*, *Lu Ping*, *Xinping Yang* and many more.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The medical imaging technology involves the creation of images of a body part to diagnose the disease in the patient. The advent of digital technology has made the medical image processing easier and very fast. The very fast computing technology helps physician diagnose diseases by real time and automated processing of medical images. This project *"Stitching of X-ray Images"* takes multiple X-ray images of a body part and creates a single, high resolution image. Stitching of medical images is similar to creation of panorama of a scene using several images of a scene. Google has implemented image stitching technology to display the street view of a city [35].

This report presents the stitching of 2D gray scale medical images; the methods and algorithms can be extended to work for color images too. An X-ray photographic plate is not large enough to fully cover some parts of the body like legs, spines, hands etc. To solve this problem, we capture multiple images of the body part. Then image stitching creates a single high resolution image representing full body part. The single image of the body part makes easy for physicians to diagnose a disease, it is easy to track, manage, store and transmit for electronic medical software.

Algorithms for aligning images and stitching them into seamless photo-mosaics are among the oldest and most widely used in computer vision. Image stitching algorithms create the high resolution photo-mosaics used to produce today's digital maps and satellite photos. They also come bundled with most digital cameras currently being sold, and can be used to create beautiful ultra wide-angle panoramas [30]. Creating high resolution images by combining smaller images are popular since the beginning of the photography [12].

There should be nearly exact overlaps between images for stitching and identical exposures to produce seamless results [34]. The stitching is not

possible if there is not any common region between images. The images of same scene will be of different intensities, scale and orientation and stitching should work or at least give visually appealing output.

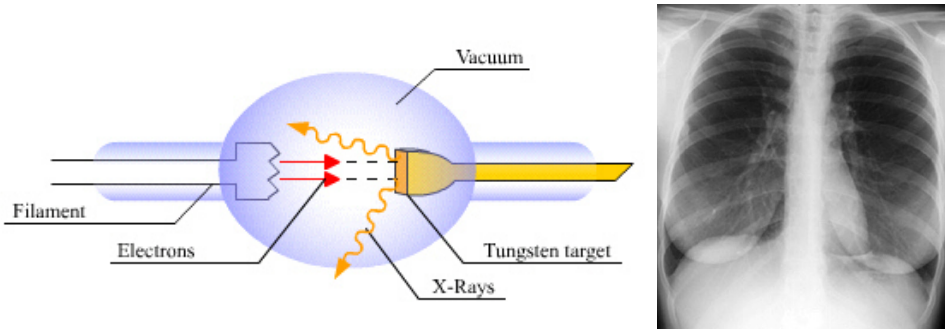
Several algorithms are there to accomplish image stitching, but no algorithm is guaranteed to work with 100% accuracy. The traditional algorithms carry out pixel wise registration (*exhaustive method*) which use some error criteria to get the best result i.e. the best registration is the one which gives least error value. Those methods are slower and sometimes there is chance of not giving the best result. The feature based registration methods find distinctive features in each image and then efficiently match to rapidly establish correspondences between pairs of images to determine the approximate *motion model*. Feature-based approaches have the advantage of being more robust against scene movement and are potentially faster, if implemented the right way[30]. The common feature points are used to create the relationships between the images which makes them suitable for automated stitching.

This first section of this chapter gives introduction to X-ray imaging; describes how X-ray images are produced and their characteristics. The following sections discuss the various types of image alignment methods and the overall stitching process. The challenges of the image stitching have been discussed in the last section.

1.1 X-Ray Imaging

The X-ray technology was developed in 1895 by *Wilhelm Roentgen*, a German physicist. X-rays are produced by changing the energy state of electrons. The highly accelerated electrons are decelerated by bombarding on a metal block. The interaction of electrons in metal block change the energy state releasing X-rays. The X-rays production process is shown in figure 1.1a.

Flat Panel Receptors The X-rays can penetrate soft body parts and go into the bones, so X-ray images are used to view and analyze inner body parts for pathology. X-rays are detected by using *image receptors (IR)*. The new digital technology has been replacing the old technology which uses films and chemicals to store X-ray images. The *Flat Panel Receptors* stores the X-ray images digitally and those digital X-ray images are portable i.e. they can easily be available in multiple places. We can use computer-assisted diagnosis methods to digital X-ray images. Figure 1.2 shows an X-ray machine which consists of X-ray generator and flat panel receptor. The body is placed in between the X-ray



(a) X-ray Tube (b) X-ray image of Chest

Figure 1.1: Production of X-ray: (a) is X-ray tube used to produce X-ray images. (b) is a sample X-ray image.

source and receptor so that it can penetrate the body part. The X-rays which pass through the body part are stored in the receptor.

X-ray images are negative images i.e. dense object like bone or metal fragments display brighter while the soft parts look darker. The X-ray images depicts all the object's features inside and out, so a point in an x-ray image is the summation of shadows; while general photograph shows only object's surface. In other words, the brightness of a point in the film is the summation of all the densities the ray encountered [5].

- *Scaled and Oriented X-ray Images* Since the X-ray tube is movable, the distance of the tube to the body part determines the scale of the image. When X-ray tube is very near to the patient, the image will be distorted because of magnification. Similarly, when X-ray tube is rotated laterally, we get perspective images. We get rotated images because of mobile patient table.

- *Density of X-ray images* The density of X-ray image is the overall blackness or darkness of the film. The flesh, or soft tissue is the least dense and therefore allows for the x-ray to pass easily to the film. Many X-ray photons interact with the film causes the density on the film to black. So, thickness of the body part is inversely proportional to the density of the film. The exposure time, operative voltage peak and current also control the density i.e. if operating voltage and current increases, film density also increases. The overexposed film has a high density, blackens the film and underexposed film has low density means it is white.

- *Contrast of X-ray images* The contrast of X-ray image depicts the difference in degree in blackness between adjacent areas. The image

(a) (b)

Figure 1.2: X-ray Imaging System. (a) X-ray Machine (b) Flat Panel (Image source: http://www.protec-med.com)

with low contrast contains many shades of gray while high contrast image consists of very dark and very light areas. X-ray contrast is produced because X-ray penetration through an object differs from the penetration through the adjacent background tissue. The radiographic recording system should be able to fully record all the contrast in the X-ray image [29].

1.2 Pixel Based Alignment

Before stitching is carried out, two images need to be aligned properly so that the same region in the images overlap each other. Pixel based alignment methods are classical methods which carry out pixel-wise comparison of the two images. We shift or warp images relative to each other and look at how much the pixels agree. The pixel-to-pixel matching methods are also called direct methods. We use suitable error metric (section 3.6) and we carry out exhaustive search for all possible alignments to get optimal alignment. This is very slow process; so hierarchical coarse-to-fine techniques based on image pyramids can be used to make it faster [30].

1.3 Feature Based Alignment

The pixel based alignment methods are not appropriate for real time image stitching applications which includes large (i.e. high resolution) X-ray images. So, *feature based alignment* methods are selected to get faster stitching. The feature based method extract the distinctive features from each image to match those features to establish global correspondence and then estimate the geometric transformation between the images [30]. Interest points (corners) in the image are selected as feature points and the feature descriptors are extracted for each point. The feature descriptors describe the point and for matching process, those descriptors should be invariant to image rotation, scaling or even intensity variations. To find the matching points, the points in one image are compared to the point in another image, an appropriate distance measure is implemented to find out the similarity between the points, and matching pairs have the minimum distance between them.

1.4 Image Stitching Process

In this section, I will describe the fundamental steps of image stitching. Image stitching system gets two or more images as input and the output will be a single stitched image. The image stitching process can be divided into 5 sub-processes mentioned below:

Feature detection This step gets the input images[1] and features of the images are extracted. The important points (also called *key points* or *corners*) in the image are identified using one of the corner detection methods. The concept of corner detection will be discussed in section 3.7. Each feature point will have unique descriptor which is used for feature matching.

Feature matching After we get a number of feature points in the images, the next step is to match the feature points. The similar points in the images are identified using one of the feature matching techniques. The feature matching step gives the best matching point pairs between the images which are used for estimation of motion parameters.

Motion estimation Based on the matching points, we estimate the motion parameters (like transformation, rotation or scale parameters). To estimate the motion parameters, we need true matched points. The false matching points gives wrong motion parameters which produce incorrect alignment. We create a mathematical model with the motion parameters, and the best model is selected which represents most of the matched points (RANSAC) or gives least error value (LMS) as described in chapter 6.

[1] input images should be already preprocessed by noise reduction, intensity leveling etc.

Transformation After we estimate the motion parameters, the next step is to transform the image. The transformation includes translation, rotation, scaling or perspective transform. After transformation, we get the aligned image with overlapping areas lying on the same location of the composite image.

Blending This is final step of image stitching. If the overlapping areas are not exact[2], we get visible lines (seams) in the composite image. So, we use blending techniques to remove those discontinuities. The blending techniques have been discussed in chapter 7.

1.5 Challenges of Image Stitching

Image stitching, in real life medical applications, consists of several challenges to get the better result. The stitching system should be able to work or to some extent give better output result for medical images. In this section, I am going to highlight some of the challenges of image stitching.

Image Noise If image is noisy, there may be chances that stitching methods fail to give accurate result. So, we have to implement some mechanism as pre-processing to suppress or remove the noise to get the better result. The corner-based stitching methods are very sensitive to noise because they give a lot of false corner-points.

Computational Time The stitching methods are slower, if we don't optimize the methods, it takes a lot of time to get the result because of heavy computation (feature based methods) or lengthy process (direct methods) required. The high resolution images contain a lot of pixels in the image, so, the direct methods require a lot of time to align the methods. The feature based methods require heavy computation to get and match the features in the image. The optimization of the methods should be done in such a way that it results acceptable accuracy (trade-off between computational-complexity and accuracy)

Intensity Variation Some image stitching methods are very sensitive to variation image intensities resulting inaccurate stitching. Again, intensity variation in images causes problem in blending also because it creates a seam line in the join of the image.

Image Orientation The images to be stitched need not necessarily in same orientation. The rotation, scaling, distortion between images should be covered by the stitching methods i.e. the stitching methods should give accurate result for rotated, scaled or distorted images.

[2]The overlapping areas are never exact in real problems

Chapter 2

Related Work

This chapter surveys previous work in image stitching. As already said in chapter 1, algorithms for aligning images and stitching them into seamless photo-mosaics are the oldest and most widely used in computer vision. The first image stitching concept was known to be implemented to create panoramas in 1787 by *Robert Barker*, an Irishman, who created panoramas of a cylindrical building [36].

In the past, exhaustive methods were used which calculate a measure of the difference between the images at all possible values in the search space. Those methods were time consuming. If we are registering two images: I_1 of size MXM and I_2 of size NXN, then the computation complexity will be $O(M^2N^2)$ [1].Those methods were used for long time until *Lucas and Kanade*'s patch-based translational alignment[18]. The registration method purposed by Lucas and Kanade [18] became widely popular at that time because it drastically reduced the computational time to $O(M^2logN)$. The matching process was an iterative *Newton Raphson* method where we go on getting better match in each iteration.

Similarly, *Xue Mei* and *Fatih Porikli* [19] have purposed a computationally inexpensive method for multi-modal image registration. Their method employs a joint gradient similarity function that is applied only to a set of high spatial gradient pixels. They used the gradient ascent method to get the maximization of similarity function which gives the motion parameters for best match.

The edge based *Chamfer matching* methods also became popular which used the edge information in the image for matching. In Chamfer matching, we select an image as template and try to match with other image using distance transform as shown in figure 2.1. We can use the various transformed

[1] matching for rotated images,complexity increases.

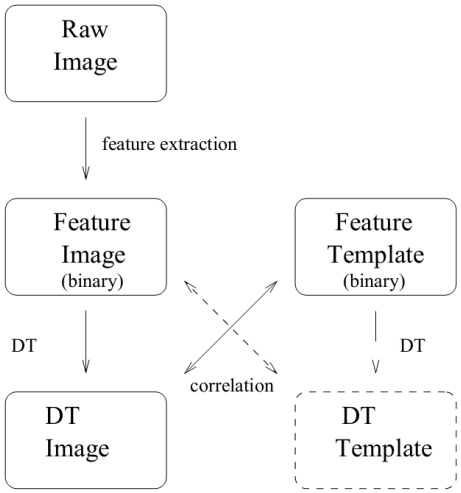


Figure 2.1: Matching using a distance transform. *(Image source: Gavrilla [9])*

templates to match rotated images [9]. Gavrilla and Philomin [10] implemented the Chamfer based matching method in real-time detection of traffic signs and pedestrians from a moving vehicle. They used coarse-to-fine approach over the shape hierarchy and over the transformation parameters to make the matching faster.

There are several research papers which describe the global image alignment methods. The computation of globally consistent alignments has been discussed in the literature by Szeliski and Shum [31] and the variations of exposure has been addressed by Sawhney *et al* [26].

More recent algorithms on image alignment extract a sparse set of feature points and match these points to each other to get the motion parameters [30]. *Brown* and *Lowe* in their paper [2] discusses on obtaining the invariant local features to find the matches between the images and they also claim the method to be insensitive to ordering, orientation, scale and illumination of input images. And there are several research papers which discuss on extracting the feature points in the image. Some basic corner detectors including *Harris* have been discussed by Parks and Gravel [22]. Similarly, the very fast corner detector(*Features from Accelerated Segment Test*) have been purposed by Rosten et al [24]. The more robust feature points extractors (*SIFT* and *SURF*) has been discussed by Lowe [17] and Bay et al [1]. The authors of the papers claim that those feature extractors are more robust and invariant to image rotation, scale or intensity changes.

Chapter 3

Background Theory

There are some basic mathematical and image processing principles that we need to be familiar before we start image stitching. In this chapter, we basically focus on 2D gray scale image processing and its not a big deal to extend the methods for colored images.

3.1 Image Representation

An image consists of information and that should be represented in a form of data structure. This section discusses two main data structures used in image analysis applicable for this thesis project.

3.1.1 Matrices

In matrix representation, each pixel of the image is represented in the form of matrix. The binary images are represented by a matrix containing only zeros and ones. For N-bit gray scale images, the matrix contains the values from 0 to $2^N - 1$. Figure 3.1 shows 8-bit gray scale image in matrix form.

The multispectral images contain multiple matrices to represent each spectrum (e.g. RGB color images are represented by 3 matrices containing red, green and blue values). All matrix related operations (like addition, subtraction, multiplication, scaling, inverse etc.) can be applied to the images represented in matrix form.

3.1.2 Pyramids

Processing higher resolution images is time consuming and are not suitable for interactive system design. So, to make it faster, we process the images in lower resolution to select the interesting parts in image and further processing is carried out to the selected parts in higher resolution [28]. This is achieved by generating matrix pyramids of an image which consists of

$$\begin{bmatrix} 0 & 12 & 53 & 93 & 146 & 53 & 73 & 166 \\ 65 & 32 & 12 & 215 & 235 & 202 & 130 & 158 \\ 57 & 32 & 117 & 239 & 251 & 227 & 93 & 166 \\ 65 & 20 & 154 & 243 & 255 & 231 & 146 & 130 \\ 97 & 53 & 117 & 227 & 247 & 210 & 117 & 146 \\ 190 & 85 & 36 & 146 & 178 & 117 & 20 & 170 \\ 202 & 154 & 73 & 32 & 12 & 53 & 85 & 194 \\ 206 & 190 & 130 & 117 & 85 & 174 & 182 & 219 \end{bmatrix}$$

Figure 3.1: Image represented in matrix form.The elements matrix are the pixel values

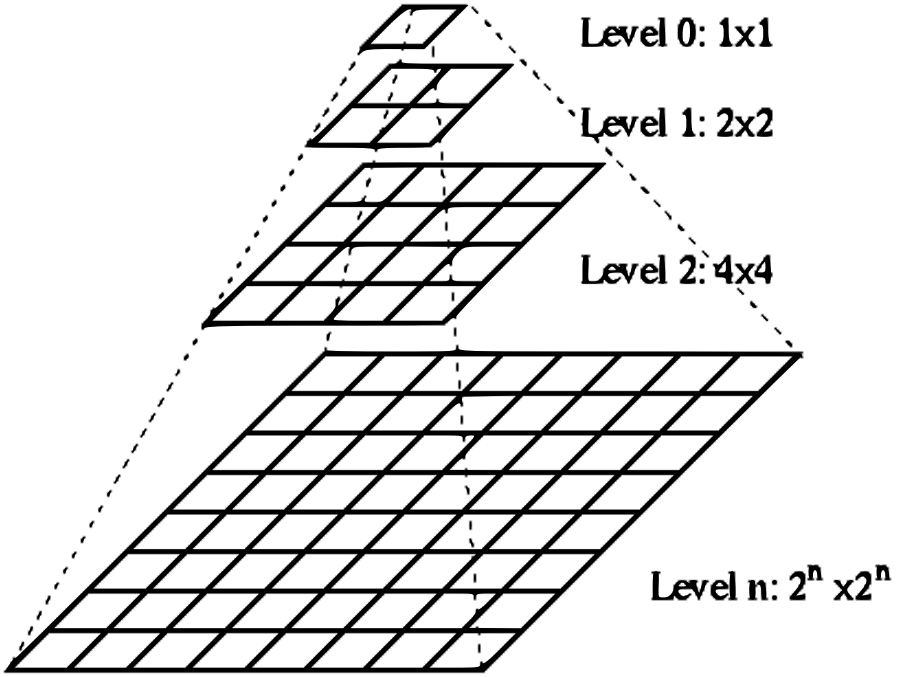
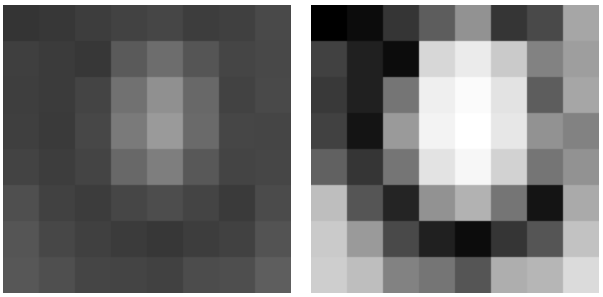


Figure 3.2: Image Pyramids. *(Image source: http://fourier.eng.hmc.edu)*

a sequence of images $\{M_L, M_{L-1}, ..., M_0\}$ (figure 3.2) where M_L has the same dimension and elements as the original image and M_{i-1} will have half resolution of M_i. We can create image up to M_0 (i.e. 1 pixel) if we have square image with dimension multiple of 2. Generally, we will select a level and generate the pyramid of images up to that level. The level to be selected depends upon the specific problem.



(a) (b)

Figure 3.3: Histogram Transformation: (a) is original image; (b) is histogram transformed image.

There are two types of pyramids: low-pass pyramids and band-pass pyramids. In low pass pyramid, we smooth the image with appropriate smoothing filter, and sub sample the smoothed image to create smaller image.[1] Band pass pyramid, on the other hand, is obtained by creating the difference between the adjacent levels in the pyramid. To compute pixel-wise differences, the size of the images should be same, so, we have to implement some interpolation or scaling techniques.

3.2 Brightness Transformation

Brightness transformation is carried out to make the images look more clearer. In brightness transformation, the intensity of the image pixels are changed using one of the following methods:

Brightness Thresholding We select an intensity value P and then the image pixels with intensity less than p are set to zero and other pixels are set to 1 resulting black-and-white image.

Histogram Equalization Histogram equalization is used to enhance contrast by creating an image with equally distributed brightness levels over the whole brightness scale. If an image consists of pixels with limited level of intensities, then the histogram equalization assigns all range of intensities to the pixels which results increase in contrast. For algorithm, please refer to the book by Sonka *et al* [28].

Look-up Table Transformation Look-up table is used to transform brightness in real time. The transformation information of all possible gray levels is stored in look-up table, and the transformation is carried out using

[1] Gaussian pyramid is created using Gaussian smoothing filter. If we go from bottom to top, in each level, the image size is reduced by $\frac{1}{2}$

the table. For example, 8 bit image contains 256 gray levels and only 256 bytes of memory is required for look-up table.

Pseudo-color Transformation The brightness of the pixels are represented by some color value to perceive more detail information. Also human eye is more sensitive to color change than brightness change.

3.3 Geometric Transformation

In geometric transformation, we use a vector function T which maps the pixel (x,y) to a new position (x',y') defined by the following two component equations:

$$x' = T_x(x, y), y' = T_y(x, y) \tag{3.1}$$

The transformation vector function **T** known in advance or sometimes we calculate from original and transformed images by matching of the corresponding pixels.

Pixel Co-ordinate Transformation

The co-ordinates of the input image pixels are mapped to the point in the output image. The geometric transform can be classified as

- *Affine Transform* The affine transform is simple and only 3 pairs of corresponding points are sufficient to find the coefficients.

$$\begin{aligned} x' &= a_0 + a_1 x + a_2 y, \\ y' &= b_0 + b_1 x + b_2 y \end{aligned} \tag{3.2}$$

  The affine transform consists of rotation, translation, scaling and skewing.

- *Perspective Transform* The perspective transform also called *homography* denoted by a 3 x 3 matrix H and the transformation is carried out as:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \text{ and } y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \tag{3.3}$$

  The perspective transforms preserve straight lines and are appropriate for 3D scenes observed under pure camera rotation or planes observed under general 3D motion [30]. We need four pairs of corresponding points for perspective transform.

Brightness Interpolation

The transformation generally results the continuous co-ordinate values (real numbers). The intensity value of specific integer grid in the output image is set by interpolating the brightness of neighboring non-integer samples.

Brightness interpolation influences image quality. The simpler the interpolation, the greater is the loss in geometric influences and photometric accuracy [28]. The most common interpolation methods are *nearest neighbor*, *linear* and *bi-cubic*.

3.4 Image Smoothing

If an image contains a lot of noise, we need to have proper mechanism to reduce the noise using image smoothing methods. Generally, smoothing methods blur the edge information, so if we need to preserve the edge information, then we have to implement "*edge preserving*" image smoothing methods.

Averaging

The noise present in the image at each pixel is an independent random value with zero mean and standard deviation σ. So, if we could get n images of the same static scene, we estimate the average of the images to remove out the noise. It is to be noted that for averaging, we need more than one images of the same scene.

Median Filter

The median filter is very effective noise reduction technique because if we use it appropriately, it can reduce the noise while preserving the edge information [3]. The median value is chosen from the pixels defined by kernel window, and since this process is carried out all the pixels in the image, it is slower method for high resolution image. Median filter is very widely used in digital image processing to remove *speckle noise* and *salt & pepper noise*.

Gaussian Filter

In Gaussian filter, the image is convolved with the Gaussian function to reduce image noise. In digital image processing, a kernel window defines the effective neighborhood pixels. So, larger window size creates more blurred image. Fourier transform of a Gaussian function is another Gaussian, so Gaussian blur has the effect of reducing the high frequency components (i.e. low pass filter).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.4}$$

where * is the convolution operation in x and y, and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{3.5}$$

Apart from smoothing, Gaussian filter can be used to generate different scales of an image as a processing stage in computer vision algorithms. For higher resolution images, the processing on original images might be complicated, so lower resolution scaled image is used for simplicity (section 3.1.2).

The derivative based edge detectors are sensitive to noise, so Gaussian blur filter is commonly used before the edge detection algorithm is carried out. This is called *Laplacian of Gaussian* or *LoG* filtering.

3.5 Edge Detection

The edge detectors are very important in computer vision which helps for image understanding and perception by finding lines, region boundaries etc. Edges are detected by identifying the intensity changes in the image; edges are the pixels in the image where intensity changes abruptly. Edge detection is opposite to smoothing; in smoothing we remove high frequency components while in edge detection, we remove low frequency component in the image.

Edge detection is not a trivial task; we can not create a general edge detector working for all types of images. Generally, edge detectors work approximating the derivative (first or second order) of the image function. There are some operators such as *Roberts*, *Sobel*, *Prewitt*, etc. which approximate the first order x- and y- derivatives of the image and calculate the resulting magnitude and direction of the edges. The alternative methods of edge detection use second derivative of the image function and the edge pixels will be the zero crossings of the second derivative.

The above mentioned derivative based methods are very sensitive to the noise in the image [37] [16]. So, we have to implement some noise reduction mechanism before differentiation of image is carried out.

3.5.1 Laplacian of Gaussian

Before we compute the image derivative, we convolve the image with Gaussian filter to suppress the noise present in the image. Suppose, $f(x, y)$ be image function, $G_\sigma(x, y)$ be the Gaussian kernel of width σ, then

$$\Delta[G_\sigma(x, y) \otimes f(x, y)] = [\Delta G_\sigma(x, y)] \otimes f(x, y) = LoG \otimes f(x, y) \tag{3.6}$$

where

$$LoG = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.7}$$

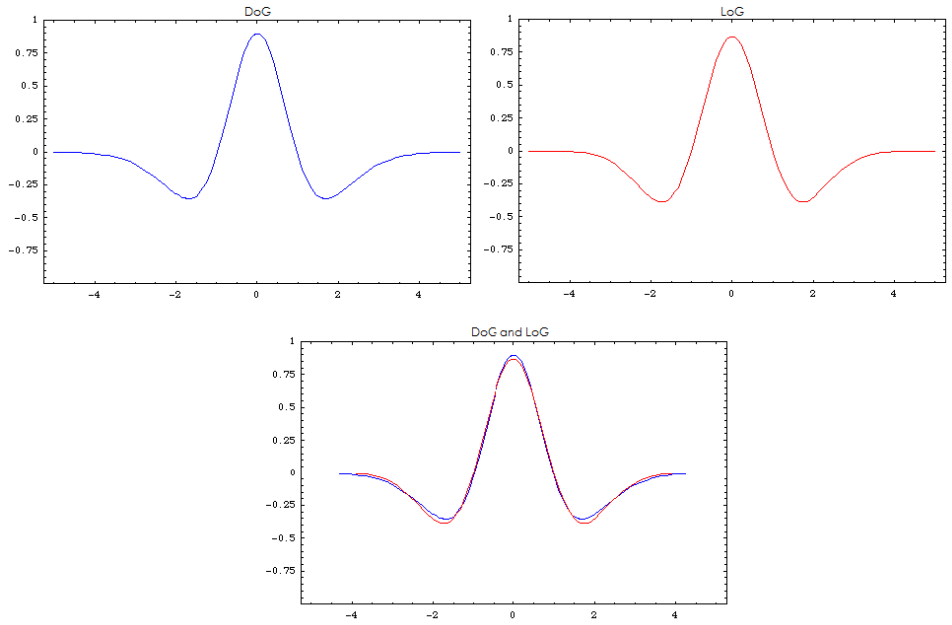


Figure 3.4: Comparison of DoG and LoG.
(Image source: http://en.wikipedia.org/wiki/Difference_of_Gaussian)

Thus, from above equation, we conclude that Laplacian operator to the Gaussian smoothed image is equivalent to applying *Laplacian of Gaussian (LoG)* operator to the original image.

3.5.2 Approximation with Difference of Gaussian

The Laplacian of Gaussian (LoG) can be efficiently implemented using *Difference of Gaussian (DoG)* at different scales. Suppose we used Gaussian kernels $G_{\sigma 1}$ and $G_{\sigma 2}$ to get smoothed images $g_{\sigma 1}(x, y)$ and $g_{\sigma 2}(x, y)$, then

$$g_{\sigma 1}(x, y) - g_{\sigma 2}(x, y) = (G_{\sigma 1} - G_{\sigma 2}) \otimes f(x, y) = DoG \otimes f(x, y) \qquad (3.8)$$

The comparison graph in figure 3.4 shows the similarity between DoG and LoG operators. Thus, we can approximate Laplacian of Gaussian by simply subtracting the Gaussian blurred images in different scales.

3.6 Error Metrics

Error metrics give the measurement of similarity between two images. So, in registration process, we try to align the two images that gives optimal error value. The direct methods of image matching choose a suitable error metrics to compare the images [30] and the search process will try to get the optimal error value.

Sum of Squared Differences

The sum of squared differences (SSD) gives the dissimilarity measurement between two images. So, in alignment process, we try to minimize the SSD value which is calculated as follows:

$$E_{SSD} = \sum_i [I_1(x_i) - I_0(x_i)]^2 = \sum_i e_i^2 \tag{3.9}$$

where $e_i = I_1(x_i) - I_0(x_i)$ is *residual error*.

Correlation

The cross-correlation of the two aligned images are calculated,

$$E_{CC} = \sum_i I_0(x_i) I_1(x_i) \tag{3.10}$$

The cross-correlation value between two images does not give accurate result in case when a very bright patch exists in one of the images [30]. So, *Normalized Cross-Correlation* (NCC) is commonly used,

$$E_{NCC} = \frac{\sum_i [I_0(x_i) - \overline{I_0}][I_1(x_i) - \overline{I_1}]}{\sqrt{\sum_i [I_0(x_i) - \overline{I_0}]^2 [I_1(x_i) - \overline{I_1}]^2}} \tag{3.11}$$

where

$$\overline{I_0} = \frac{1}{N} \sum_i I_0(x_i) \tag{3.12}$$

$$\overline{I_1} = \frac{1}{N} \sum_i I_1(x_i) \tag{3.13}$$

The NCC value 1 indicates the two images are exactly same. To get the best alignment, we transform the image in such a way that it gives maximum NCC value.

Key Points Matching

The key-points (feature points) between the images are identified by using some key-point detectors. We implement matching algorithm to get the matching points. The points which do not get any matching point are counted for both the images to measure the dissimilarity between the images. For perfectly matching images, the count will be zero implies the images are same. The larger the number of count, the more dissimilarity between the images.

3.7 Corners in Image

The geometric transformation parameters are estimated using the position of corresponding points. The same transformation generally hold for almost all pixels of the image and the necessary number of corresponding pairs of points is usually rather small and is equal to the number of parameters of the transformation. The same transformation usually holds for almost all pixels of the image. We have to examine all possible pairs of pixels to find out the correspondence and this is computationally expensive. If two images have n pixels each,the complexity is O(n^2). So, to simplify this problem, we find out the *interest points (corners)* and those interest points are used to find out correspondences for estimation of transformation parameters. The corner in the image can be defined as a pixel in its small neighborhood where there are two dominant and different edge directions[28]. The number of interest points are much smaller than the pixels in the image, so it greatly reduces the computational complexity.

The corner detectors generally use the gray scale image as input and do some processing and the final result is an image with pixel values proportional to the likelihood of the corner and we use thresholds to find out the corner points. We can get the required number of interest points by using proper threshold. Corner detectors are not usually very robust. To overcome this problem, larger number of corners are detected than needed for estimating accurate transformation. This is achieved by decreasing the threshold value. We must be careful it should not be too less; otherwise it might get very large number of corners which makes the further processing very slow. The threshold value is specific to the type and property of the image for example, the image which contains a lot of variations, larger threshold value is capable of giving sufficient number of corners while image containing plain regions might need smaller threshold value.

3.7.1 Requirements of a Corner Detector

Parks *et al* [22] defines some criteria for a corner detector:

- All “true corners” should be detected.
- No “false corners” should be detected.
- Corner points should be well localized.
- Detector should have a high repeatability rate (good stability).
- Detector should be robust with respect to noise.
- Detector should be computationally efficient.

3.7.2 Corner Detection

This section describes the general steps for corner detection. The following basic steps (flowchart in figure 3.5) are carried out by corner detectors:

i. Apply Corner Operator: Gray scale image is as input and for each pixel, the corner operator is applied to get the *cornerness measure* of the pixel [22]. Generally, a small window centered on a pixel is used to measure the cornerness of the pixel. The output of this step is *cornerness map* and it has the same dimension as the input image.

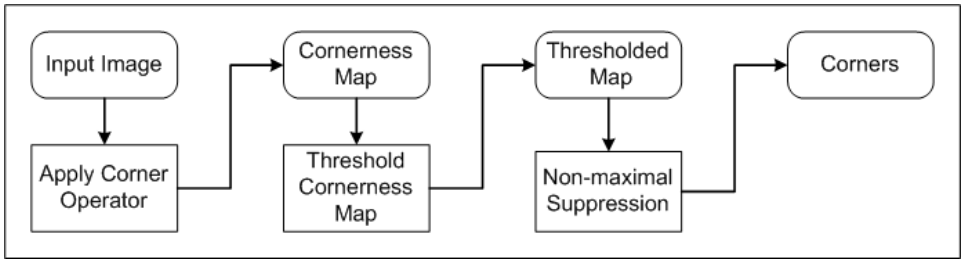


Figure 3.5: Flowchart for corner detectors.

ii. Threshold Cornerness Map: The cornerness map is now thresholded to remove the false corners. We set a threshold value that will be able retain true corners.[2]. There is no straightforward method to choose the threshold value, it is application dependent and requires trial and error experimentation [22].

iii. Non-maximal Suppression: The non-maximal suppression is carried out to get the local maxima of the thresholded cornerness map. A distance value is chosen and cornerness measure of all the pixels within the distance will be zero except the largest one. Then the result is cornerness map with non-zero points as corners.

[2]Generally, there is no threshold value that can remove all false corner while retaining all true corners. So, the appropriate threshold is dependent on application requirements.

Chapter 4

Feature Extraction

Image stitching process starts from feature extraction. In previous chapter, I introduced the basics of corner detection in an image. Each corner in an image has features and we try extract the corners in images and assign features to them. There are some notable advantages using corners as features:

1. Instead of matching all the pixels in an image, we focus on matching corners only. Since the number of corners is much smaller than the total number of pixels, matching is faster.

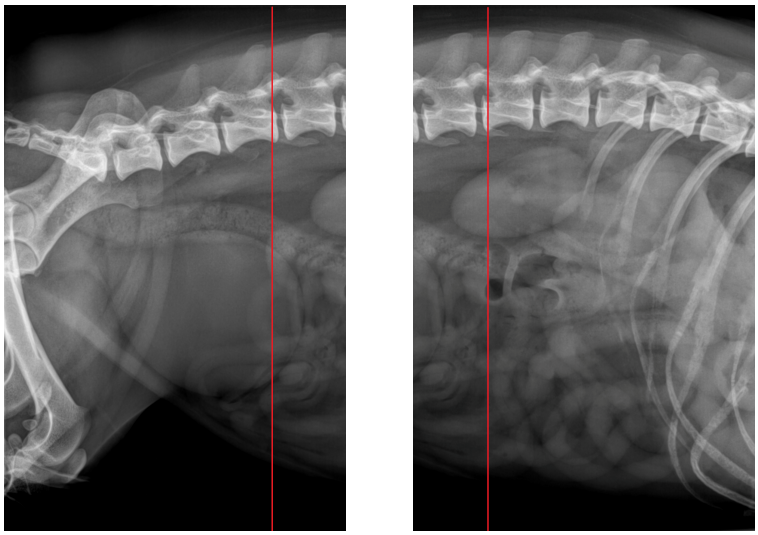
2. Since corner is a point, it is easy to get transformation parameters using the locations of the matching corners.

Like every image processing task, the feature extraction starts with preprocessing (section 4.1) to remove image noise, correct intensity differences etc. There are several algorithms for feature extraction, I have discussed the most popular feature extraction methods (i.e. Harris, SIFT, SURF) in section 4.2.

4.1 Preprocessing

This section discusses some preprocessing techniques before we actually carry out the stitching. The X-ray images might not be perfect for stitching; the noise present in the image cause inaccurate stitching, the high resolution images are slower or sometimes the intensity different between the images gives unpleasant stitched result. The X-ray images should be preprocessed before they are input for feature extraction. For image stitching problem, we generally focus on the following preprocessing steps:

Noise Reduction The noise in the image gives inaccurate result; so noise reduction is a crucial step in stitching. There are several smoothing operators (see section 3.4) which can be implemented to suppress the noise in the image. The smoothing should not reduce the image



(a) (b)

Figure 4.1: The second half of figure (a) and the first half of figure (b) chosen for faster estimation of motion parameters. The vertical line separates the overlapped region.

information that we use for stitching. So, experimentally decide the effective image smoother and its parameters.

Intensity Leveling To get the better stitching result, the images to be stitched should have similar intensity. The intensity differences in the images actually effects the key points identification process because the same image with different intensities result different key points. Similarly, we have to implement more effective blending operations because different intensity images generate a visible seam in the join of images. So, intensity leveling technique makes the image intensities similar to get better stitching result.

Overlapping Area Prediction & Selection Sometimes, if we already know some information regarding alignment, we can use this information to simplify the image stitching task. For X-ray images, images are always either aligned vertically or horizontally. So, if images are aligned horizontally, we can predict that the second half of the first image and first half of the second image have high probability of overlapping. We select the second half of first image and first half of the second image for faster estimation of motion parameters.

4.2 Feature Extraction

There are several corner detection algorithms. In this section, I will describe *Harris Corner Detector*, and modern corner detectors such as *Scale Invariant Feature Transform (SIFT) & Speeded UP Robust Feature (SURF)*:

4.2.1 Harris Corner Detection

The Harris Corner Detection was developed by *Chris Harris* and *Mike Stephens* in 1988 [22]. It is widely used algorithms for corner detection.

The Harris algorithm can be described as follows [22]:

Input:*Gray-scale Image, Gaussian Variance (radius of window=3 x standard deviation),k value, threshold T*
Output:*Map with position of detected corners*

1. For each pixel (x, y) in the image, calculate the *auto correlation matrix* M as follows:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{4.1}$$

where

$$A = \left(\frac{\partial I}{\partial x}\right)^2 \otimes w \tag{4.2}$$

$$B = \left(\frac{\partial I}{\partial y}\right)^2 \otimes w, \tag{4.3}$$

$$C = \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \right) \otimes w \tag{4.4}$$

$\otimes$ is the convolution operator
and, w is the Gaussian window

2. Construct the *cornerness map* by calculating the cornerness measure $Cornerness(x, y)$ for each pixel (x,y):

$$Cornerness(x, y) = det(M) - k(trace(M))^2 \tag{4.5}$$

where

$$det(M) = AB - C^2 \approx \lambda_1\lambda_2, \tag{4.6}$$

$$trace(M) = A + B \approx \lambda_1 + \lambda_2 \tag{4.7}$$

and
k=a constant (generally, k between 0.04 to 0.5 gives good result.)

3. Threshold the interest map by setting all $Cornerness(x, y)$ below a threshold T to zero. The number of corners depends upon the selected threshold T, decreasing T results increment in corners.

4. Perform non-maximal suppression to find local maxima. The non-zero points remaining in the cornersness map are corners.

4.2.2 SIFT

The simple corner detectors like Harris (section 4.2.1) can work only when the images are similar in nature (same scale, orientation, intensity etc) [17]. The SIFT features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. The features are highly distinctive ensuring a single feature to be correctly matched with high probability against a large database of features, thus making it applicable to image registration [17].

SIFT features detection consists of following steps:

1. **Scale-space extrema detection:** This step finds out the potential interest points which are invariant to scale and orientation. This is carried out by using a difference of Gaussian (DoG) function. Extreme points are searched over all scales and image locations. The *Difference of Gaussian* function is convolved with the image to get DoG image $D(x, y, \sigma)$. Mathematically, this can be represented as follows:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{4.8}$$

which is equivalent to

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{4.9}$$

where $G(x, y, \sigma)$ is Gaussian function [equation 3.5]. k is constant multiplicative factor. The *DoG* function is preferred to *Laplacian of Gaussian (LoG)* because it is simple to compute and the result can be close approximation to LoG [17]. David Lowe has derived the relationship of LoG and DoG images as:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\Delta^2 G \tag{4.10}$$

which shows DoG and LoG are differed only by a constant factor $k-1$.

This stage consists of two processes:
Construction of DoG images As shown in figure 4.2, the initial image is incrementally convolved with Gaussians to produce images separated

by a constant factor k in scale space (the left column in the figure). The approximation error will be zero when k is 1 and David Lowe in [17] also claims the stability of extrema even with significant differences in scale even when $k = \sqrt{2}$. For each octave of scale space (doubling of σ),the top image is re-sampled by taking every second pixel in each row and column to create the initial image for next octave with double σ which greatly simplifies the computation. The adjacent images in the stack of the octave are subtracted to produce the DoG images as shown in figure 4.2.

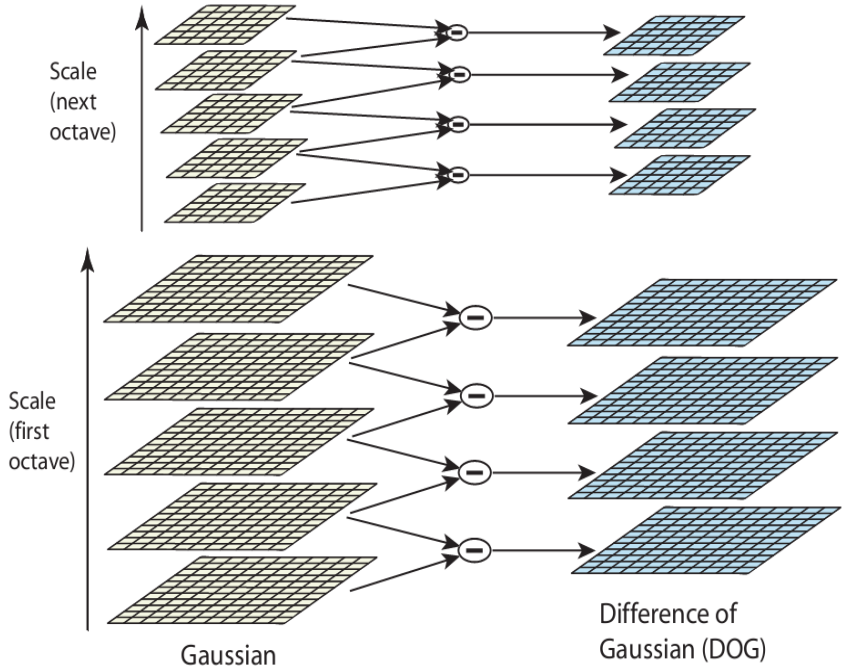


Figure 4.2: Construction of DoG image. *(Image source: Lowe [17])*

Local extrema detection As shown in figure 4.3, the local maxima and minima of DoG images are found out by comparing each sample point to its eight neighbors in the current image and nine neighbors in the scale above and below. In 2D situation, we need to compare 3 DoG images in each octave, so we have to construct 4 different scale images [15].

2. **Key point localization:** This step performs a detailed fit to nearby data for location, scale, and ratio of principal curvatures so that we can remove the points with low contrast or poorly localized along the edge.[17]. To remove low contrast points, the magnitude of intensity is compared with a certain value, if it is less than the value, then reject

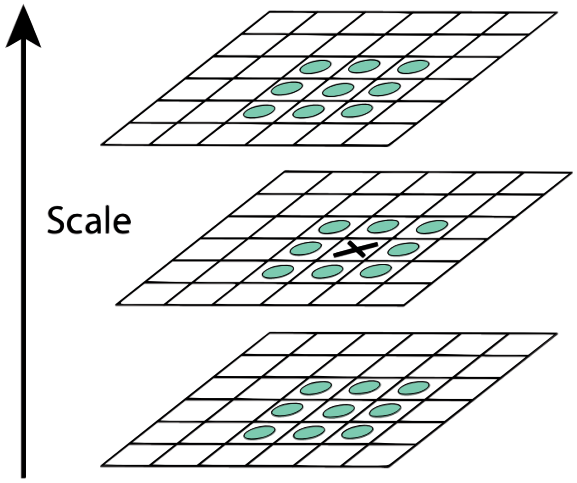


Figure 4.3: Maxima and minima idenfication of DoG images. *(Image Source: Lowe [17])*

it. Brown [2] suggested to use *Taylor expansion* of the value at the extreme points to get the intensity. Similarly, to remove the edges, we use principal of Harris Corner Detector (section 4.2.1) i.e. in any point, if the ratio of largest magnitude eigenvalue and the smaller one is greater than some threshold, then it indicates the edge point. Suppose, r is the threshold ratio of the two principal eigenvalues and M is auto correlation matrix, we check the condition (modification of equation 4.5):

$$\frac{trace(M)^2}{det(M)} < \frac{(r+1)^2}{r} \tag{4.11}$$

3. **Orientation assignment:** The orientation is assigned as key point descriptor so that we can achieve invariance to image rotation [17]. Lowe suggests to use the following method for each key point to have stable results. The magnitude, $m(x, y)$ and orientation, $\theta(x, y)$ of the gradient is computed as:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{4.12}$$

$$\theta(x,y) = \tan^{-1} \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \tag{4.13}$$

Lowe suggests to form an *orientation histogram* from the gradient orientations of sample points withing a region around a key point and detect the highest peak. Any other local peak that is within 80% of the highest peak is used to also create a key point with that orientation. So, for locations with multiple peaks of similar magnitude, there will

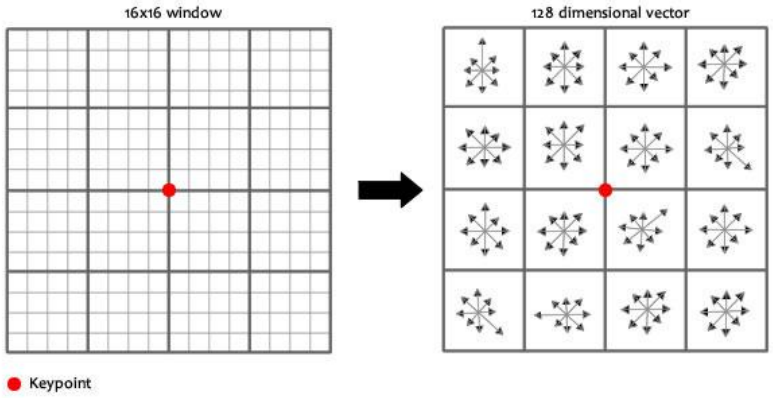


Figure 4.4: Creation of keypoint descriptor. Image gradients are used to create image descriptor. *(Image source: Sinha [27])*

be multiple key points created at the same location and scale but with different orientations [17].

4. **Keypoint descriptor** In above steps, we defined the image location, scale, orientation of each key point. In this step, we compute a descriptor which is highly distinctive for each key point while being invariant to change in illumination or 3D viewpoint. This can be achieved by selecting a window around the key point and generating the feature vectors. Suppose we selected 16x16 window, then we break it into 4x4 windows as shown in figure 4.4. For each window, we calculate the gradient magnitude and orientation at each point. Again, the magnitude is weighted according to the distance from key point (use Gaussian weighting function [17]), i.e. gradients that are far away from the key point will add smaller values. A 8 bin orientation histogram thus carries 8 features for each small window. Thus for 16x16 window, we can have a feature vector of the key point with 16x8=128 features in it which uniquely describes the key point.

4.2.3 SURF

The dimension of the descriptor has a direct impact on the time it takes and less dimensions are desirable for fast interest point matching [1]. SIFT has 128 dimensional feature vector, which makes it very slow making it not suitable for real time applications. the high dimensionality of the descriptor is a drawback of SIFT at matching step. For online applications replying only on regular PC, each one of the three steps (detection, description, matching) has to be fast. Therefore, SURF method of feature detection has been developed to overcome the drawback of SIFT i.e. making matching

algorithm faster by creation of low dimensional feature vector.

The SURF method can be described by the following two steps:

1. **Interest Point Detection**

   - *Integral Images* The computation time is reduced drastically by using an intermediate representation of the image called *integral image* [33]. The value at a location $\mathbf{X}=(x,y)^T$ of integral image $II(\mathbf{X})$ is calculated as:

$$II(\mathbf{X}) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j) \tag{4.14}$$

     After computing of integral image, we can calculate the intensity of any vertical rectangle as shown in figure 4.5. The calculation time is independent of the the size.

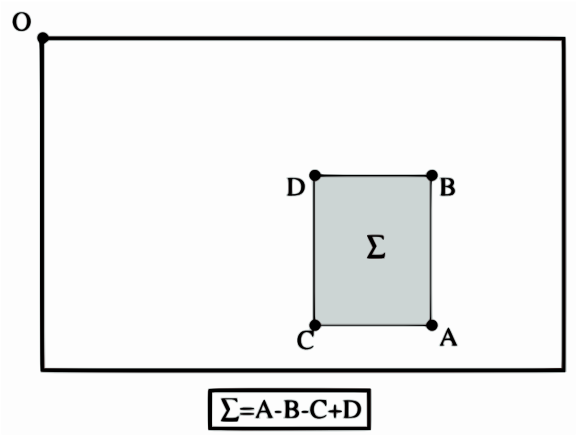


Figure 4.5: Calculation of sum of intensities inside a rectangular region of integral image. *(Image source: Bay* et al *[1])*

   - *Hessian matrix* For any point $\mathbf{x}$ = (x,y) in image I, the Hessian matrix $H(x,\sigma)$ can be defined as:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{4.15}$$

     where

$$L_{xx}(x,\sigma) = \frac{\partial^2}{\partial_x^2} g(\sigma) \otimes I \tag{4.16}$$

$$L_{yy}(x,\sigma) = \frac{\partial^2}{\partial_y^2} g(\sigma) \otimes I \tag{4.17}$$

$$L_{xy}(x,\sigma) = \frac{\partial^2}{\partial_x \partial_y} g(\sigma) \otimes I \tag{4.18}$$

The Gaussian functions are discretized and cropped, so there is some loss of repeatability of Hessian based detectors under image rotations; but this is out-weighted by the advantage of fast convolution by the descretization and cropping [1]. The approximated determinant of Hessian matrix represents the blob response in the image at location **x**. Bay *et al.*[1] suggests to use 9x9 box filters as shown in figure 4.6 to create the blob responses D_{xx}, D_{yy}, D_{xy}. Then computation of determinant of Hessian i.e. $det(H_{approx})$ is computed as follows:

$$det(H_{approx}) = D_{xx}D_{yy} - w(D_{xy}^2) \tag{4.19}$$

here, w is used to balance the expression for the Hessian's determinant. Generally, it does not have a significant impact on the result, so we keep this value as constant[1].

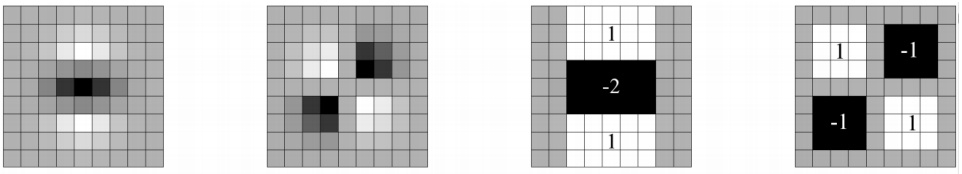


Figure 4.6: Left two images are cropped and decretized Gaussian second order partial derivative (L_{yy}, L_{xy}), right two images are corresponding approximated filters (D_{yy}, D_{xy}).

- *Scale Space Representation* The scale space is analyzed by up-scaling the box filter size rather than iteratively reducing the image size (figure 4.7). The output of the 9 x 9 filter is the initial scale layer (scale=1.2, because the approximation was with σ=1.2). By doing this, we achieve computational efficiency and there will be no aliasing because we don't need to down-sample the image [1].

  The scale space is divide into octaves. An octave represents a series of filter response maps obtained by convolving the same input image with a filter of increasing size. Each octave is divided into a constant number of scale levels as shown in figure 4.8

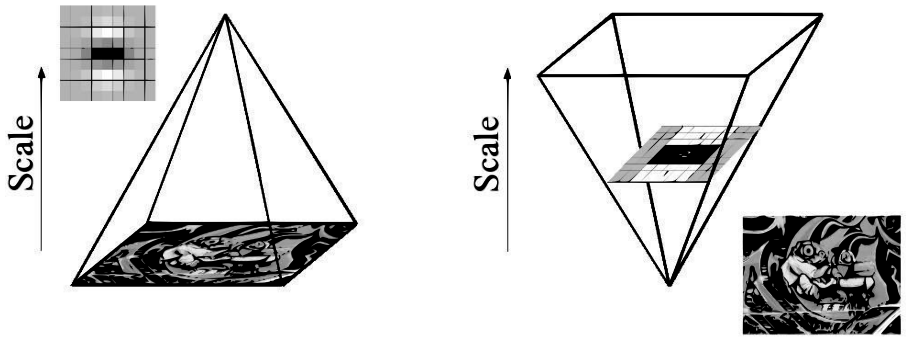


Figure 4.7: The use of integral images allows up-scaling of the filter at constant cost.

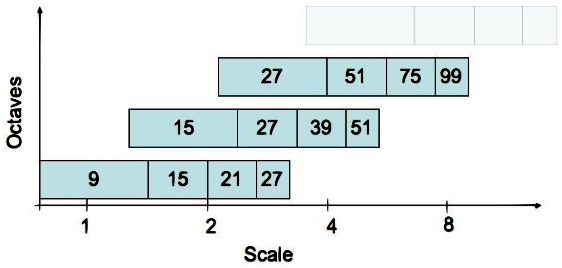


Figure 4.8: Filter side lengths for three different octaves. *(Image source: Bay* et al *[1])*

- *Interest Point Localization* After successful, scale-space creation, the next task is to localize the interest points.To localize interest points in the image and over scales, a non-maximum suppression in a 3 x 3 x 3 neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space.[1].

2. **Interest Point Description** The method of descriptor extraction is similar to SIFT described in previous section. The distribution of first order Haar wavelet responses in x and y direction (figure 4.9) instead of the gradient is used for descriptor extraction. We exploit integral images for speed. The use of only 64 dimensional feature vector greatly reduces the time for matching while increasing the robustness [1]. The authors of SURF claims the new indexing step based on the sign of the Laplacian increases the robustness of the descriptor and matching speed also; so the name SURF-Speeded-Up Robust Features.

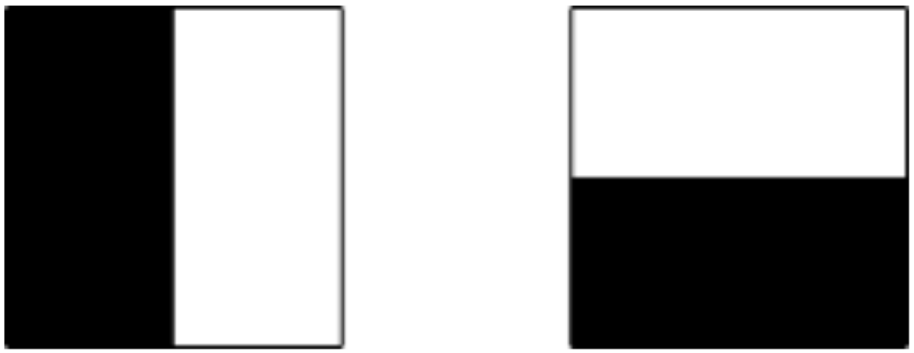


Figure 4.9: Haar Wavelets: The left one is response in x-direction, the right one is response in y-direction. Weight=1 for black region, -1 for white region. *(Image source: Evans [6])*

The interest points descriptors are assigned by the following two steps:

- *Orientation Assignment* Orientation assignment is carried out for rotation invariance. Haar wavelet responses of size 4σ are calculated for a set of pixels within a radius of 6σ of detected point.[1] The specific set of pixels is determined by sampling those from within the circle using step size of σ [6]. Weighted responses with a Gaussian centered at the interest point are used to find the dominant orientation in each circle segment of angle $\frac{\pi}{3}$ (i.e. 60°) around the origin. At each position, the x and y- responses within the segment are summed and used to form a new vector. The longest vector is the orientation of the interest point [6].This process is shown in figure 4.10.

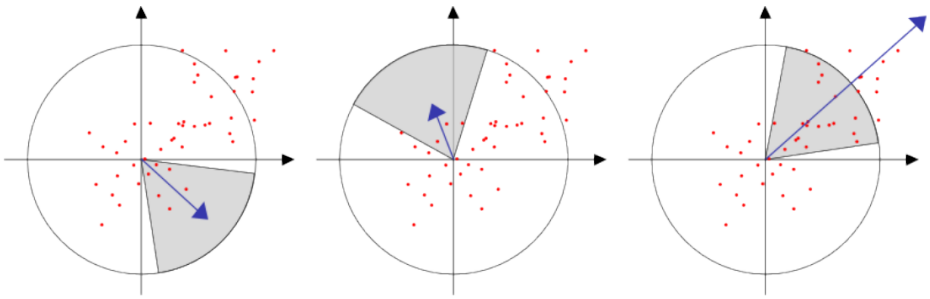


Figure 4.10: Orientation Assignment: The blue arrow is the sum of the responses. The largest one determines the dominant orientation. *(Image source: Evans [6])*

- *Descriptor Components* We construct a square window of size 20σ

[1] σ is the scale at which the point was detected.

around the interest point. The orientation of the window will be the dominant orientation we calculated above. This descriptor window is divided into 4 x 4 regular sub-regions. We select 25 regularly distributed sample points in each sub-region and Haar wavelets of size 2σ are calculated for those points [6]. Then, the feature vector for each sub-region will be

$$V_{subregion} = [\Sigma dx, \Sigma dy, \Sigma |dx|, \Sigma |dy|] \tag{4.20}$$

As shown in figure 4.11, each sub-region will have four values to the descriptor vector which results the overall vector length 4 x 4 x 4=64. Evans in his article [6] claims that the result SURF descriptor is invariant to rotation, scale, brightness and contrast[2].

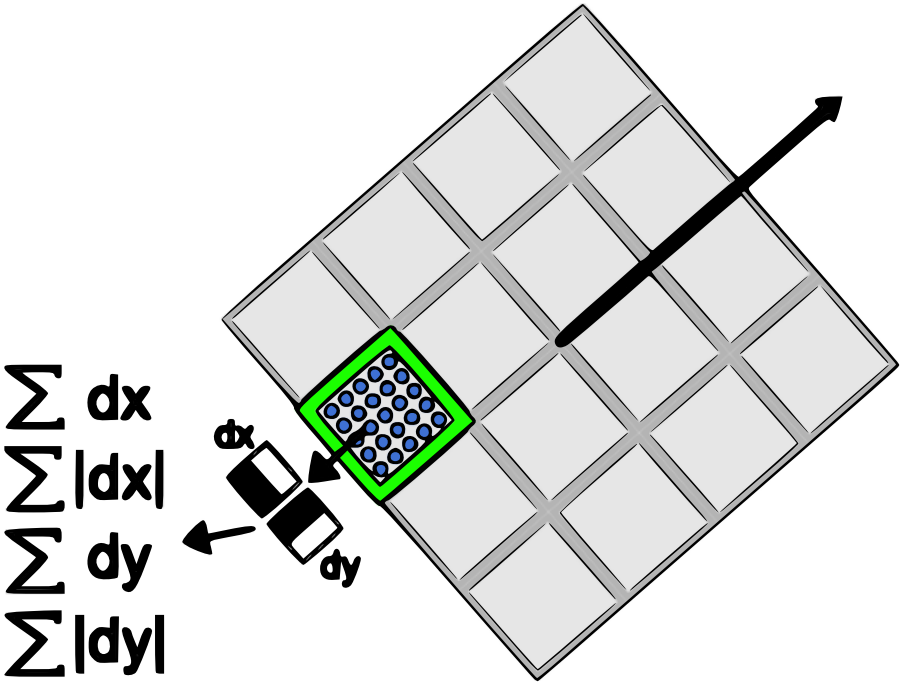


Figure 4.11: Descriptor Components. *(Image source: Evans [6])*

[2]invariant to contrast is achieved after reduction to unit length [6].

4.3 Experimental Results

In this section, I will present experimental results on feature extraction and reason for choosing specific algorithm. I used Intel *OpenCV (Open Source Computer Vision)* library version 2.3 to test the performance of the feature extractors. I developed the algorithms using *C++* in *Microsoft Visual Studio 2010*. I carried out tests in my machine with following configurations:

- Operating System: Windows 7 Professional 64 bit
- Processor: Intel Core i5 2.80 GHz
- Memory: 8.00 GB

I will focus on the following two measures to select best algorithm:

Accuracy The inaccurate stitching result may give false information to the analysts; we choose the algorithm which results best accuracy for medical images. I will check the stability of the detected corners for different intensity, orientation, and scaling.

Computational Complexity High computational complexity will take longer time to perform a task which is undesirable. We try to get the algorithm which has real time performance. Sometimes, we tweak the algorithms to work faster (parallel processing for example). There is trade-off between computational complexity and accuracy. So, we select the algorithm which is faster and gives acceptable result.

I have tested 3 popular feature extractors: Harris, SIFT and SURF. A good corner detector should fulfill certain requirements as discussed in section 3.7.1, so I will evaluate the feature detectors based upon those requirements i.e. computational time and repeatability rate of key-points for different intensity, orientation, scaled images.

To evaluate the performance of the detectors, I have used the different images mentioned below:

Normal Image This is normal X-ray image of resolution 666 x 1024 (i.e. 681984 pixels). This is standard image and I have modified this image to create other different intensity or orientational images.

High Intensity Image The pixels of the normal image are increased by 25% to get high intensity image. The high intensity image and normal image have same number of pixels.

Rotated Image The normal image rotated by 8° and the unassigned pixels have been removed and the image size is 538 x 892 (i.e. 479896 pixels). Thus, the rotated image pixels have been reduced by 30%.

Scaled Image The height and width of the normal image are increased to get scaled image. The resulting scaled image size is 1138x1751 (1992638 pixels) which contains nearly double the pixels in the normal image.

Noisy Image The normal image consists of very little noise. To carry out the tests for noise image, I have added some Gaussian noise (with mean=0, variance=0.001) to the standard image to get the noisy image.

4.3.1 Experiment 1: Computational Time

I have estimated the computational time for the feature extractors for different intensity and orientation images mentioned above. The computational time of the key-point detectors has been presented in chart shown in figure 4.12.

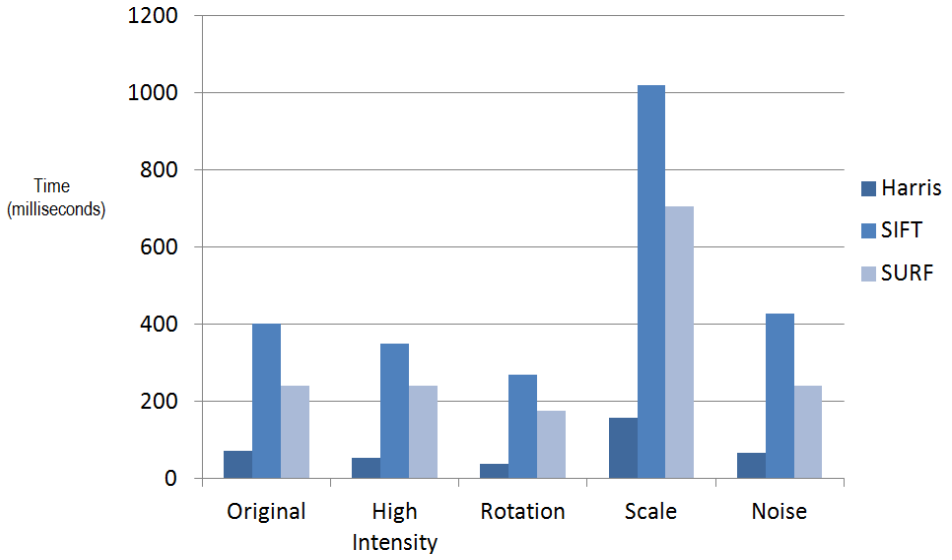


Figure 4.12: Chart showing the computational time (milliseconds) of feature extractors for different images

The original, high intensity and noisy images contain same number of pixels; the time taken by the detectors seems similar for those images. For rotated image, the pixels are reduced by 30%, the computational time also decreases. Similarly, the detectors are slower for scaled image because it contains almost double the number of pixels in original image.

The chart shown in figure 4.12 clearly reveals that Harris corner detector is the fastest among detectors and SIFT is the slowest for all types of images.

Keeping this in mind, we go on further experiments to evaluate the accuracy of the feature detectors to figure out the best feature extractor.

4.3.2 Experiment 2: Stability

In this experiment, the stability of the feature detectors is computed for different intensity or orientation images. The stability of the key-point detectors is an important property because it determines the accuracy of the detected key-points. To measure the stability of a key-point detector, we apply key-point detecting algorithm to the standard image (i.e. normal image) and other images; then count the number of key-points. The repeated key-points lie on the same feature location of the standard image. The repeated key-points on the other images are counted. Suppose, we got N key-points in standard image and N' key-points in other image; out of which $N_{repeated}$ are repeated. We estimate the stability of the key-point detector as follows:

$$Stability = \frac{N_{repeated}}{N} \tag{4.21}$$

The stability is generally expressed in percentage (%) and the higher the stability the better is the key-point detector. So, we select the feature detector which gives highly stable corners.

Table 4.1 presents the detected key-points count for original image and other variant images. I have included the counts for repeated key-points to estimate the stability of the feature detector.

	HARRIS		**SIFT**		**SURF**	
	Original image: 245		Original image: 306		Original image: 291	
Image	**Identified**	**Repeated**	**Identified**	**Repeated**	**Identified**	**Repeated**
Brighter	183	136	261	225	255	234
Rotated	173	85	196	139	177	125
Scaled	481	82	265	259	299	164
Noisy	273	174	488	290	325	214

Table 4.1: Feature points count for stability measurement

The count of generated and repeated key-points of the images are used to compute the stability of the feature extractors (using equation 4.21 above). The stability of the key-point detectors for different images has been presented in chart shown in figure 4.13.

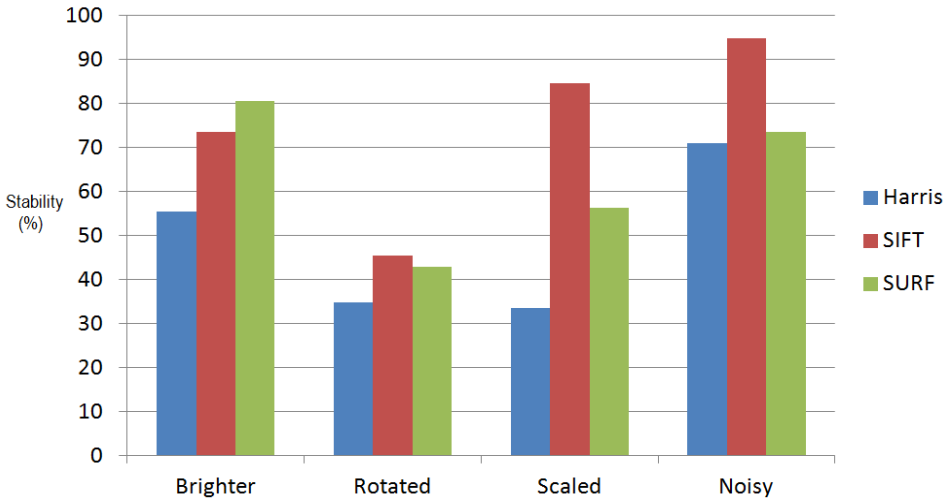


Figure 4.13: Chart showing the stability of feature extractors

The chart presented in figure 4.13 shows that SIFT has high stability for all types of images while Harris giving low stable corners. It is interesting that all key-point detectors have high stability for noisy image. It is because the detectors generate very large number of key-points for noisy image (see table 4.1). Therefore, we get higher number of repeated key-points which results higher stability. The less stable key-points for rotated images is because of the decreased size of rotated image i.e. some part of the rotated image is trimmed off to remove unassigned pixels.

In conclusion, Harris corner detector is computationally efficient; but produces less stable key-points which means it generates more inaccurate or inconsistent key-points for different intensity or orientational images. Stability is an important feature for key-point detectors and SIFT and SURF are good to generate highly stable key-points. SIFT is slower but giving highly stable key-points while SURF is faster but less stable than SIFT. So, both SIFT and SURF are chosen as the good key-point detectors. In the next chapter, we will evaluate the matching performance of SIFT and SURF.

Chapter 5

Features Matching

The next step after feature extraction is feature matching. As explained in section 4.2, we will get a set of key points and their descriptors for each image. The matching process uses the descriptor information to find out the corresponding matching points in the images. All the points in one image are compared to all the points in other image and best matched point is identified. The nearest-neighborhood based algorithms are basically used for feature matching and since those methods are slower, we have to optimize to get the faster matching. The first section of this chapter gives exhaustive *k-nearest-neighborhood (kNN)* method, then we discuss *approximate nearest neighborhood (ANN)* method in the following section. The fine-tuning of the matching points has been described in last two sections.

5.1 kNN Matching

The k-nearest neighbor (kNN) search problem is widely used in classification problem. If we have a set P of reference points $P = \{p_1, p_2, p_3, ..., p_m\}$ in d-dimensional space, and suppose q be the query point defined in the same space, then kNN search problem determines the k points closest to q among P as shown in figure 5.1.

5.2 ANN Matching

Since, we have high dimensional feature vector [1], and obviously we will have a lot of key points. So, in image stitching problems, exhaustive matching process (such as kNN) is very slow, we select an alternative approximate nearest neighborhood matching i.e. *ANN Matching* where priority-search is carried out on hierarchical k-means trees [20]. The nearest points need not necessarily be the actual matched points, further tests are necessary to

[1] Each SIFT point has 128 features while a SURF point has 64 features

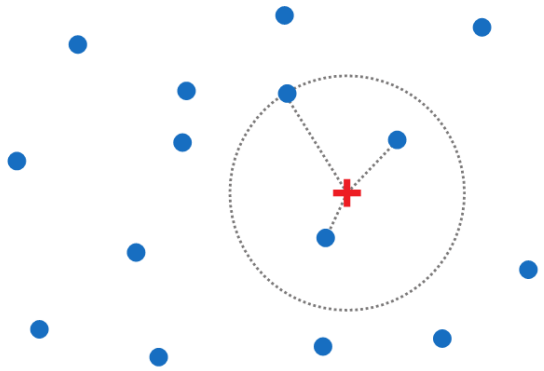


Figure 5.1: kNN search problem with k=3. *(Image source: Garcia et al. [8])*

increase matching accuracy (e.g. Ratio Test, Symmetry Test).

The ANN algorithms can be orders of magnitude faster than kNN search, and provides nearest optimal accuracy [20].

5.3 Removing False Matches

The nearest neighborhood based feature matching techniques mentioned above might contain a lot of false matches. Before we go for estimation of transformation parameters, we have to remove those false matches. This section describes the two effective methods to remove false matches: *Ratio Test* and *Symmetry* Test [13].

5.3.1 Ratio Test

For kNN search, if we set $k = 2$, the matching finds the two nearest points. The ratio test is based upon the ratio of distances of the two nearest points. Suppose d_1 and d_2 $(d_1 < d_2)$ be the distances of a point to its nearest matched points, and a we define a threshold T. Then, we reject the points as false matches which satisfy the following condition:

$$\frac{d_1}{d_2} > T \tag{5.1}$$

If two nearest points are almost same distance, the ratio tend to be higher which implies the false matches, so ratio test confirms the nearest point should be very near and other point should be far. To what extent we remove the points depends upon the threshold T selected i.e. higher the threshold, larger number of matches. Generally, we select T=0.8 i.e. if the distance ratio between the two nearest points is greater than 0.8, then ratio test discards that matching pair. Generally, we get more accurate matches

by decreasing threshold value; but this is not always true. In some cases, if the image contains similar points (i.e. $d_1 \approx d_2$) in different locations, then ratio $\frac{d_1}{d_2}$ for a match might be higher than threshold (T) and the match is discarded.

5.3.2 Symmetry Test

In symmetry test, we compute the matches between the images in both direction and select the matches which pass the symmetry test i.e. any matching pair should be symmetrically valid to be accurate match; otherwise the match is discarded as false match. This is very effective test to identify the false matches and we generally prefer to carry out this test after ratio test.

Suppose, a key-point p_1 in the first image gets a matched point p'_1 in the second image, then match pair $\{p_1, p'_1\}$ to be an accurate match, the key-point p'_1 in the second image should have matched point p_1 in the first image. If p'_1 gets other key-point as matched point then *Symmetry Test* discards the matched pair $\{p_1, p'_1\}$. This test is simple to implement and we do not need any parameters to perform this test.

5.4 Experimental Results

In this section, I will evaluate the feature matching methods in term of computational complexity. Section 5.4.1 compares SIFT and SURF features matching and section 5.4.2 compares the nearest neighborhood methods (i.e. kNN and ANN) for feature matching.

5.4.1 SIFT versus SURF

To carry out the evaluation between SIFT and SURF, I varied the number of feature points in first image by changing threshold values while the number of feature points in the second image made fixed with constant threshold value. The chart presented in figure 5.2 shows that SURF is quite faster than SIFT. The computational cost for SIFT increases drastically as the number of points increases. I used exhaustive kNN matching to match the features.

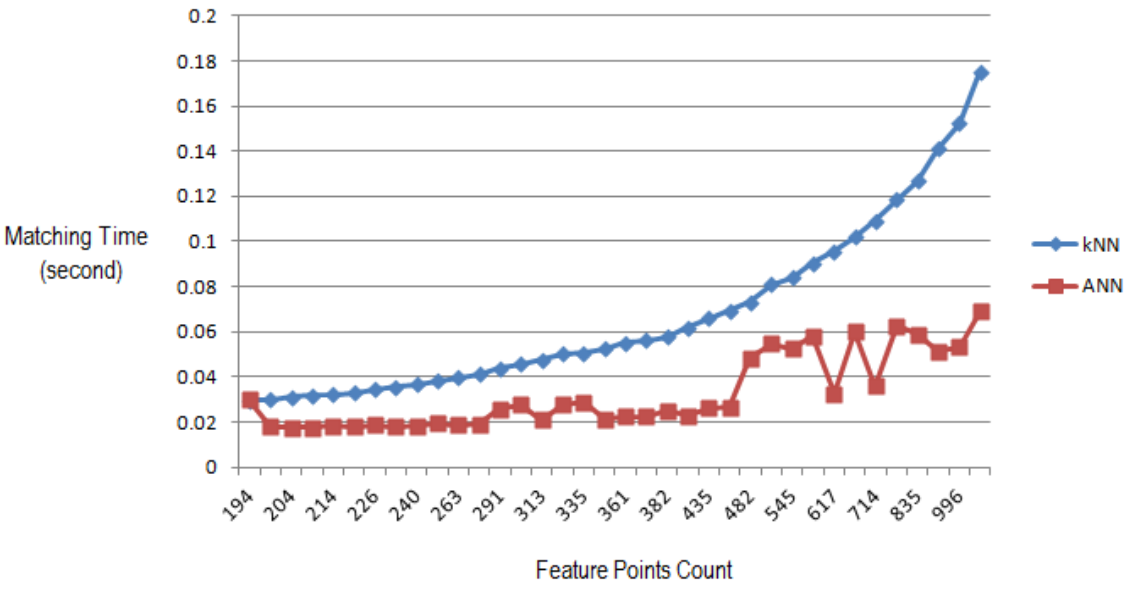


Figure 5.2: Comparison of SIFT and SURF: Feature points versus computational time

SIFT is more accurate because of its high dimensional features, its high computational time is the main drawback making it inappropriate for real time or user-centered applications like medical software. I have chosen SURF method because it is faster and still it gives accurate matches.

5.4.2 kNN versus ANN

In this section, I will present experimental results on kNN and ANN matching methods for SURF features (64 dimensions) which is presented in graph shown in figure 5.3. The graph clearly shows that ANN matching always faster than kNN. kNN matching is showing exponential increase in computational complexity when key-points are increased which implies that kNN becomes expensive for large key-points. In practice, we have larger number of feature points to be matched, so ANN matching is preferred to kNN.

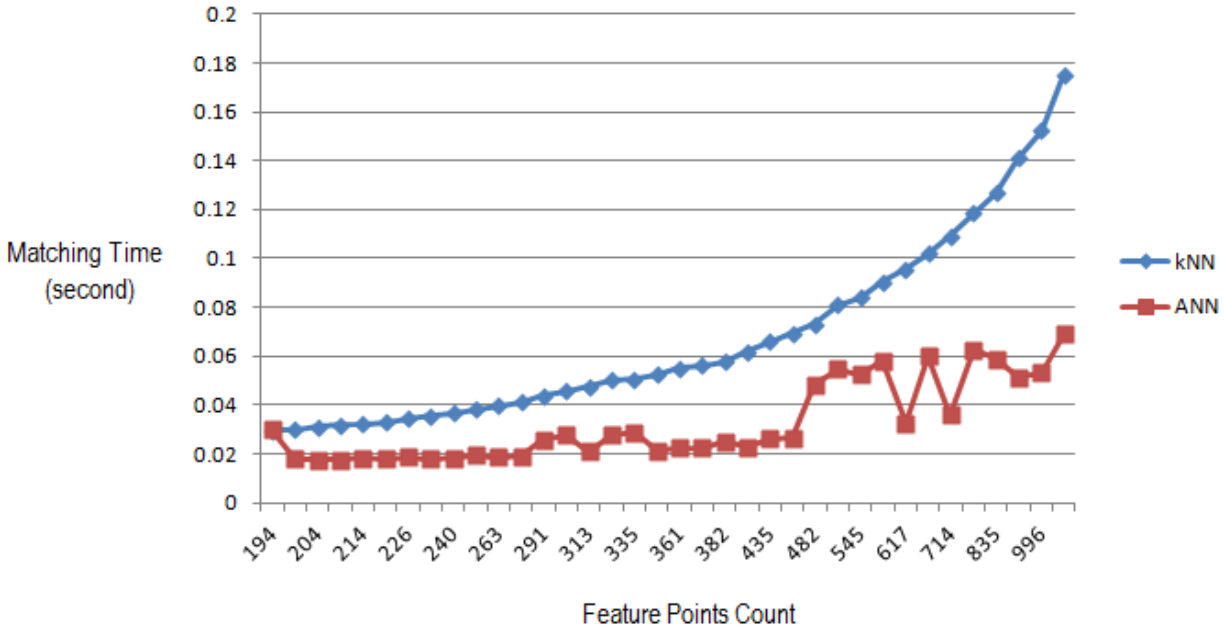


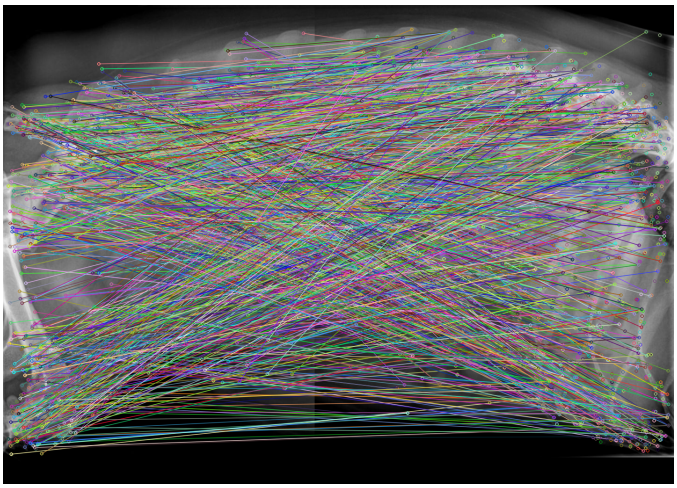
Figure 5.3: Comparison of kNN and ANN: Feature points vs. matching time

5.4.3 Getting Accurate Matches

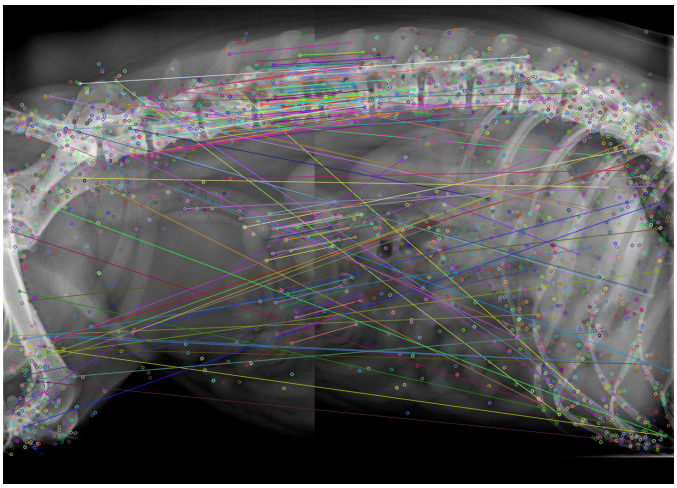
The nearest neighborhood method just gives the closest point as the matching point which is not necessarily the true match. So, we have to implement tests to remove the false matches identified by nearest neighborhood method. In this section, I implemented some tests to increase the accurate matches by identifying possible false matches. The increment of accurate matches helps to get more accurate transformation model (i.e. *homography*).

I have implemented *Ratio* (section 5.3.1) and *Symmetry* (section 5.3.2) tests and presented the results in graphical form as shown in figure 5.4. The ANN matching resulted 1295 matches (figure 5.4a) , then a significant number of inaccurate matches (1118 matches) are removed by Ratio test (figure 5.4b). For remaining 177 matches, I carried out Symmetry test,

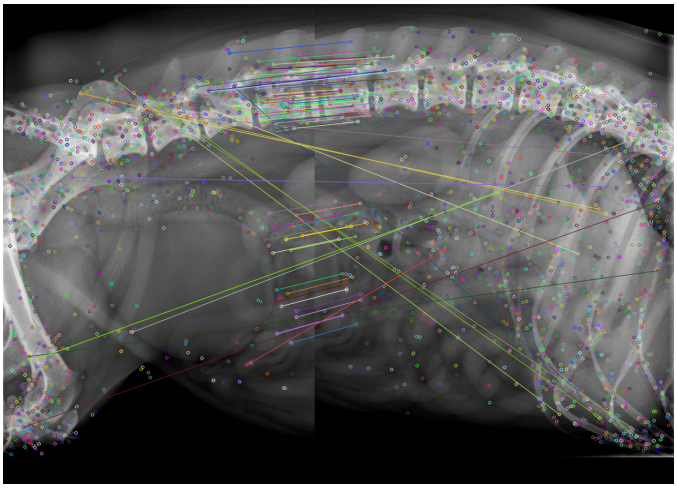
which again removed 80 inaccurate matches to get 97 best matches as shown in figure 5.4c. Although these tests removed significant number of inaccurate matches, the obtained best matches are not 100% accurate which we can be clearly seen in figure 5.4c. The remaining inaccurate matches can be identified as outliers by robust estimation methods discussed in the next chapter.



(a) Nearest neighborhood matches



(b) Ratio Test



(c) Symmetry Test

Figure 5.4: Steps of getting best matches: (a) a lot of matches found using NN method (b) Ratio test removed a significant number of false matches. (c) The false matches from ratio test are further removed by symmetry test.

Chapter 6

Homography Estimation

This chapter discusses about the estimation of transformation model using matches estimated in previous chapter. The best matches we estimated still not guaranteed to be true matches, so there may be chances of getting wrong motion parameters if we include such false matches to estimate motion model. We use homography matrix as motion model for transformation. We use robust methods to estimate homography which uses only true matches.

If there is only translation or rotation between the images to be stitched, then stitching process will be simpler; we need only affine transformation parameters. The real problems will be to register images obtained using different camera angle; we have to estimate translation, rotation and projection parameters which can be represented by estimating *homography matrix*.

A homography is a 3x3 matrix H and the elements of the matrix contains the rotation, translation and projection parameters. We can change the value of homography H without changing the projective transformation. Therefore, H can be considered as homography matrix and it has 8 degrees of freedom (although it has 9 elements) [4]. In other words, we need to solve for 8 unknowns to get the homography matrix.

6.1 Algorithm for homography Estimation

The homography matrix is solved using *Direct Linear Transform (DLT)* algorithm if we have sufficient set of point correspondences. We solve the homography matrix by using the following equation:

$$x_i' = Hx_i \tag{6.1}$$

where x_i and x_i' are 3 element vectors. For stitching problem, we use the corresponding points as vectors. In 2D problem, suppose, $\mathbf{x} = (x, y, 1)^T$ and

$\mathbf{x'} = (x', y', 1)^T$ are two corresponding points, then the relationship will be

$$c \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = H \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{6.2}$$

Hartley and Zisserman [11] suggested to use a normalization step in DLT because the DLT algorithm is dependent on the origin and scale of coordinate system of the image. The normalized DLT algorithm can be summarized by the following steps:

1. Translate the points such that the centroid lies at the origin.
2. The scaling of the points is carried out to maintain their average distance to be $\sqrt{2}$.
3. Get transformations for both the images independently and apply DLT algorithm to get homography matrix.

For more detail algorithm, it is suggested to study chapter 4 of Hartley and Zisserman book [11]

6.2 Robust Estimation

The homography estimation requires 4 corresponding points, and the matching methods described in previous sections are not robust i.e. there may be chances of some false correspondences. The two features in the images might not correspond to same the real feature. So, we need to identify the inlier and outlier correspondences and only inlier matches can be used for homography estimation. This section discusses two most popular methods for homography estimation.

6.2.1 RANSAC

RANSAC (Random Sample Consensus) was first purposed by Fischler *et al* [7] and it is the most commonly used robust estimation method for homographies [4]. This method selects 4 correspondences randomly and computes homography matrix H. Then other correspondences are classified as inliers or outliers depending on its concurrence with H. This process is repeated for a number of iterations and the iteration which gives largest number of inliers is identified. The homography H for that iteration is chosen as the best transformation model.

The classification of inliers or outliers is carried out by assigning some distance threshold t and if $\|\mathbf{x'} - H\mathbf{x}\| > t$, then the point is considered as outlier. The threshold value is problem specific. The higher the threshold

value, the larger the inliers we get. Similarly, we choose a number of iterations N so that at least one of the random samples will be free from outliers[1]. Hartley and Zisserman [11] derived a formula to compute the number of iterations required for RANSAC:

$$N = log(1-p)/log(1-(1-\epsilon)^s) \tag{6.3}$$

where p = probability at least one of the samples is free from outlier. We generally use p=0.99
ϵ=probability of outlier which can be expressed as

$$\epsilon = 1 - \frac{number\_of\_inliers}{total\_number\_of\_points} \tag{6.4}$$

s =number of correspondences used in each iteration. ($s = 4$)

Hartley and Zisserman [11] [2] claims that if we have 5% outliers(i.e. ϵ=0.05) then only 3 iterations are needed to get at least one pure sample with probability=0.99.

6.2.2 Least Median of Squares Regression

In RANSAC, we use distance threshold to identify outliers and exclude them for homography estimation. This method, as the name suggests, calculates the median of squares of the error value (i.e. difference between transformed and actual points) for each point in each iteration. The best homography is the one which gives least median value.

The Least median of Squares (LMS) was first introduced by Peter J. Rousseeuw [25] and he claims that the estimator can resist the effect of nearly 50% of contamination in the data. The ordinary least squares (OLS) estimators can not identify the non normal errors, so, LMS estimator is claimed to be a robust [25] [21] and it has the characteristics of being highly resistance to high proportion of outliers [21].

So, in summary, LMS is an effective method to detect the outliers. We do not need any initial parameters (unlike t and ϵ in RANSAC). The only disadvantage of this method is that it can not work well if there are more than half outliers.

[1] We have to test each combination of random samples, so we need to limit the number of iterations

[2] see comparison table in chapter 4 of the literature.

6.3 Experimental Results

The first part of this section presents the result of RANSAC for homography estimation by identifying outliers. While LMS method (section 6.2.2) works good when there are more than 50% inliers, it is not always guaranteed to have more than half accurate matches[3], so I prefer RANSAC method for robust estimation and it works even if there are large number of outliers. The equation 6.3 reveals that we need few iterations to get a pure sample with high probability even if there are a lot of outliers.

In the next section, I include experiment on the effect of distance threshold value (σ) to the number of inliers or outliers. The distance threshold σ has been measured in pixels.

6.3.1 Robust Estimation with RANSAC

For RANSAC, I have set the maximum possible number of iterations (N) up to 2000 and distance threshold (σ) 2 pixels. After some iterations, I got inliers (accurate matches), outliers (inaccurate matches) and the homography matrix H. Figure 6.1 shows the inlier matches after I obtained using RANSAC method. From the figure, we can easily see that the all inliers are true matches.

[3]there might be chances of being more false matches since we have used less accurate methods like SURF and ANN to get faster result

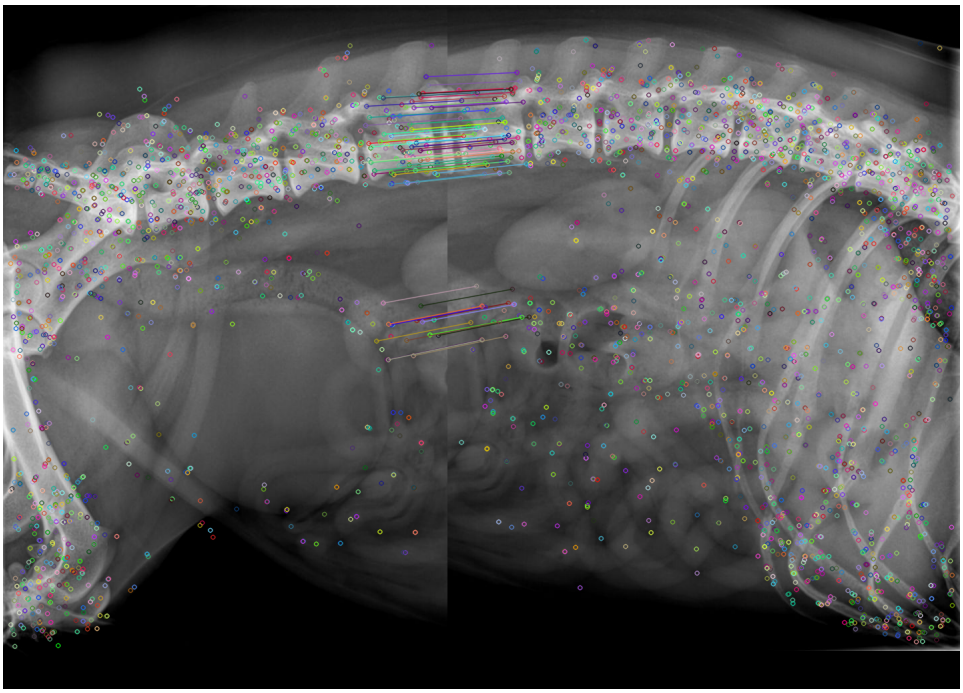


Figure 6.1: Matches after removing outliers (σ=3 pixels). This figure shows all matches are accurate.

6.3.2 Effect of Distance Threshold (σ)

The number of inliers is changed when we change the distance threshold σ. This section includes experiment on the effect of σ to the number of inliers we obtained. The graph presented in figure 6.2 shows that increase of σ, increases the number of inliers. So, we have to select proper σ so that all inliers are true matches. With small possible value of σ=1 pixel, I got 50 inliers. Since the figure 6.1 shows there are 62 accurate matches which implies that $\sigma = 1$ is missing some true matches. The increase of σ increases the inliers which also increases probability of including false matches as inliers. So, the best accepted values of σ seems to be either 1 or 2.

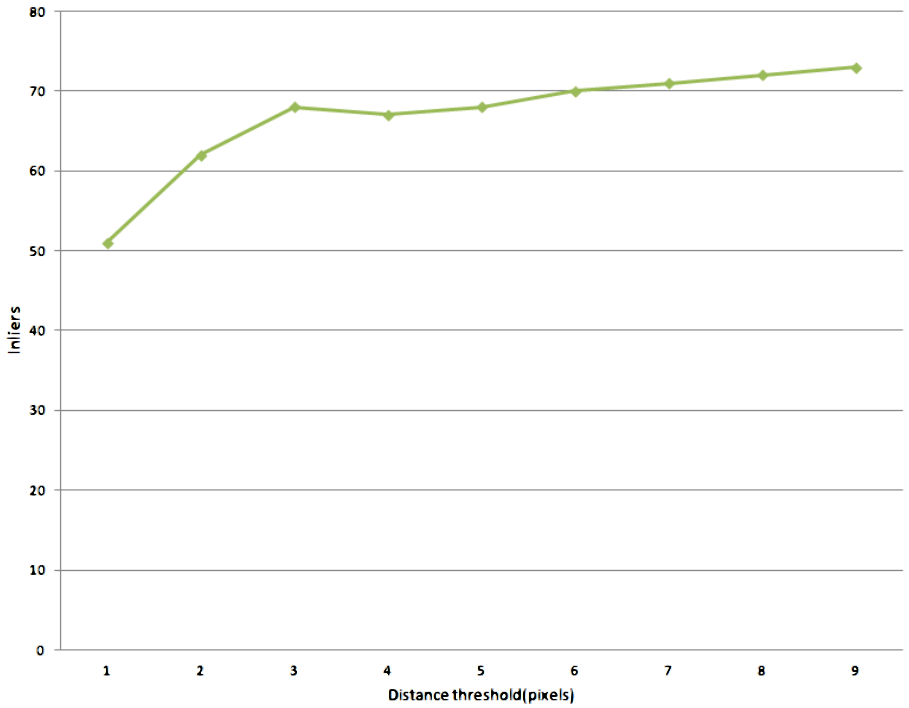


Figure 6.2: Graph showing inliers versus distance threshold

In some cases, if the stitching X-ray images are not exactly same, then key-points do not go with the transformation model (i.e. H) which results very few or sometimes insufficient inliers to guarantee the accuracy of homography. In that situation, we increase σ value to get more inliers while estimating homography.

The number of inliers [4] can also be used to identify the feasibility of stitching between two images. There might be cases when we try stitching images which do not have any common region. The inliers are counted to confirm whether the images can be stitched. We define any number greater than sample size. Most of the time, if we have more than 15 inliers, we accept the homography to create composite image. For very few inliers[5], there is chances of getting inaccurate homography and we stop the stitching process.

[4]The inliers in this context is the best inliers we got after a number of iterations.
[5]Few inliers imply limited or no overlapping between input images.

Chapter 7

Compositing

This chapter focuses on the compositing techniques which include *transformation* and *blending* to get the final stitched image. We use two or more input images for stitching and we do not change the co-ordinates of *reference image* and all other images (i.e. *floating images*) are transformed into the reference image co-ordinates using *homography*. Section 7.1 discusses about *transformation* which overlaps the common part of images. The resulting *composite image* might contain visible seams (due to exposure differences), blur (due to mis-registration) and ghosting (due to moving objects). The quality of the stitched image is defined by the similarity of the stitched image to the input images and the visibility of seam between the images [14].

7.1 Transformation

In this section, the transformation of images into the the co-ordinates of the reference image is carried out and we estimate the composite size and overlapped regions of the the images. The medical images are always flat without any radial distortion, homography can be used to transform the co-ordinates [30].

Estimation of Composite Size The estimation of the composite size is carried out by transforming the corners of the floating images. If we already know the direction of stitching (see section 4.1), then it is easy to estimate the size. We have to develop a general algorithm which works for any alignments. Since, most of the X-ray stitching problems, we already know the alignment (either vertical or horizontal) between images. So, we can modify the algorithm that uses direction information for faster estimation. A simple example on how to compute the composite size has been presented below:

Suppose, the transformed corners of the floating image are $\{(x_{f1}, y_{f1}), (x_{f2}, y_{f2}), (x_{f3}, y_{f3}), (x_{f4}, y_{f4})\}$ and the corners of the reference image

are $\{(x_{r1}, y_{r1}), (x_{r2}, y_{r2}), (x_{r3}, y_{r3}), (x_{r4}, y_{r4})\}$. Then, the corners of the composite image are calculated as follows:

1. Calculate the minimum and maximum of x and y values of corners of transformed float image and reference image i.e.

$$\begin{aligned} x_{min} &= min(x_{f1}, x_{f2}, x_{f3}, x_{f4}, x_{r1}, x_{r2}, x_{r3}, x_{r4}) \\ y_{min} &= min(y_{f1}, y_{f2}, y_{f3}, y_{f4}, y_{r1}, y_{r2}, y_{r3}, y_{r4}) \\ x_{max} &= max(x_{f1}, x_{f2}, x_{f3}, x_{f4}, x_{r1}, x_{r2}, x_{r3}, x_{r4}) \\ y_{max} &= max(y_{f1}, y_{f2}, y_{f3}, y_{f4}, y_{r1}, y_{r2}, y_{r3}, y_{r4}) \end{aligned} \tag{7.1}$$

2. Now the corners of the composite image are (x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max}). Obviously, the width and height of the composite image will be $(x_{max}\text{-}x_{min})$ and $(y_{max}\text{-}y_{min})$ respectively. Unless the two images are exactly same, the composite image size is always greater the either image.

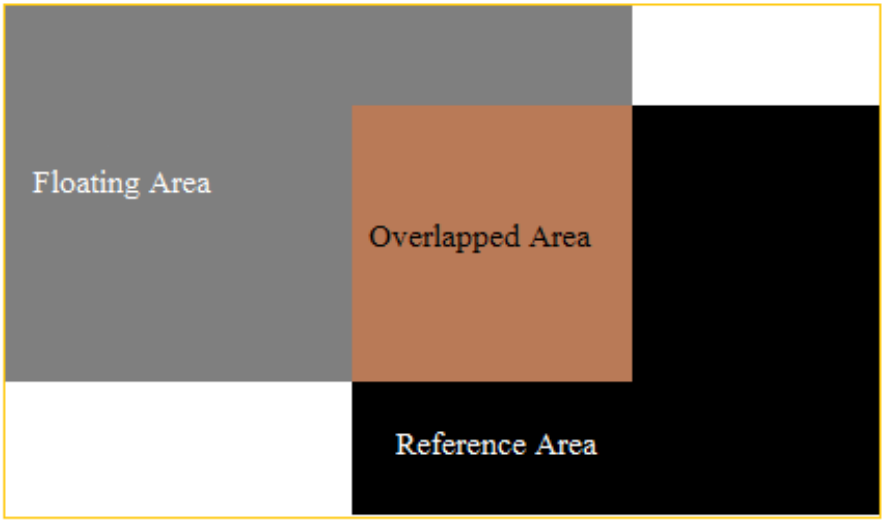


Figure 7.1: Compositing of two images. The area within golden line is the compositing area. The gray area is floating image while black area is reference image.The overlapped area is painted with brown color.

Overlapping area identification After we calculate the composite area i.e. the size of the stitched image, next task is to create an image with composite size and assign the pixel values of the float and reference images. Till now everything is okay except the overlapping areas. We assigned the overlapping pixels two times because those part is

common to both floating and reference images (see figure 7.1). Since most of the real time problems, the overlapping areas are not same due to exposure differences and illumination which results visible seams in composite images. Again, we get blurred or ghosts in the overlapping region because of not accurate registration. To remedy these problem, we implement blending techniques. Some popular blending techniques are discussed in section 7.2. Sometimes, if the intensity difference between images is large, the blending techniques are not capable of completely removing the visible seams [4], we have to implement exposure compensation technique discussed in section 7.3.

7.2 Blending

The overlapping regions are blended for exposure compensation and misalignments. There are blending techniques which remove discontinuities in the composite image and create visually appealing stitched image. In this section, I am going to discuss some popular blending techniques: *Optimal Seam Blending*, *Alpha Blending* and *Pyramid Blending*.

7.2.1 Optimal Seam Blending

Optimal seam blending method search for a curve in the overlap region on which the difference of the images are minimal. If I_1 and I_2 are overlapping regions, for any y=y_1, a point (x,y_1) lies on the optimal seam curve if I_1(x,y_1) - I_2(x,y_1) is minimum for all x. Then, each image is copied to corresponding side of the seam. This simple method, however, does not give good result when there is global intensity difference between the images I_1 and I_2 [14].

7.2.2 Alpha Blending

Alpha blending, also called *feathering*, is simple but effective algorithm. Alpha blending assigns the weight values (i.e. α) to the pixels of the overlapping area. For α=0.5, we get simple averaging, where both the overlapped areas will contribute equally to create stitched image. The value of α ranges from 0 to 1; if α=0, then the pixel has no effect in composite region while α=1 implies the pixel is copied there. Suppose, composite image I is created from horizontally aligned images I_1 (left) and I_2 (right), then

$$I = \alpha I_1 + (1 - \alpha) I_2 \tag{7.2}$$

Initially, we start with α=1 (i.e. fully opaque) from I_1 until we reach overlap region. We go on decreasing α until it reaches to 0 (i.e. fully transparent) at the end of overlap region (see figure 7.2).

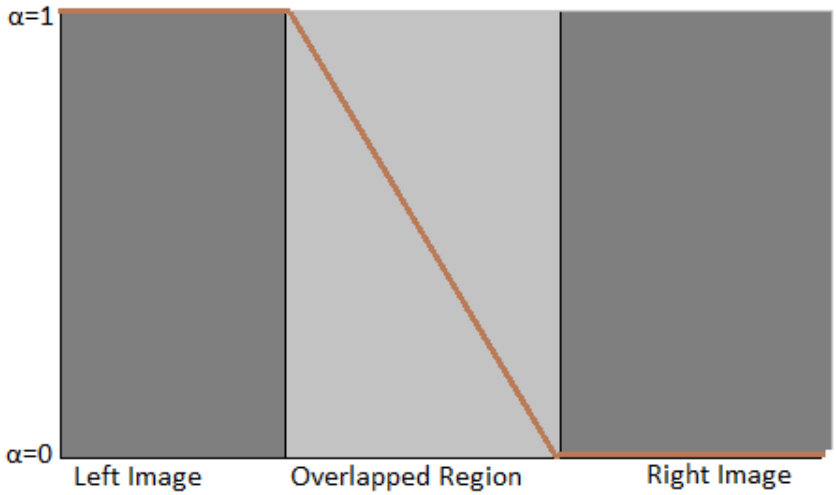


Figure 7.2: Alpha blending: α decreases from 1 to 0 in the overlapping region.

The above method is for horizontally aligned images; and similar technique can be used for vertically aligned images. If alignment is both horizontal and vertical, then left, right, top and bottom regions of the blending region will have effect in blending.

Alpha blending technique works well if the intensities of images I_1 and I_2 are similar. The advantage of alpha blending is its simplicity and we can tweak it to make it faster e.g. *Look Up Table* [23].

7.2.3 Pyramid Blending

The *pyramid blending* uses image pyramids to blend and we use blending mask to mask the blended area. The blending mask assigns the weights of the pixels of the compositing images. The method consists of the following steps [30]:

- Create Laplacian pyramids L_1 and L_2 from images I_1 and I_2.
- Create a Gaussian pyramid G_M of the blending mask M.
- Construct a combined pyramid $L_{combined}$ from L_1, L_2 and G_M as:

$$L_{combined}(i,j) = G_M(i,j) * L_1(i,j) + (1 - G_M(i,j)) * L_2(i,j) \quad (7.3)$$

- Construct the blended image by collapsing the $L_{combined}$ pyramid.

The blending mask consists of the weights of the pixels in the compositing images I_1 and I_2. The values generally vary from 0 to 1 in the overlapping areas whereas either 0 or 1 in the non-overlapping parts. To make the process faster, the only overlapped areas are chosen for blending and other pixel are simply copied to the composite image.

7.3 Exposure Compensation

The *alpha blending* and *pyramid blending* methods give a good blending result, compensate for moderate amounts of exposure difference between images. The methods, however, fail to give pleasing blending result when exposure difference become large [30]. This problem is more dominant if there is some rotation between the images while registration.[1]

The transfer function based approach defined by Uyttendaele *et al* [32] seems to be effective to remove the exposure related artifacts. This method fits the block-based quadratic transfer function between each source image and an initial composite image. Then, the averaging of the transfer functions is carried out for smoother result. Per pixel transfer functions are calculated by interpolating between neighboring block values. This method does a better job of exposure compensation than simple feathering [32].

The above method can be simplified and faster by estimating the transfer function between the overlapped areas (i.e. $I_1 \Rightarrow I_2$). Considering each and every pixels in the image makes the algorithm slower, so we only take care of true matching points. The transfer function parameters are estimated using a number of matched pairs[2]. The estimated transfer function maps the intensity of I_1 to intensity of I_2; thus works as an *intensity leveling function*.

[1] the rotated image will create extra pixels around the image which increases the complexity of the blending task.

[2] number depends upon the transfer function parameters e.g. linear transfer function $y = ax + b$, there are two unknown parameters to estimate, so we need two matched pairs

7.4 Experimental Results

This section presents some analysis and evaluation of blending algorithms with experimental data. The good blending algorithm should give visually appealing composite image with less change of image information and it should be computationally inexpensive. In the first part of experiment will select the best method among *alpha blending* and *pyramid blending*; and second part of experiment will focus on the tweaking of the selected algorithm to get the best optimized blended result for all alignments.

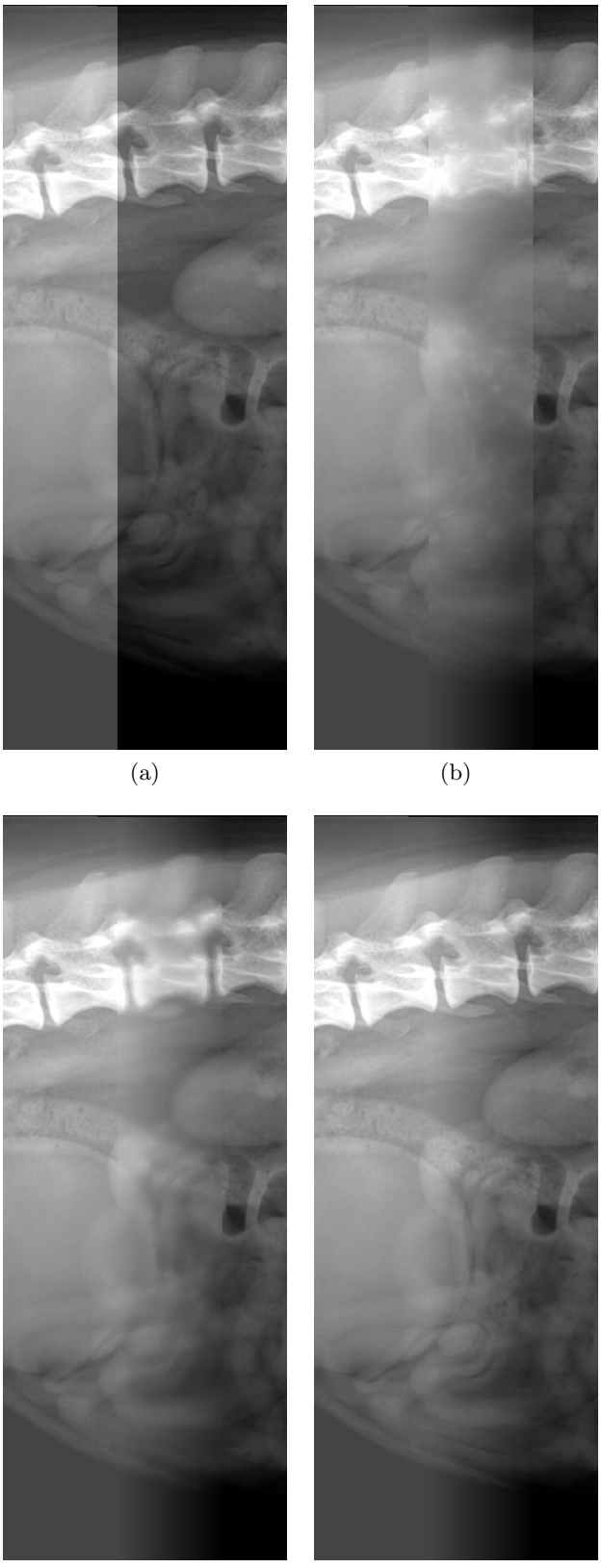
7.4.1 Alpha Blending versus Pyramid Blending

In this section, I will present performance measures in terms of computational time and accuracy. The algorithms are implemented for both similar intensity and different intensity images. I have measured the time to produce the blended result for both the algorithms. The accuracy measurement is carried out in terms of error in pixel intensities from the corresponding actual pixels.

Blending Method	Similar Intensity		Different Intensity	
	Computational Time (seconds)	Error	Computational Time (seconds)	Error
Pyramid Blending (Level2)	0.105	1278.8044	0.106	18750.8828
Pyramid Blending (Level 4)	0.119	4097.8959	0.140	21644.5312
Alpha Blending	0.015	260.5863	0.015	17564.5039

Table 7.1: Alpha Blending versus Pyramid Blending

From the table 7.1, pyramid blending(level 2 and level 4) is taking longer time as compared to alpha blending. The error measurement shows that pyramid blending is losing more image information than alpha blending (see figure 7.3). The blended X-ray image should preserve image information and alpha blending is showing a better result than pyramid blending.



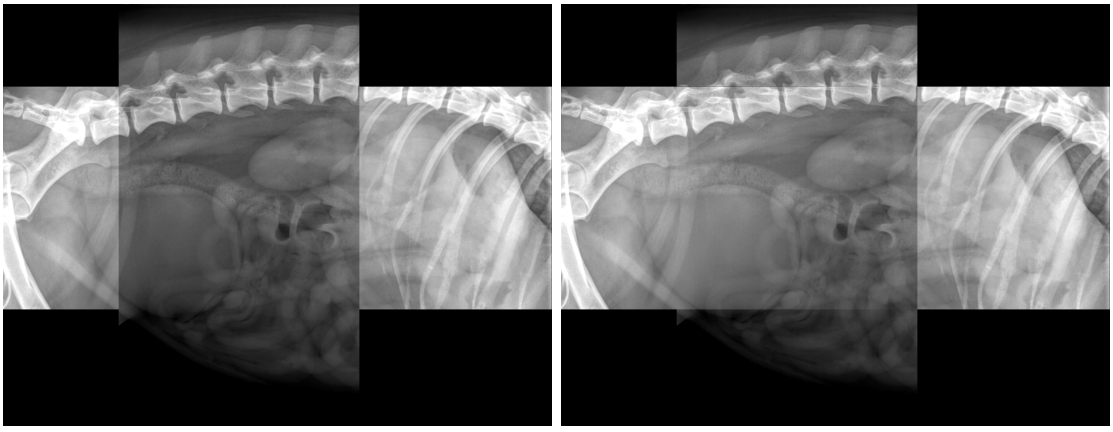
(a) (b)

(c) (d)

Figure 7.3: Comparison of results of Pyramid Blending and Alpha Blending. (a) is the composite image, (b) is level 4 pyramid blended, (c) is level 2 pyramid blended and (d) is alpha blended image. The blurring effect in Pyramid Blending is undesirable because it loses image information.

7.4.2 Blending Masks

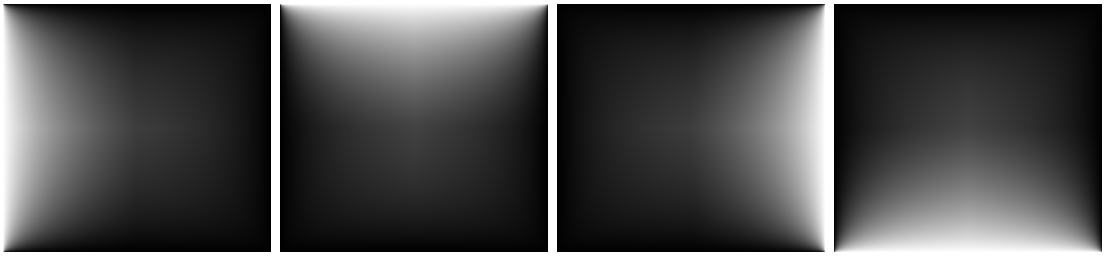
In the above section, I found alpha blending is faster and gives more accurate result as compared to pyramid blending. In this section, I will carry out experiments on the images with complex alignments. The horizontal or vertical blending for the image shown in figure 7.4a produces visible seams shown in figure 7.4b.



(a) (b)

Figure 7.4: Figure (a) is raw composite image, figure (b) is horizontally blended image. The seams are clearly visible on the join of the images.

So, to remedy this problem, a *modified alpha blending method* has been implemented in which we create four blending masks (*left*, *top*, *right*, *bottom*) as shown in figure 7.5. The values in the mask indicates the effect of the image in the corresponding blending pixel (i.e. the higher value, the higher effect). For example, the left blending mask has higher values (shown white in the image) in left side means left image will have higher effect in the left side of the blended region to remove discontinuities in the composite image.



(a) Left (b) Top (c) Right (d) Bottom

Figure 7.5: Blending masks. (a) is mask for left image,(b),(c), (d) are masks for top, right and bottom images respectively.

The blended image using multiple blending masks is shown in figure 7.6 which is far better than the image produced with either horizontal or vertical blending (compare images: 7.4b and 7.6)

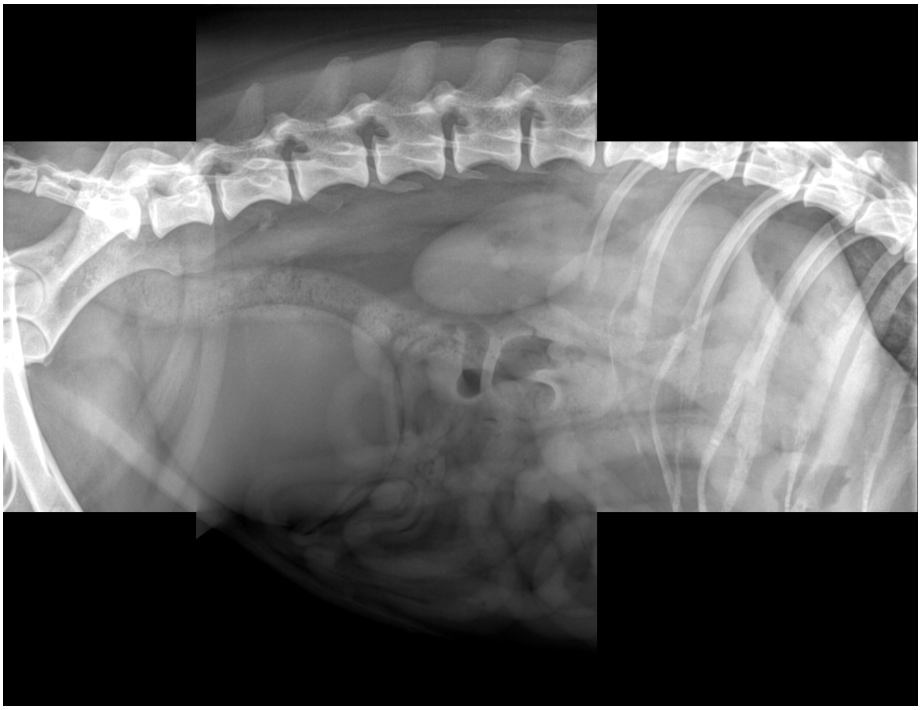


Figure 7.6: Use of multiple blending masks gives better blending result.

Chapter 8

Conclusion

This thesis evaluates image analysis methods that can be used for stitching of medical X-ray images. Generally, image stitching process includes several steps. If we do not improve or optimize the methods, the process takes a lot of time that makes it inappropriate for real-time or user interactive applications. I have analyzed, evaluated and optimized the methods based on computational complexity and accuracy to get faster stitching result.

Image alignment using feature points are faster and always guaranteed to give best result while other methods such as pixel based alignment, area or pattern based alignment are slower and not guaranteed to provide best result. I have analyzed and evaluated three most popular feature extractors: *Harris*, *SIFT* and *SURF*. Harris is good for faster detection of corners in the image, but it can not give accurate corners if the images to be stitched have different intensity or orientation. So, for feature extractors, either SIFT or SURF methods can be employed. For feature matching, although, it gives accurate result, SIFT is suffering from its high computational complexity, so, SURF is better option because it is faster than SIFT. The exhaustive *nearest neighborhood* (kNN) method is modified to *approximate nearest neighborhood* (ANN) to get faster matching while sacrificing minor matching inaccuracies. Since, nearest neighborhood based methods gives just nearest key-points as a match which is not always a true match. The application of accuracy tests such as *Ratio Test* and *Symmetry Test* are quite effective to remove those false matches. The use of SURF features matching and approximate nearest neighborhood methods are intended to make the stitching process faster sacrificing some minor inaccuracy in matching resulting from those methods.

The transformation model is used to create the composite image. The homography matrix is suitable transformation model for medical images. The *RANSAC* method is preferable for robust estimation of homography

because it works quite effectively even if we have a lot of false matches.

I have evaluated two popular blending techniques: *alpha blending* and *pyramid blending*. Although pyramid blending is an effective method for blending, it is computationally expensive and loses image information. So, alpha blending is chosen for blending which is faster and gives more accurate result. The alpha blending can be modified to work on complex alignments which occurs if the images are rotated and/or are taken in perspective view. The modified alpha blending use blending masks to assign the weights of the pixels of contributing images on the blended region. The method is effective because it successfully removes the seams and discontinuities on the composite image.

Chapter 9

Limitations & Future Work

The image alignment methods implemented in the current version of stitching are giving very good result for different intensity, scaling or rotated images. Although the methods are optimized to get faster and better result, still for very high resolution images, the process is bulky. So, an extension of this project could be to carry out research on obtaining more optimized algorithms to work for very high resolution images. An X-ray image generally consists of a lot of background region which consists of very little or no information for image stitching. So, we can implement some technique that selects bones and muscles and discard the other areas in the image.

There still needs to work more on the blending part. For different intensity images, if the images are rotated, the current blending creates a seam on the overlapped region. The problem is because of unassigned pixels created when image is rotated. The current version employs the filling of unassigned pixels with pixels of other image which works perfect for images with same intensity. Because of intensity difference, the unassigned pixels get intensity which is different from their actual pixels and we see visible line after blending. The extension could be to use some transformation method to get the accurate pixel intensity to be filled in the unassigned pixels.

Bibliography

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Matthew Brown and David Lowe. Invariant features from interest point groups. In *In British Machine Vision Conference*, pages 656–665, 2002.

[3] David L. Donoho and Ery Arias-Castro. Does median filtering truly preserve edges better than linear filtering? 2006.

[4] Elan Dubrofsky. Homography estimation. Master's thesis, The University of British Columbia, 2009.

[5] Joe Durnavich. Making sense of the head x-rays. http://mcadams.posc.mu.edu/xray/reading/reading.htm, apr 2010. Accessed: 20/07/2012.

[6] Christopher Evans. Notes on the opensurf library. 117 SCTR-09-001, University of Bristol, January 2009. http://www.cs.bris.ac.uk/Publications/Papers/200970.pdf.

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[8] Vincent Garcia, Eric Debreuve, Frank Nielsen, and Michel Barlaud. K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching. In *Image Processnig(ICIP),2010 17th IEEE International Conference on*, pages 3757–3760, sep 2010.

[9] D M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1 - Volume 1*, ICPR '98, pages 439–, Washington, DC, USA, 1998. IEEE Computer Society.

[10] D.M. Gavrila and V. Philomin. Real-time object detection for ldquo;smart rdquo; vehicles. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 87 –93 vol.1, 1999.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] Abhinav Kumar, Raja Sekhar Bandaru, B Madhusudan Rao, Saket Kulkarni, and Nilesh Ghatpande. Automatic image alignment and stitching of medical images with seam blending. *World Academy of Science, Engineering and Technology*, 2010.

[13] Robert Laganire. *OpenCV 2 Computer Vision Application Programming Cookbook*. 2011 Packt Publishing, Packt Publishing Ltd.,Birmingham, UK, May 2011. http://www.packtpub.com.

[14] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In *In Proceedings of the European Conference on Computer Vision*, 2006.

[15] Shuo Li. Registration of 3D volumetric CT images. Master's thesis, Uppsala University, Department of Information Technology, November 2011. IT series, 11 080.

[16] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, CVPR '96, pages 465–, Washington, DC, USA, 1996. IEEE Computer Society.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[18] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[19] Xue Mei and Fatih Porikli. Fast image registration via joint gradient maximization: application to multi-modal data. In *Proceedings of SPIE Volume 6395 Electro-Optical and Infrared Systems: Technology and Applications III*, September 2006.

[20] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference*

on Computer Vision Theory and Application VISSAPP'09), pages 331–340. INSTICC Press, 2009.

[21] Özlem Önder. Least median squares: A robust regression techique. *Ege Academic Review*, 1(1):185–191, 2001.

[22] Donovan Parks and Jean-Philippe Gravel. Corner detection. Technical report, Faculty of Engineering, McGill University, Natural Science and Engineering Research Council of Canada, 2011. http://kiwi.cs.dal.ca/ dparks/CornerDetection/index.htm.

[23] Vladan Rankov, Rosalind J. Locke, Richard J. Edens, Paul R. Barber, and Borivoj Vojnovic. An algorithm for image stitching and blending. In *Proceedings of SPIE*, pages 190–199, 2005.

[24] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

[25] Peter J. Rousseeuw. Least median of squares regression. *The American Statistical Association*, 79(388), December 1984. http://web.ipac.caltech.edu.

[26] Harpreet S. Sawhney and Rakesh Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(3):235–243, March 1999.

[27] Utkarsha Sinha. Sift:Scale Invariant Feature Transform. http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/7/, May 2010. Accessed: 17/09/2012.

[28] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* Library of Congress Control, 3rd edition, 2008.

[29] Perry Sprawls. X-ray Image Formation and Contrast. http://www.sprawls.org/ppmi2/XRAYCON/. Accessed: 17/09/2012.

[30] Richard Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, January 2006.

[31] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 251–258, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[32] Matthew Uyttendaele, Ashley Eden, and Richard Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *CVPR (2)*, pages 509–516. IEEE Computer Society, 2001.

[33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 1:I–511–I–518 vol.1, 2001.

[34] Greg Ward. Hiding seams in high dynamic range panoramas. In Roland W. Fleming and Sunghee Kim, editors, *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization, APGV 2006, Boston, Massachusetts, USA, July 28-29, 2006*, volume 153 of *ACM International Conference Proceeding Series*, page 150. ACM, 2006.

[35] Wikipedia. Google street view. http://en.wikipedia.org/wiki/Google_Street_View, September 2012. Accessed: 16/09/2012.

[36] Harald Woeste. *Mastering Digital Panoramic Photography.* Rocky Nook, 1st edition, November 2009.

[37] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.