

Stitching of X-ray Images

Krishna Paudel

October 17, 2012

Abstract

This thesis report discusses how two or more X-ray images can be joined together to form a single image. The stitching process includes registration, transformation and blending steps. The basics of registration starts from pixel wise matching(like exhaustive matching, Chamfer matching etc.). To make those methods faster and more accurate, I have chosen feature based matching algorithms where a number of feature points are extracted and matched.

Some of the popular feature extraction methods are *Harris Points Detector*, *SIFT*, *SURF* etc. The improved Random Sample Consensus(RANSAC) method has been used to get the best matched points which gives the transformation parameters(Homography Matrix, Fundamental Matrix etc.). The final phase of image stitching is to remove the seams and discontinuities on the joined part of the stitched image. Some blending methods like alpha blending, pyramid blending etc. have been analyzed and implemented.

The selection criteria of the above mentioned algorithms are speed, accuracy, intensity invariance, scale invariance, rotation invariance. The analysis part is mainly focused on getting the best feature extraction, registration and blending methods. Finally, those selected methods have been implemented to develop a good image stitching application which can be integrated in any medical software.

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 Rutsch* 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 and Michael Mller.

I am thankful to the *Center for Image Analysis* for accepting my thesis proposal and my examiner *Cris Luengo* for his invaluable suggestions and providing his invaluable time to read my report. Many many thanks to my friends who helped me directly or indirectly to successfully complete this report. Special thanks to Mesfin Mamuye, Sujan Kishor Nath.

Contents

1	Introduction	13
1.1	X-Ray Imaging	14
1.2	Pixel Based Alignment	16
1.3	Feature Based Alignment	17
1.4	Image Stitching Process	17
1.5	Challenges of Image Stitching	18
2	Related Work	19
3	Background Theory	23
3.1	Data Structures for Image Analysis	23
3.1.1	Matrices	23
3.1.2	Pyramids	23
3.2	Brightness Transformation	25
3.3	Geometric Transformation	26
3.4	Image Smoothing	27
3.5	Edge Detection	28
3.5.1	Laplacian of Gaussian	28
3.5.2	Approximation with Difference of Gaussian	29
3.6	Error Metrics	29
3.7	Corners in Image	31
3.7.1	Requirements of a Corner Detector	31
3.7.2	Corner Detection	32
4	Feature Extraction	33
4.1	Preprocessing	33
4.2	Feature Extraction	34
4.2.1	Harris Corner Detection	35
4.2.2	SIFT	36
4.2.3	SURF	39
4.3	Experimental Results	45
4.3.1	Experiment 1: Computational Time	45
4.3.2	Experiment 2: Stability	46

5 Features Matching	49
5.1 kNN Matching	49
5.2 ANN Matching	49
5.3 Removing False Matches	50
5.3.1 Ratio Test	50
5.3.2 Symmetry Test	51
5.4 Experimental Results	52
5.4.1 SIFT verses SURF	52
5.4.2 kNN verses ANN	53
5.4.3 Getting Accurate Matches	53
6 Homography Estimation	57
6.1 Algorithm for homography Estimation	57
6.2 Robust Estimation	58
6.2.1 RANSAC	58
6.2.2 Least Median of Squares Regression	59
6.3 Experimental Results	60
6.3.1 Robust Estimation with RANSAC	60
6.3.2 Effect of Distance Threshold(σ)	61
7 Compositing	63
7.1 Transformation	63
7.2 Blending	65
7.2.1 Optimal Seam Blending	65
7.2.2 Alpha Blending	65
7.2.3 Laplacian Pyramid Blending	66
7.3 Exposure Compensation	67
7.4 Experimental Results	68
7.4.1 Alpha Blending verses Pyramid Blending	68
7.4.2 Blending Masks	70
8 Conclusion	73
9 Limitations & Future Work	75

List of Tables

4.1	Computational time of feature detectors	45
4.2	Feature Points Count	47
4.3	Stability measurement	47
7.1	Alpha Blending verses Pyramid Blending	68

List of Figures

1.1	Production of X-ray	15
1.2	X-ray Imaging System	16
2.1	Matching Using a DT	20
3.1	Image in matrix form	24
3.2	Image Pyramids.	24
3.3	Histogram Transformation	25
3.4	Comparison of DoG and LoG	29
3.5	Flowchart for Corner Detectors	32
4.1	Overlapping Area Prediction	34
4.2	Construction of DoG Image	37
4.3	Maxima and Minima Idenfication of DoG Images	38
4.4	Creation of Key-point Descriptor	39
4.5	Calculation of sum of intensities	40
4.7	Scale space generation	41
4.6	Approximation of Gaussian partial derivative	41
4.8	Scaling in Octaves	42
4.9	Haar Wavelets	43
4.10	Orientation Assignment	43
4.11	Descriptor Components	44
4.12	Comparison: Computational Time	46
4.13	Comparison: Repeatability of corners	47
5.1	kNN Search	50
5.2	Comparison of SIFT and SURF	52
5.3	Comparison of kNN and ANN	53
5.4	Steps of getting best matches	55
6.1	Matches After Removing Outliers	61
6.2	Distance Threshold and Inliers Count	62
7.1	Compositing	64
7.2	Alpha Blending	66

7.3	Alpha Blending verses Pyramid Blending	69
7.4	Complex Alignment	70
7.5	Blending Masks	71
7.6	Blending with Masks	71

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” helps medical professionals by providing a single, high resolution image of many x-ray images of a body part. Stitching of medical images is similar to creation of panorama of a scene using several images of a scene. Google also has implemented image stitching technology to display the street view of a city [Wik12].

This report presents the stitching of 2D medical images; the methods and algorithms can be extended to work for 3D images too. The x-ray machines generally are not large enough to cover all the area of some body parts like legs, spines, hands etc. To solve this problem, we capture multiple images of the body part. Then image stitching process creates a single high resolution image representing full body part. The single image of the body part makes easy to the physicians to diagnose for 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 [Sze06]. Creating high resolution images by combining smaller images are popular since the beginning of the photography [KBR⁺10].

There should be nearly exact overlaps between images for stitching and

identical exposures to produce seamless results [War06]. The stitching is not possible if there is not any common region between images. The images of same scene will be of different intensities, different 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 the error criteria to get the best result i.e. the best registration is the one which gives least error. Those methods are pretty much 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[Sze06]. 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 image, describes how X-ray images are produced and their characteristics. The next sections discuss about the various types of image alignment methods and the overall stitching process. The challenges of the image stitching have been discussed in the final section.

1.1 X-Ray Imaging

The X-ray technology was developed by *Wilhelm Roentgen*, a German physicist in 1895. 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

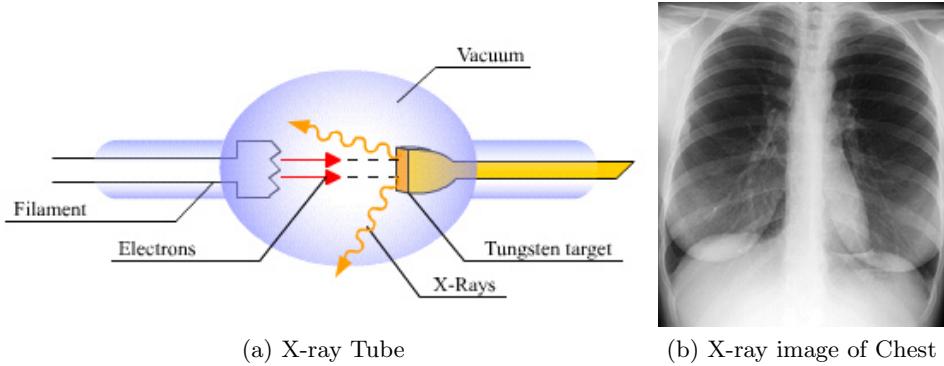


Figure 1.1: Production of X-ray: X-ray tubes are used to produce X-ray images. Figure (b) is a sample X-ray image

X-ray generator and *Flat Panel Receptor*. The body is placed in between the X-ray source and receptor so that it can penetrate the body part. The X-rays which passes 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 [Dur10].

- *Scaled and Oriented X-ray Images* Since the X-ray tube is movable, so 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. 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.

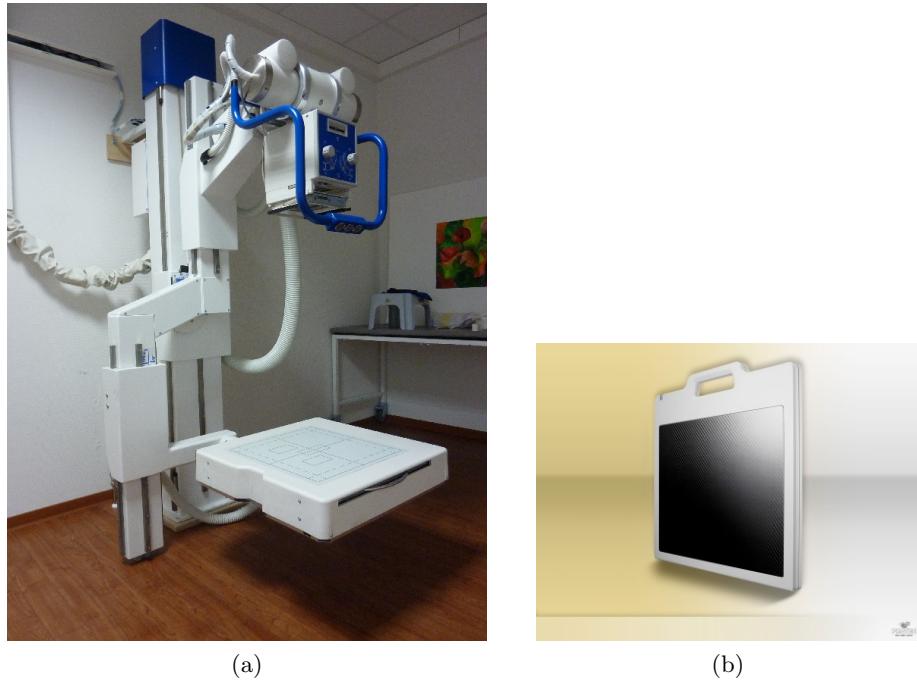


Figure 1.2: X-ray Imaging System. (a): X-ray Machine (b):Flat Panel ¹
Image Source: <http://www.protec-med.com>

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

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 alignments 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 [Sze06].

1.3 Feature Based Alignment

The pixel based alignment method is not suitable for large size images requiring real time result. So, this problem of pixel based alignment is overcome by *feature based alignment* methods. 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 [Sze06]. 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 images² 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 result incorrect stitching result. So, 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) will be described in chapter 6).

²the input images are already preprocessed like 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 area lying in the same position.

Blending This is final step of image stitching. If the overlapping areas are not exact³ which results visible lines(seams) in the joined image. So, we use blending techniques to remove those discontinuities. The blending techniques have been discussed later.

1.5 Challenges of Image Stitching

Image stitching is not a simple task; it consists of several challenges we have to focus on to get the best 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 be 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.

³in real stitching problems, the overlapping areas are not exact most of the time.

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 [Woe09].

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 $M \times M$ and I_2 of size $N \times N$, 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 [LK81]. The registration method purposed by Lucas and Kanade [LK81] became widely popular at that time because it drastically reduced the computational time to $O(M^2\log N)$. The matching process was an iterative *Newton Raphson method* where in each iteration we go on getting better match.

Similarly, Xue Mei and Fatih Porikli [MP06] 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 tem-

¹matching for rotated images, complexity increases.

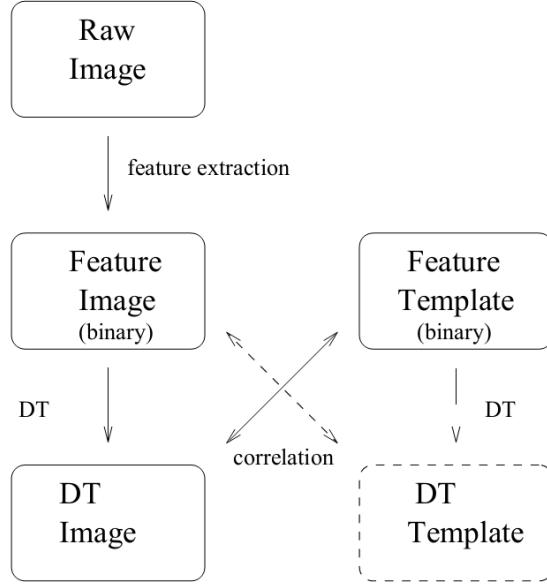


Figure 2.1: Matching using a distance transform. Image source [Gav98]

plates to match rotated images [Gav98]. The authors in the paper [GP99] 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 describes the global image alignment methods. The computation of globally consistent alignments has been discussed in the papers [SS97] and the variations of exposure has been addressed in [SK99].

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 [Sze06]. Brown and Lowe in their paper [BL02] 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 corner detector has been discussed in [PG11]. Similarly, the very fast corner detector(Features from Accelerated Segment Test) have been purposed in [RD06]. The more robust feature points extractors(SIFT and SURF) has been discussed in papers [Low04], [?]. 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. We basically focus on 2D gray scale image processing and its not a big deal to extend the methods for 2D to 3D images.

3.1 Data Structures for Image Analysis

An image consists of information and that should be represented in the form of data structure. This sections discusses two main data structures used in image analysis that is applicable in 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 contains multiple matrices to represent each spectrum(for e.g. the RGB color images are represented by 3 matrix containing red, green and blue values). All matrix related operations(like addition, subtraction, multiplication, scaling, inverse etc.) can be applied to the image 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 the processing faster, we process the image in lower resolution to find out the interesting part of image and image processing is carried out only those parts in higher resolution [SHB08]. This is achieved by generating matrix pyramids of an image which consists

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

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

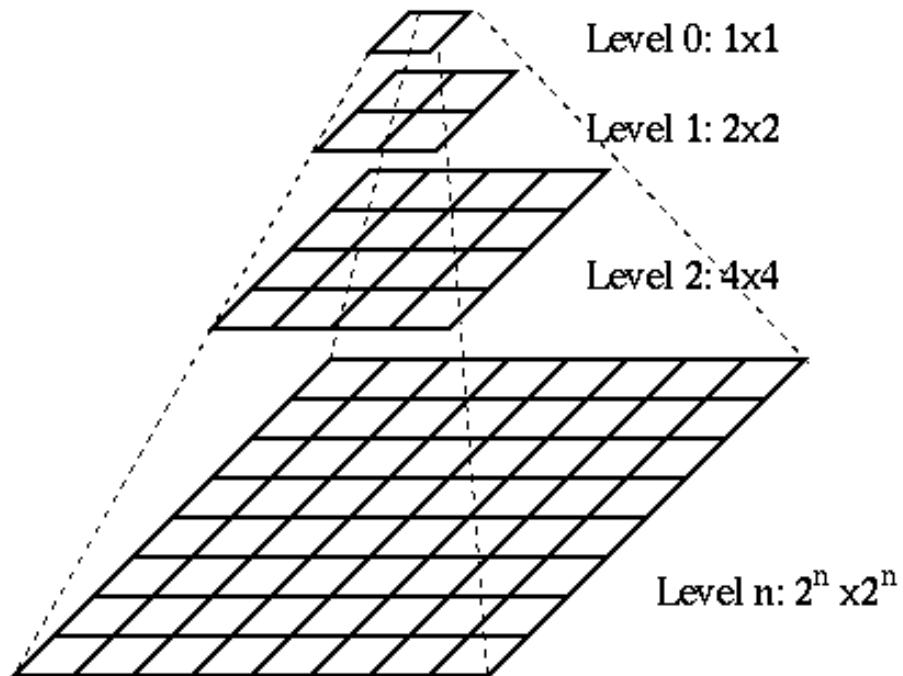


Figure 3.2: Image Pyramids.(Source: <http://fourier.eng.hmc.edu>)

of a sequence of images $\{M_L, M_{L-1}, \dots, M_0\}$ (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. Most of the times, we do not need to calculate all the pyramid images, how many pyramids to generate depends upon the problem.

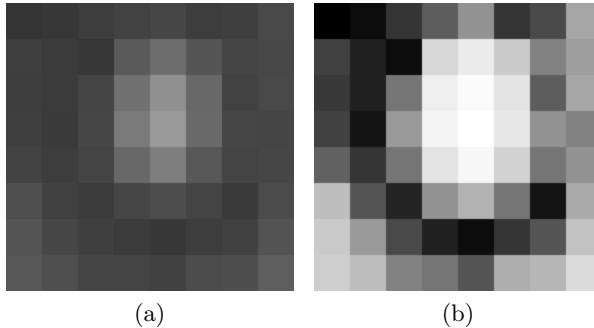


Figure 3.3: Histogram Transformation: (a) is original image; (b) is histogram transformed image.

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 pixelwise 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 for contrast enhancement 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, see [SHB08].

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

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

out using the table. For e.g. 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 \mathbf{T} which maps the pixel (x,y) to a new position (x',y') defined by the following two component equations:

$$x' = T_x(x, y), y' = T_y(x, y) \quad (3.1)$$

The transformation vector function \mathbf{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.

$$\begin{aligned} x' &= a_0 + a_1x + a_2y, \\ y' &= b_0 + b_1x + b_2y \end{aligned} \quad (3.2)$$

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

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

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

The perspective transforms preserve straight lines and are appropriate for 3D scenes observed under pure camera rotation or planes observed under general 3D motion [Sze06]. 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.[SHB08]. 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 select *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, then 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 [DAC09]. 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 large size 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).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (3.4)$$

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

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

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 [ZT98] [Lin96]. 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

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

where

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

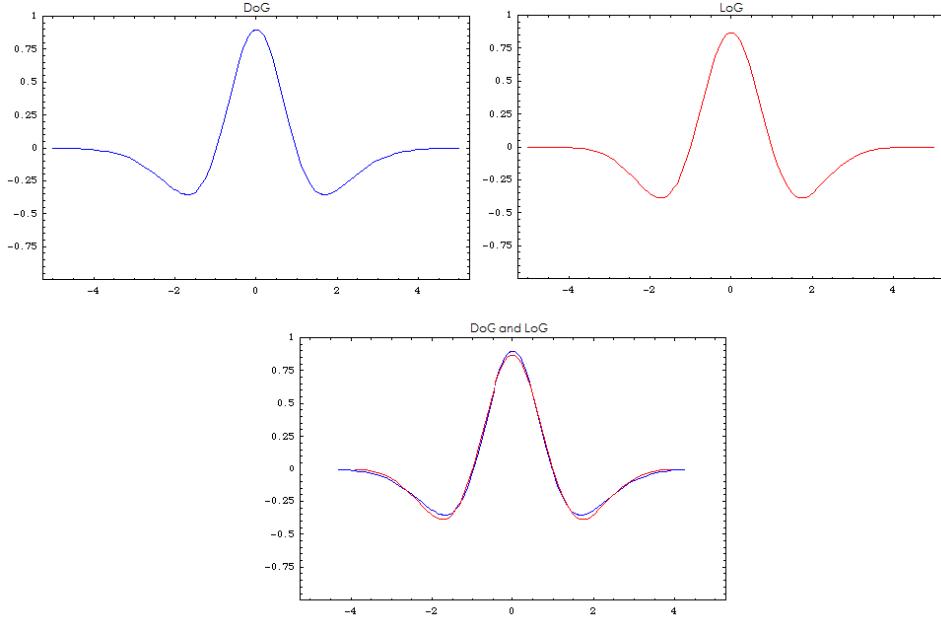


Figure 3.4: Comparison of DoG and LoG. Image Source: url-http://en.wikipedia.org/wiki/Difference_of_Gaussian

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 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

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

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 metric to compare the images [Sze06] 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:

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

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,

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

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 [Sze06]. So, *normalized cross-correlation* is commonly used,

$$E_{NCC} = \frac{\sum_i [I_0(x_i) - \bar{I}_0][I_1(x_i) - \bar{I}_1]}{\sqrt{\sum_i [I_0(x_i) - \bar{I}_0]^2[I_1(x_i) - \bar{I}_1]^2}} \quad (3.11)$$

where

$$\bar{I}_0 = \frac{1}{N} \sum_i I_0(x_i) \quad (3.12)$$

$$\bar{I}_1 = \frac{1}{N} \sum_i I_1(x_i) \quad (3.13)$$

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.

Matching Error

Matching error based techniques can be used in feature based registration methods which gives the number of false matches when establishing the correspondence between feature points. The false matches are removed to obtain more accurate registration. False matches can be identified by Consistency Check [ZF03], Ratio Test [Lag11] or Symmetry Test [Lag11].Consistency Check applies two different matching methods to the same set of feature points. Only those pairs found by the both methods are considered as valid match points[ZF03]. Ratio Test takes a feature point in one image and then gets the distances of the two nearest matching points. Then, ratio of the

distances of the nearest points is used to remove out the unwanted match points. The principle is that the best match point is always nearer than other points [Lag11]. Symmetry Test performs two way matching to remove out false matches. Suppose, a point p₁ in first image has match point p₂ in second image. Then, in reverse matching, point p₂ should have p₁ as match point. The match points which hold this condition are considered as valid pairs which can be used for further processing [Lag11].

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 transform. 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 the corresponding point pairs to estimation 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.[SHB08] 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 thresholding to find out the corner points. We can get the required number of interest points by using proper threshold value. Corner detectors are not usually very robust. To overcome this problem, we select a very large number of corners are detected in two images than are needed for estimating a transformation. That is achieved by setting the proper threshold values.

3.7.1 Requirements of a Corner Detector

[PG11] 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).

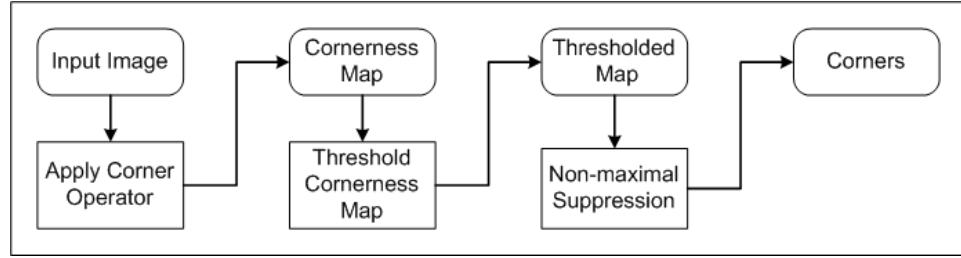


Figure 3.5: Flowchart for Corner Detectors

- 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 is shown in figure 3.5] are carried out by corner detectors:

- 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 [PG11]. 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.
- Threshold Cornerness Map:** The cornerness map is now thresholded to remove the false corners. We set a threshold value that will be able to retain true corners.². There is no straightforward method to choose the threshold value, it is application dependent and requires trial and error experimentation [PG11].
- 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.

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

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 pre-processing (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

The preprocessing step in image stitching improves image quality by removing noise and other unwanted artifacts. Sometimes, if the image size is larger, we select important smaller portion of the image which has higher probability of overlapping.

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. We also should note that the smoothing should not reduce the image information that we use for stitching. So, experimentally decide the effective image smoother and its parameters.

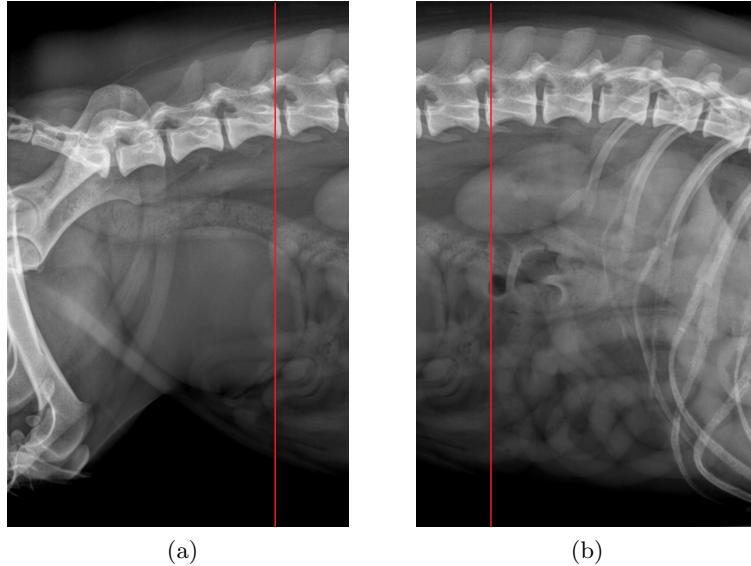


Figure 4.1: The second half of figure (a) and the first half of figure (b) chosen for faster estimation of motion parameters. The vertical line separates the overlapped region

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 image information, then 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 [PG11]. It is widely used algorithms for corner detection. The algorithm can be described as follows [PG11]:

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:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \quad (4.1)$$

where

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

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

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

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

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

$$C(x, y) = \det(M) - k(\text{trace}(M))^2 \quad (4.5)$$

where

$$\det(M) = \lambda_1 \lambda_2 = AB - C^2, \quad (4.6)$$

$$\text{trace}(M) = \lambda_1 + \lambda_2 = A + B \quad (4.7)$$

and

k=constant

3. Threshold the interest map by setting all C(x,y) below a threshold T to zero.
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)[Low04]. 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 [Low04].

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:

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

which is equivalent to

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (4.9)$$

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 [Low04]. David Lowe has derived the relationship of LoG and DoG images as:

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

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 [Low04] 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.

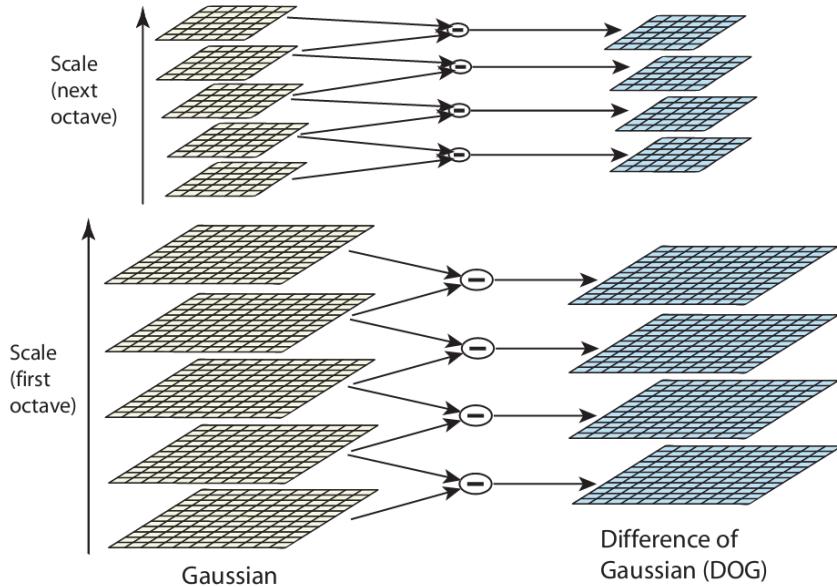


Figure 4.2: Construction of DoG Image [Low04]

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 [Li11].

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.[Low04]. To remove low contrast points, the magnitude of intensity is compared with a certain value, if it is less than the value, then reject it. Brown [BL02] 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, we check the condition(modification of equation 4.5):

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

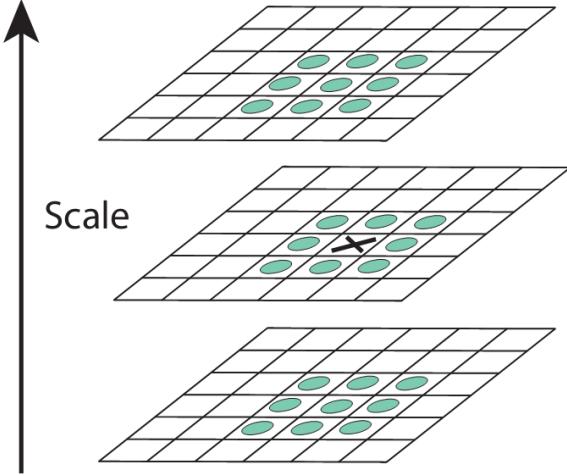


Figure 4.3: Maxima and minima idenfication of DoG images Image Source:[Low04]

3. **Orientation assignment:** The orientation is assigned as key point descriptor so that we can achieve invariance to image rotation [Low04]. 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:

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

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

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 be multiple key points created at the same location and scale but with different orientations [Low04].

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

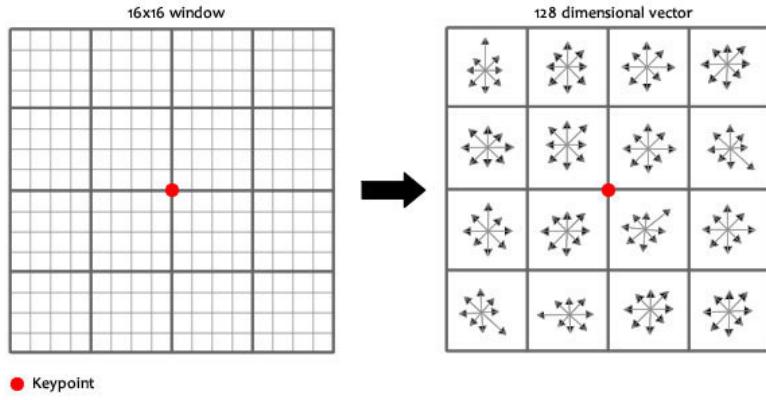


Figure 4.4: Creation of keypoint descriptor. Image gradients are used to create image descriptor. Image Source:[Sin10]

gradient magnitude and orientation at each point. Again, the magnitude is weighted according to the distance from key point(use Gaussian weighting function [Low04]), 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 $16 \times 8 = 128$ 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 [BETG06]. 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 relying only on regular PC, each one of the three steps (detection, description, matching) has to be fast. So, 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*[VJ01]. The value at a location $\mathbf{X}=(x, y)^T$ of integral image

$II(\mathbf{X})$ is calculated as:

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

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.

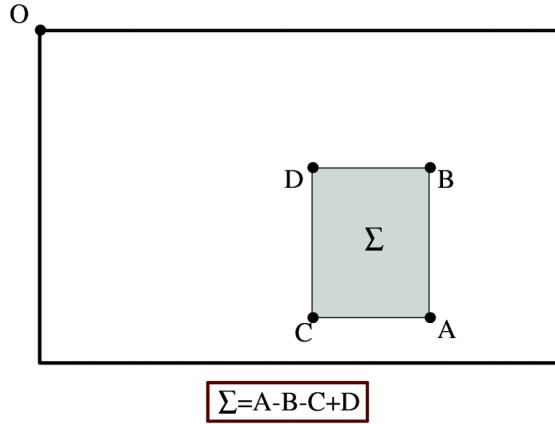


Figure 4.5: Calculation of sum of intensities inside a rectangular region of integral image. Image source: [BETG06]

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

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

where

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

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

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

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 [BETG06]. The approximated determinant of Hessian matrix represents the blob

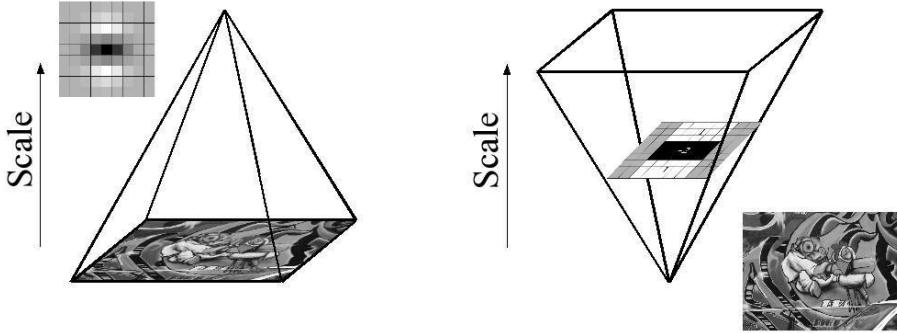


Figure 4.7: The use of integral images allows up-scaling of the filter at constant cost

response in the image at location \mathbf{x} . [BETG06] 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:

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

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[BETG06].

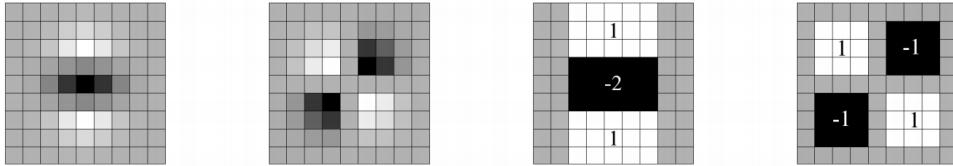


Figure 4.6: Left two images are cropped and decretized Gaussian second order partial derivative(L_{yy} , L_{xy}), right two images are corresponding approximated filters(D_{yy} , D_{xy})

- *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 $\sigma=1.2$). By doing this, we achieve computational efficiency and there will be no aliasing because we don't need to down-sample the image [BETG06].

The scale space is divided 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

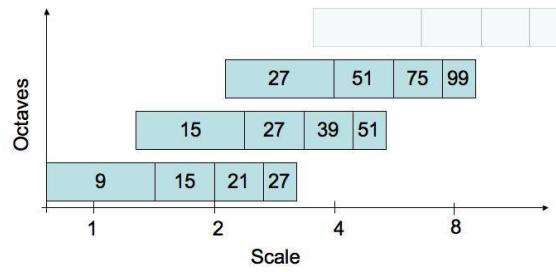


Figure 4.8: Filter side lengths for three different octaves. Image source: [BETG06]

- *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 $3 \times 3 \times 3$ neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space.[BETG06].
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 [BETG06]. 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.

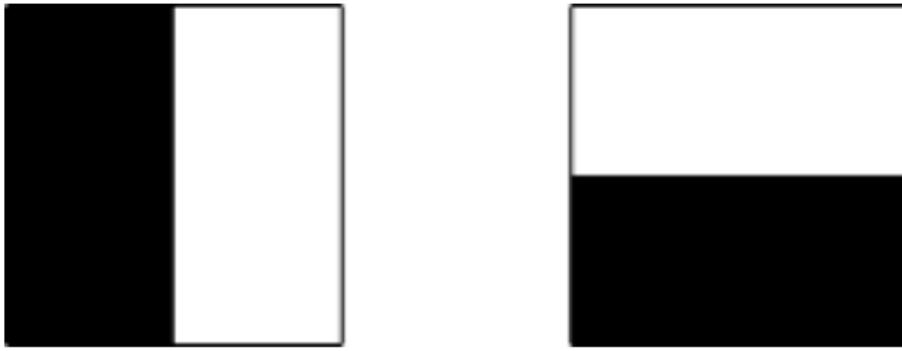


Figure 4.9: Haar Wavelets: The left one is response in x-direction, the right one is response in y-direction. Weight=1 for black region, -1 for white region. Image source [Eva09]

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 [Eva09]. 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 [Eva09]. This process is shown in figure 4.10.

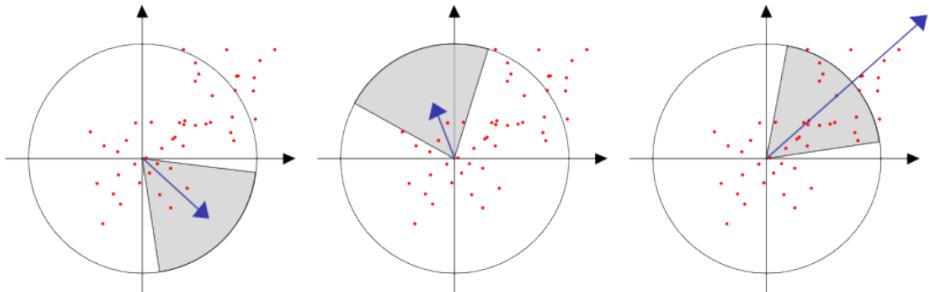


Figure 4.10: Orientation Assignment: The blue arrow is the sum of the responses. The largest one determines the dominant orientation. Image source [Eva09]

¹ σ here is the scale at which the point was detected.

- *Descriptor Components* We construct a square window of size 20σ around the interest point. The orientation of the window will be the dominant orientation we calculated above. This descriptor window is divided into 4×4 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 [Eva09]. Then, the feature vector for each sub-region will be

$$V_{subregion} = [\Sigma dx, \Sigma dy, \Sigma |dx|, \Sigma |dy|] \quad (4.20)$$

As shown in figure 4.11, each sub-region will have four values to the descriptor vector which results the overall vector length $4 \times 4 \times 4 = 64$. Evans in his article [Eva09] claims that the result SURF descriptor is invariant to rotation, scale, brightness and contrast².

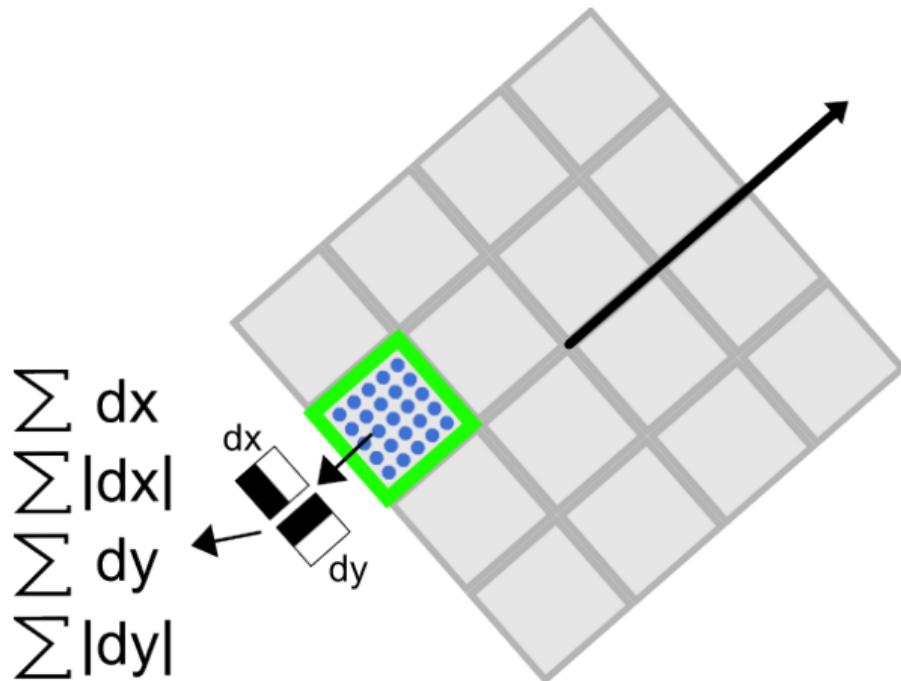


Figure 4.11: Descriptor Components Image source [Eva09]

²invariant to contrast is achieved after reduction to unit length [Eva09].

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 most popular feature extractors i.e. Harris, SIFT and SURT. 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.

4.3.1 Experiment 1: Computational Time

I have estimated the computational time for the feature extractors for different intensity and orientation images. The computational time measurement(in milliseconds) is presented in table 4.1 and the chart is shown in figure 4.12 .

Computing Time(milliseconds)					
	Original	High Intensity	Rotation	Scale	Noise
Harris	72.714	54.013	38.076	158.564	53.619
SIFT	402.993	350.434	270.318	1019.413	379.934
SURF	242.045	240.767	177.066	705.498	244.308

Table 4.1: Computational time of feature detectors

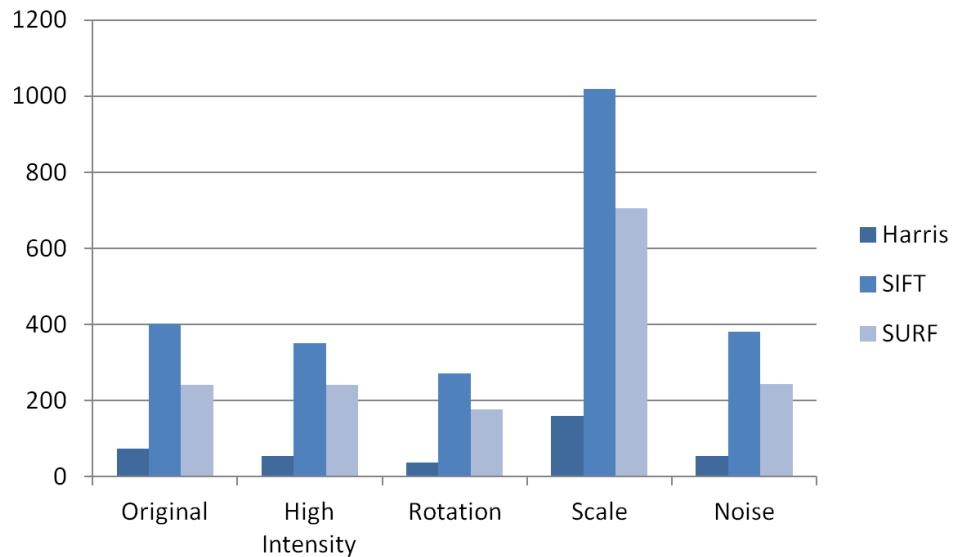


Figure 4.12: Chart showing the computational time(milliseconds) of feature extractors for different images

The chart shown above clearly reveals that Harris corner detector is fastest among detectors, and SIFT being 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

The stability is measured by the repeatability rate of the key-points for images with different intensity, orientation and scaling. The key-points detected in the original image is set as reference, and the repeated key-points for other images are counted. The higher number of repeated key-points indicates the higher stability implies a good feature extractor.

Table 4.2 presents the detected key-points count for original image and other variant images. I have also 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 4.2: Feature points count for stability measurement

From table 4.2, I have computed the repeatability rate(i.e. stability) of the feature extractors and presented in table 4.3.

Stability(%)		HARRIS	SIFT	SURF
	Brighter	55.51	73.52	80.41
	Rotated	34.69	45.42	42.96
	Scaled	33.46	84.64	56.35
	Noisy	71.02	94.77	73.53

Table 4.3: Stability measurement of the corner detectors

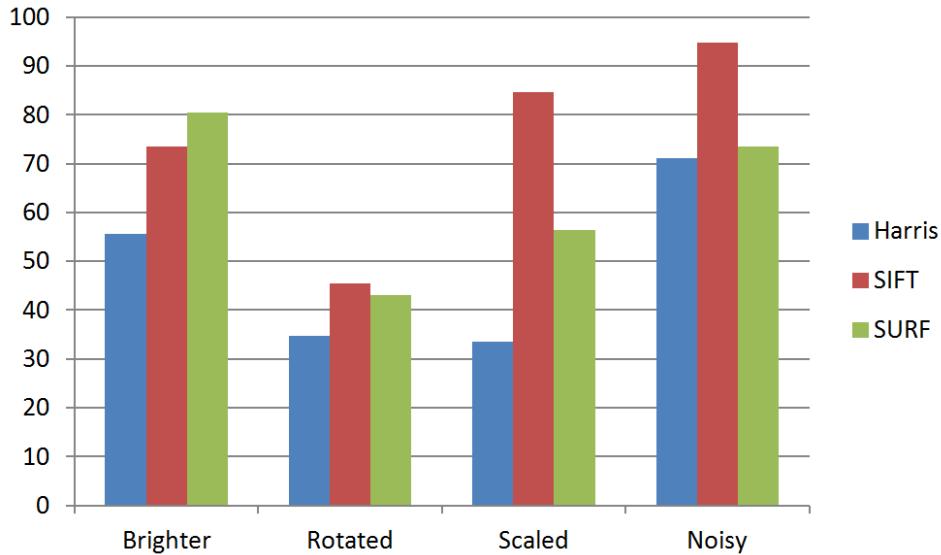


Figure 4.13: Chart showing the repeatability rate for feature extractors

Repeatability rate is an important factor to select a good feature extractor and the chart presented in figure 4.13 clearly shows that Harris detector is not appropriate feature extractor because of its low repeatability rate. So, I will select only SIFT and SURF extractors for further experimentation.

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 part of the section gives exhaustive *k-nearest-neighborhood(kNN)* method, then we discuss *approximate nearest neighborhood(ANN)* method. The final step is to fine-tune the matched points which 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 $P = \{p_1, p_2, p_3, \dots, p_m\}$ 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. The nearest points need not necessarily be the actual matched points, further tests are necessary to increase

¹Each SIFT points has 128 features while SURF point has 64 features

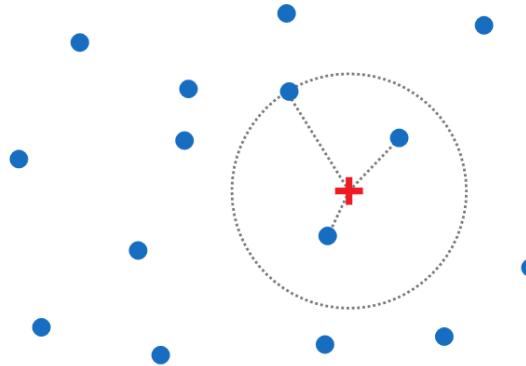


Figure 5.1: kNN search problem with $k=3$. Image source: [GDNB10]

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 [ML09].

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* [Lag11].

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 we define a threshold T . Then, we reject the points as false matches which satisfy the following condition:

$$\frac{d_1}{d_2} > T \quad (5.1)$$

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 selected i.e. higher the threshold, larger number of matches.

5.3.2 Symmetry Test

The symmetry test identifies the false matches by two way checking of a match. In symmetry test, suppose, for a match, a point p_1 has a matched point p'_1 , then, for the point p_1 to be true matching point, p'_1 should have p_1 as matching point. It is very simple and we do not need any parameters for 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 verses 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 features matching is quite faster than SIFT features matching. The computational cost for SIFT features matching increases drastically when the number of points increases. I have carried out exhaustive kNN matching to match the features.

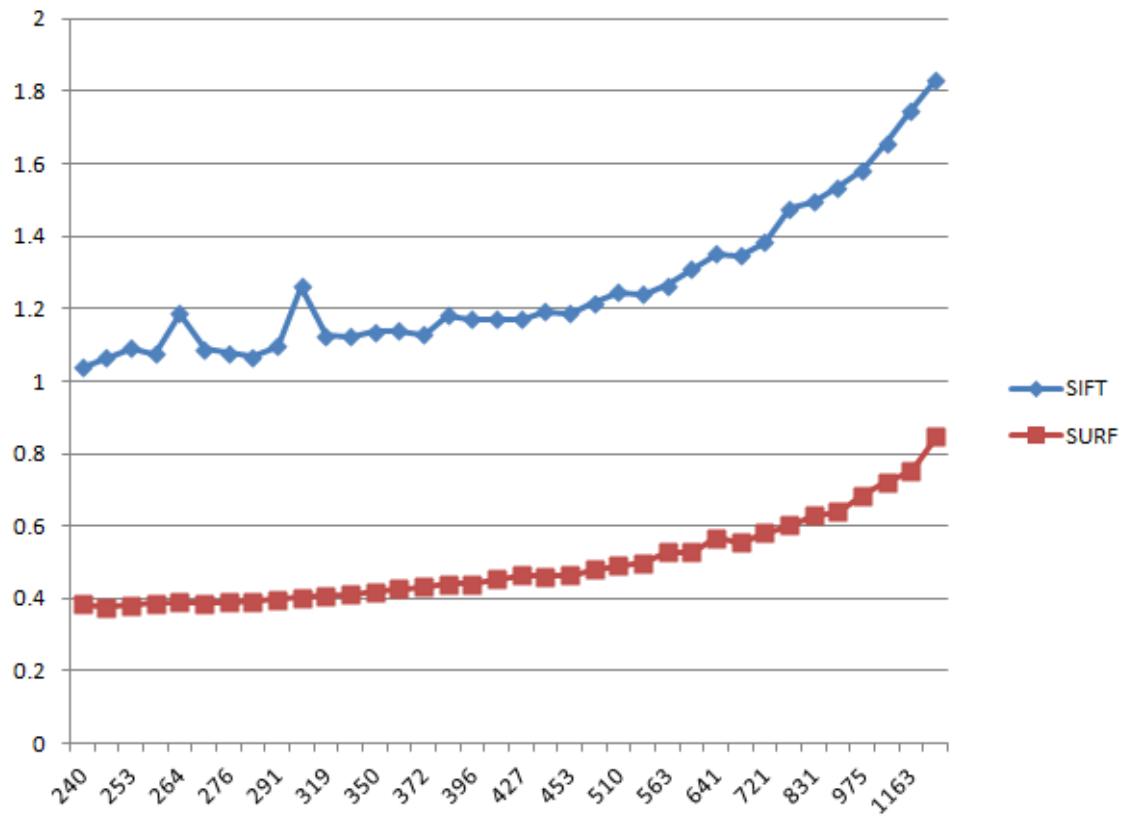


Figure 5.2: Comparison of SIFT and SURF: Feature points verses computational time

SIFT is claimed to more accurate because of its high dimensional fea-

tures, its high computational time is the main drawback making it inappropriate for real time or user-centered applications like medical software. I have chosen less accurate SURF method because it is faster, the inaccuracy in matches are later identified by further tests. So, overall matching process is increased by using SURF and some tests to get accurate matches(section 5.4.3).

5.4.2 kNN verses ANN

In this section, I will present experimental results on kNN and ANN matching methods for SURF features which is presented in graph shown in figure 5.3. The graph clearly shows that ANN matching faster than kNN matching. The graph shows the exponential increase of computational complexity for kNN matching which implies the matching of larger points is computationally expensive. Since, in practice, we have larger number of feature points to be matched, ANN matching is better.

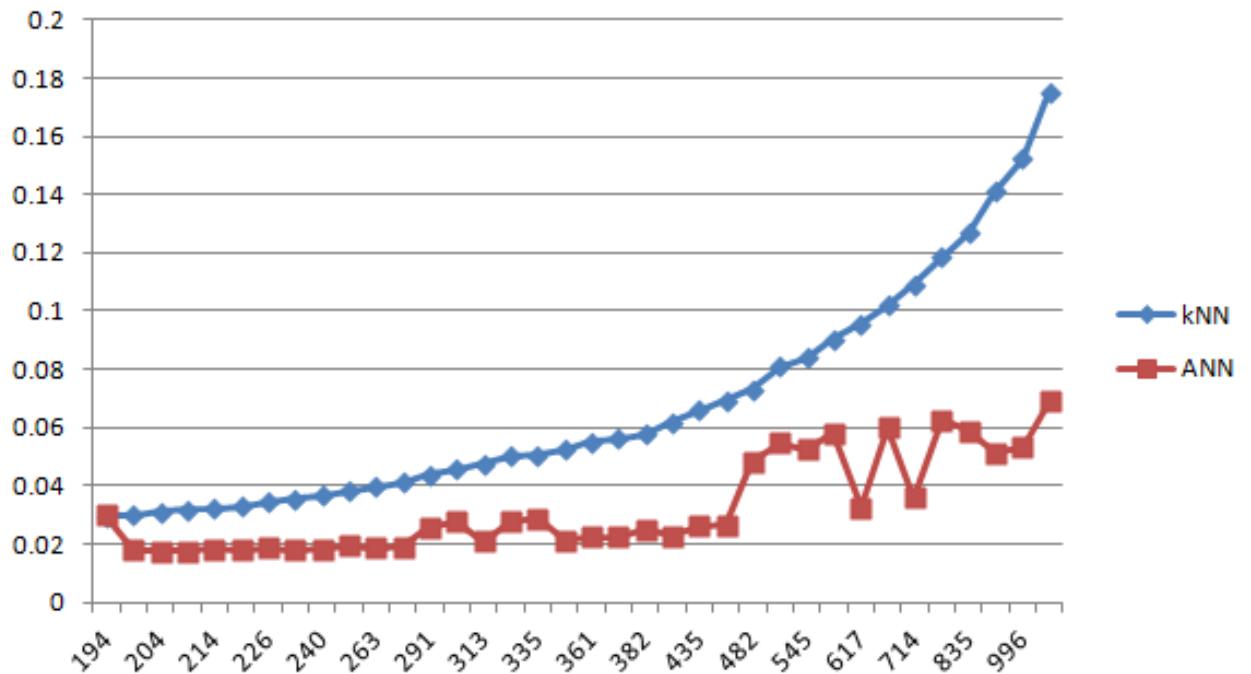


Figure 5.3: Comparison of kNN and ANN: Feature points vs. matching time

5.4.3 Getting Accurate Matches

In this section, I will experiment on getting correct matches. 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 match-

ing 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 20 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 clearly see in figure 5.4c. The remaining less number of inaccurate matches(outliers) are removed by Robust Estimation methods in next chapter.

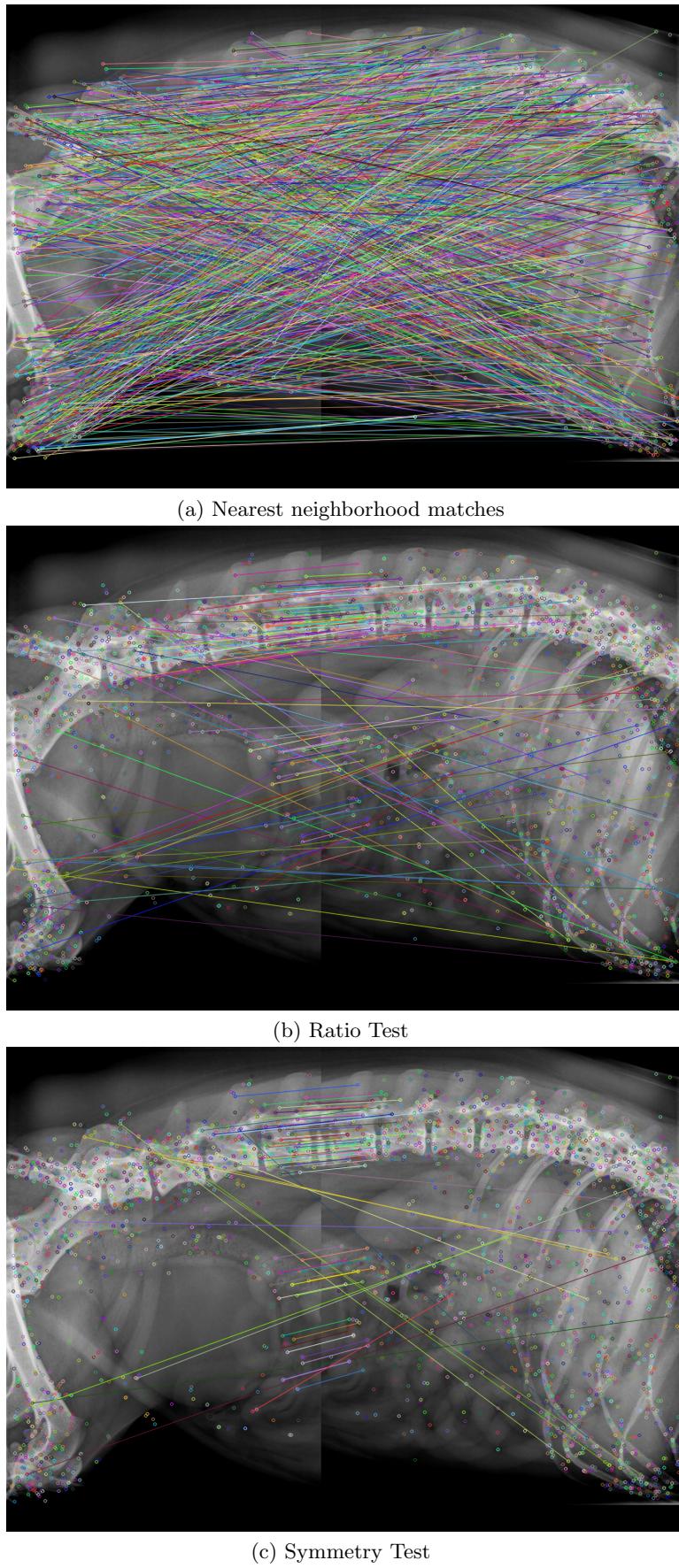


Figure 5.4: Steps of getting best matches

Chapter 6

Homography Estimation

After we get the matching point pairs, the next step is to estimation transformation model using those pairs. This chapter describes the methods to get the accurate motion parameters for image transformation.

If there is only translation or rotation between the images to be stitched, then stitching process would be simpler only affine transformation parameters would solve the problem. Since we have to register images which are taken with different camera angle; we have to estimate translation, rotation, projection parameters for which we need to estimate *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. So H can be considered as homography matrix and it has 8 degrees of freedom although it has 9 elements [Dub09]. In other words, we need to solve for 8 unknowns to get the homography matrix. The feature correspondences obtained using matching methods (discussed in chapter 5) are used to estimate 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:

$$x'_i = Hx_i \quad (6.1)$$

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

$$c \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = H \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \quad (6.2)$$

Hartley and Zisserman [HZ04] suggested to use a normalization step in DLT because the DLT algorithm is dependent on the origin and scale of co-ordinate 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, the interested readers are suggested to study chapter 4 of Hartley and Zisserman book [HZ04]

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 [FB81] and it is the most commonly used robust estimation method for homographies [Dub09]. This method selects 4 correspondences randomly and computes a homography 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 has largest number of inliers is selected. Then homography is recomputed from all the inlier correspondences in that iteration.

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 outliers¹. Hartley and Zisserman in [HZ04] derived a formula to compute the number of iterations required for RANSAC:

$$N = \log(1 - p) / \log(1 - (1 - \epsilon)^s) \quad (6.3)$$

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

$$\epsilon = 1 - \frac{\text{number_of_inliers}}{\text{total_number_of_points}} \quad (6.4)$$

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

[HZ04]² claims that if we have 5% outliers(i.e. $\epsilon=0.05$) 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 [Rou84] 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 [Rou84] [Ond01] and it has the characteristics of being highly resistance to high proportion of outliers [Ond01].

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

¹it is almost to test each combination of random samples, so we need to limit the number of iterations

²see comparison table in chapter 4 of [HZ04]

6.3 Experimental Results

The first part of this section presents the result of RANSAC for homography estimation by identifying outliers. Since LMS method(section 6.2.2) works good when there are more than 50% inliers, but it is not always guaranteed to have more than half accurate matches³, so I will implement RANSAC method for robust estimation because it works even if there are larger 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.

Next, I will carry out the experiment on the effect of distance threshold values(σ) for inliers and outliers. In experiment, the distance threshold σ measured in terms of the number of pixels.

6.3.1 Robust Estimation with RANSAC

I have carried out experiments with RANSAC method. I have set the selected maximum possible number of iterations upto 2000 and distance threshold(σ) 2 pixels. The result identified the outliers as inaccurate matches and inliers as accurate matches. From the figure 6.1, we can see that all matches RANSAC identified are correct.

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

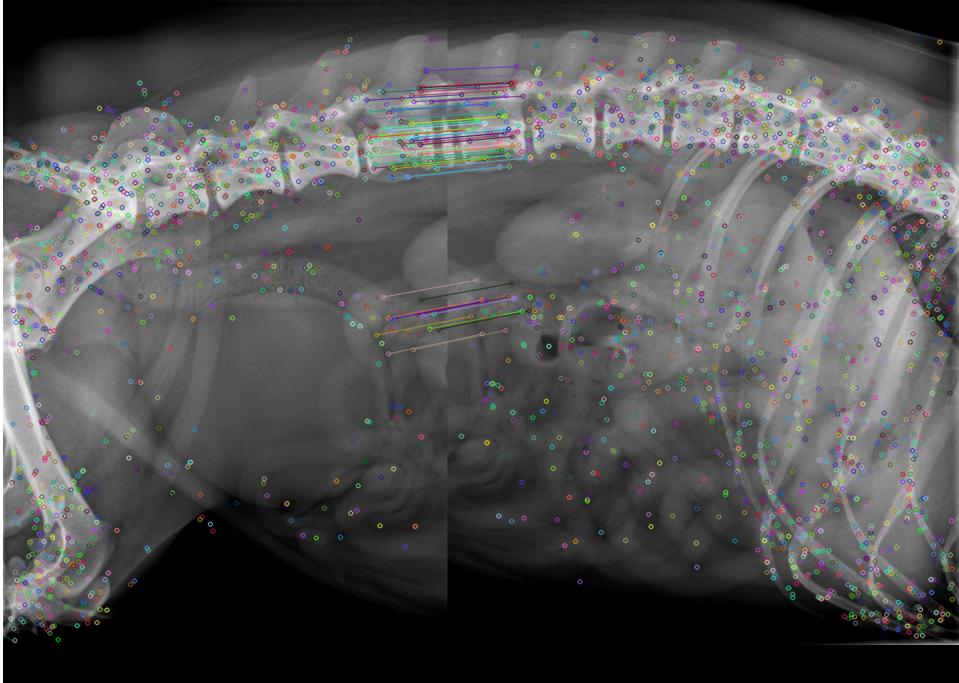


Figure 6.1: matches after removing outliers($\sigma=3$ pixels). This figure shows all matches are accurate.

6.3.2 Effect of Distance Threshold(σ)

The number of inliers is changed when we change the distance threshold. 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 we get true inliers. With small possible value of $\sigma=1$ pixel, I got 50 matches. Since the figure 6.1 shows there are 62 accurate matches which implies that $\sigma = 1$ is missing some true inliers. The increase of σ increases the inliers resulting high probability of getting false inliers. So, the best accepted values of σ seems to be either 1 or 2.

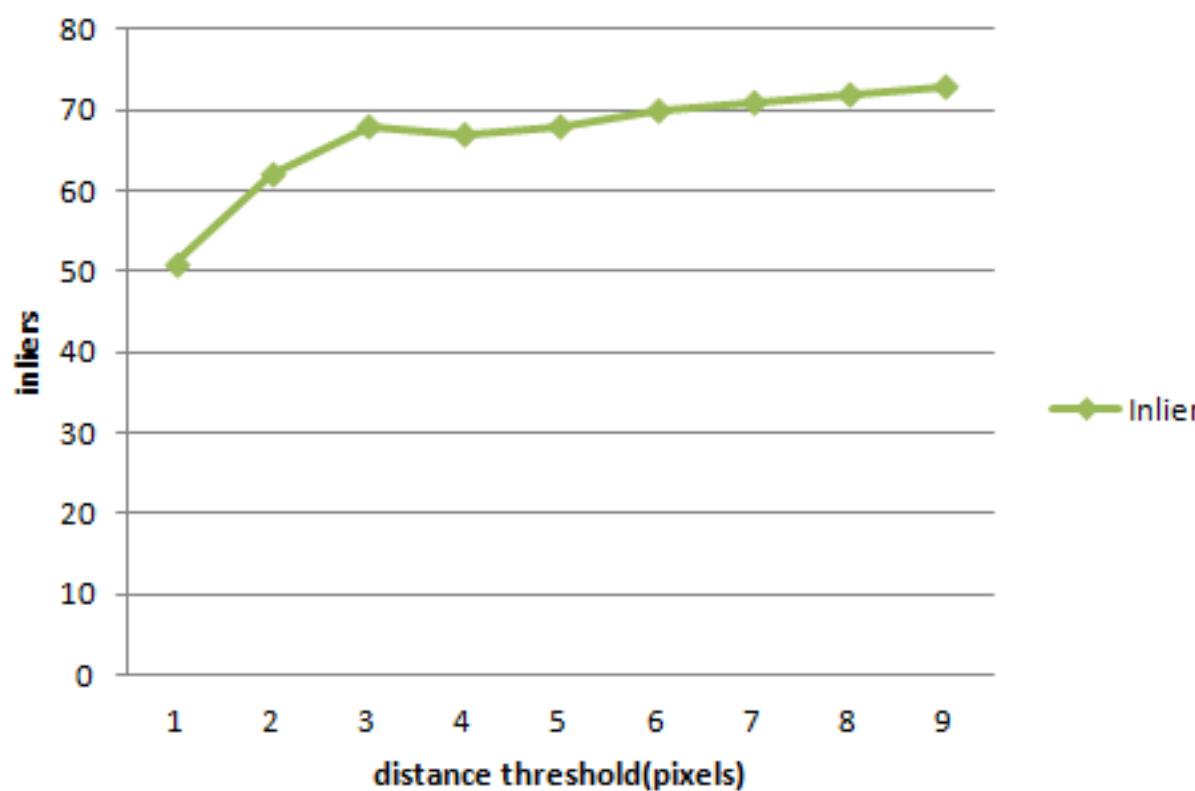


Figure 6.2: graph showing inliers verses distance threshold

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 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. Transformation overlaps the common part of images and 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 [LZW06].

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 [Sze06].

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. Generally, we have to devise the general algorithm which does not use any direction information. Later, we can modify the algorithm which used direction information for faster stitching. 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.

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

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 $(x_{max}-x_{min})$ and $(y_{max}-y_{min})$ respectively. Unless the two images are exactly same, the composite image size is always greater than either image.

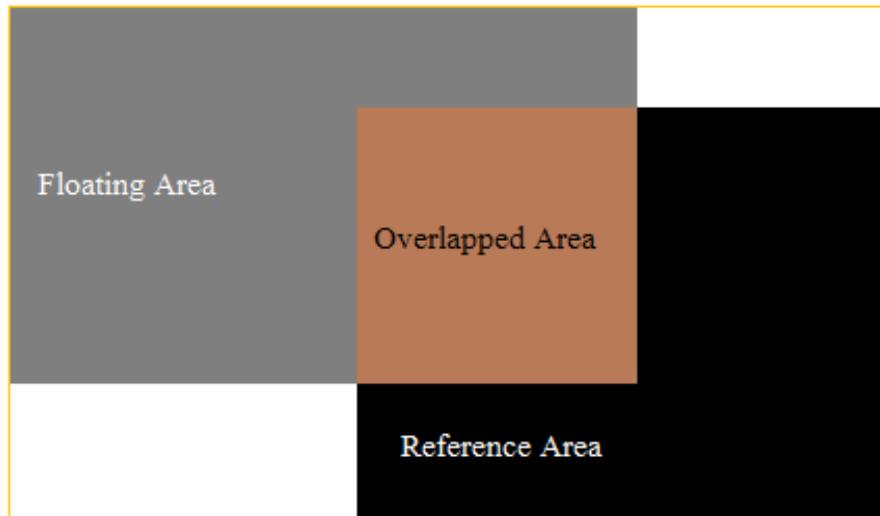


Figure 7.1: Compositing of two images. The area within golden line is the compositing area. The gray area is floating image while black area is reference image. The overlapped area is painted with brown color.

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 [Dub09], we have to implement exposure compensation technique discussed in section 7.3.

7.2 Blending

The overlapping regions are blended for exposure compensation and mis-alignments. In this section, I will describe some blending techniques which remove discontinuities in the composite image to create visually appealing stitched image. In this section, I will discuss some popular blending techniques: *Optimal Seam Blending*, *Alpha Blending* and *Laplacian Pyramid Blending*.

7.2.1 Optimal Seam Blending

Optimal seam blending method search for a curve in the overlap region on which the difference the images are minimal. If I_1 and I_2 are overlapping parts, if 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 [LZPW06].

7.2.2 Alpha Blending

Alpha blending, also called *feathering*, is simple and effective algorithm. Alpha blending assigns the weight values(i.e. α) to the pixels of the overlapping area. For $\alpha=0.5$, 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 $\alpha=0$, then the pixel has no effect in composite region while $\alpha=1$ implies the pixel is copied there. Suppose, composite image I is created from horizontally aligned images I_1 (left) and I_2 (right), then

$$I = \alpha I_1 + (1 - \alpha) I_2 \quad (7.2)$$

Initially, we start with $\alpha=1$ (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).

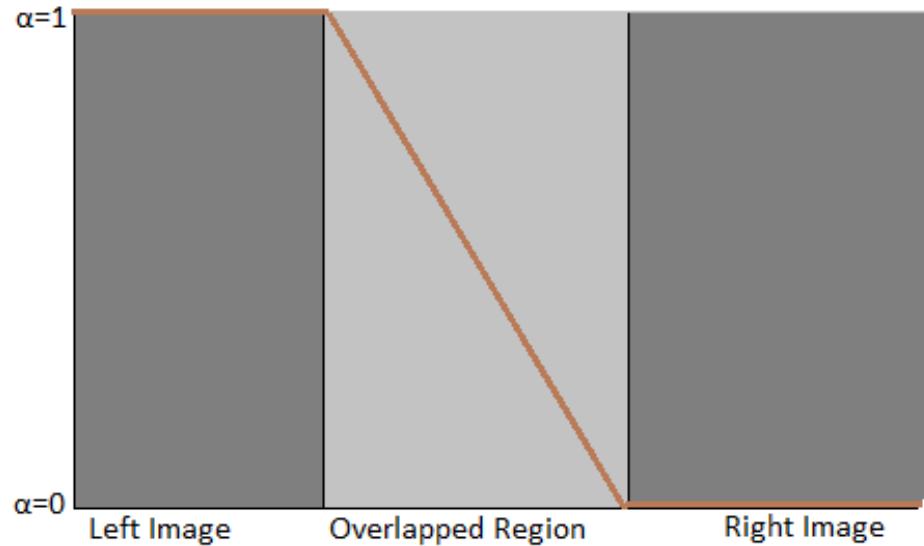


Figure 7.2: Alpha Blending: α decreases from 1 to 0 in the overlapping region

The above method is for pure horizontally aligned images; and similar technique can be used for pure 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* [RLE⁺05].

7.2.3 Laplacian Pyramid Blending

The *Laplacian 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 Laplacian Pyramid Blending consists of the following steps [Sze06]:

- 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 result 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 [Sze06]. This problem is more dominant if there is some rotation between the images while registration.¹

The transfer function based approach defined by [UES01] 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 [UES01].

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 the matched points. The transfer function parameters are estimated using a number of matched pairs². The estimated transfer function maps the intensity of I_1 to intensity of I_2 ; thus works as an intensity leveling function.

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

²number 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 on two 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 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 types of alignments.

7.4.1 Alpha Blending verses 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)	0.105029	1278.804443	0.106111	18750.882813
Pyramid Blending (Level 4)	0.119235	4097.895996	0.140580	21644.531250
Alpha Blending	0.015721	260.586365	0.015268	17564.503906

Table 7.1: Alpha Blending verses 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 loses more image information than Alpha Blending(see figure 7.3). The stitched medical image should preserve image information while blending and Alpha Blending is doing well.

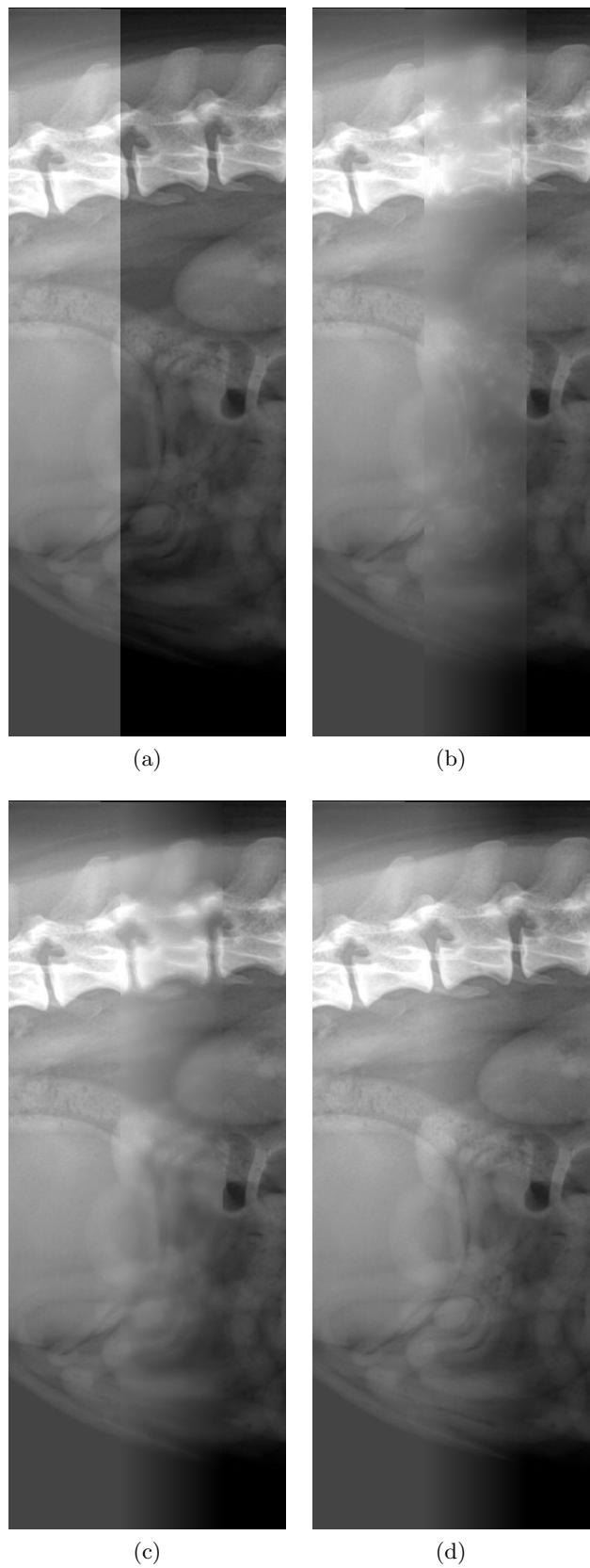


Figure 7.3: Comparison of results of Pyramid Blending and Alpha Blending. (a) is the composite image, (b) is level 4 pyramid blended, (c) is level 2 pyramid blended and (d) is alpha blended image. The blurring effect in Pyramid Blending is undesirable because it loses image information.

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 some experiments on the images with complex alignment. The horizontal or vertical blending for the image shown in figure 7.4a produces visible seams shown in figure 7.4b.

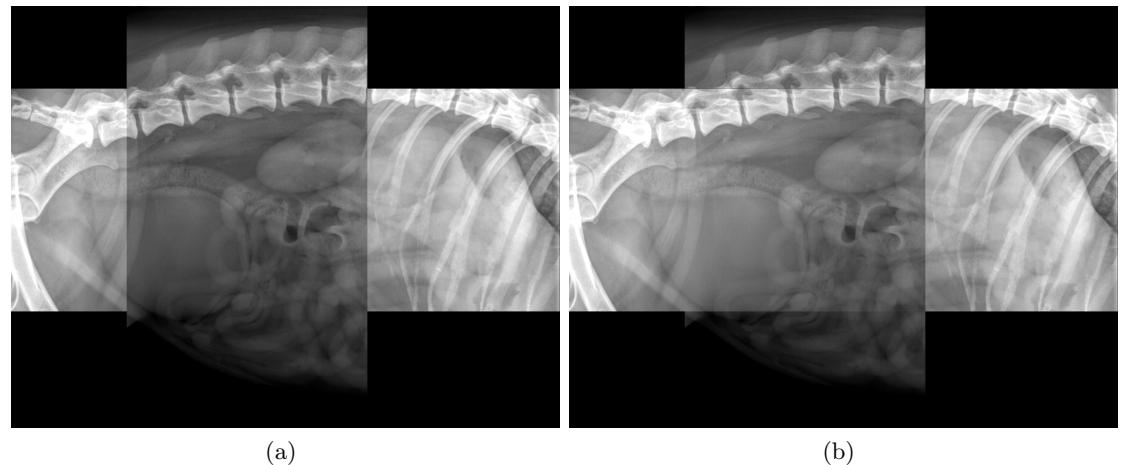


Figure 7.4: Figure (a) is raw composite image, figure (b) is horizontally blended image. The seams are clearly visible on the join of the images.

So, to remedy this problem, a modified Alpha Blending method has been implemented in which we create separate 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.

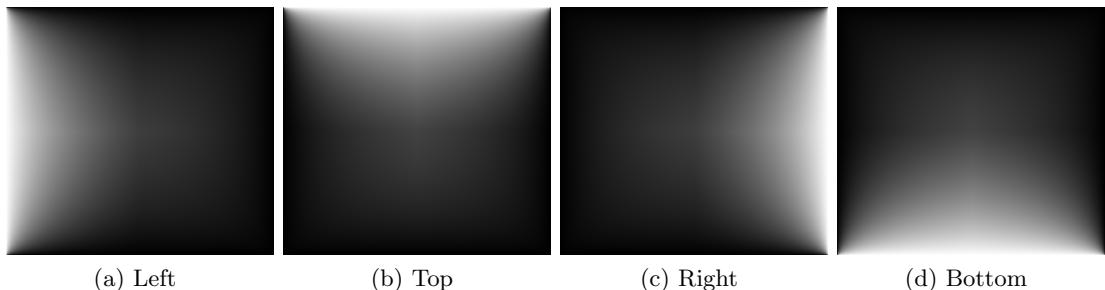


Figure 7.5: Blending masks. (a) is mask for left image,(b),(c), (d) are masks for top, right and bottom images respectively.

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)

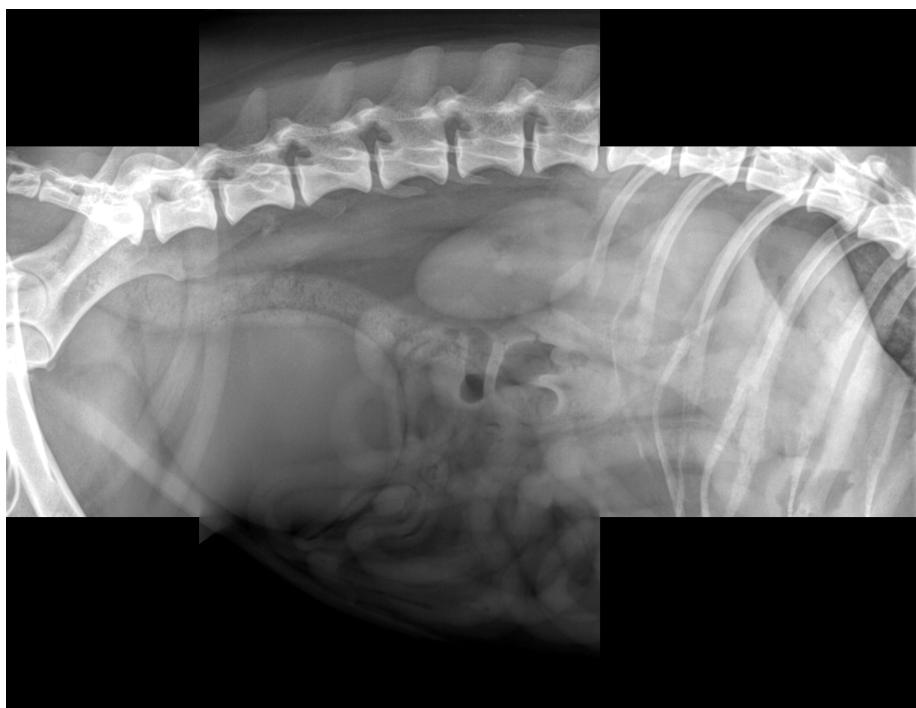


Figure 7.6: Use of multiple blending masks gives better blending result.

Chapter 8

Conclusion

This thesis evaluates the image analysis methods which can be used for stitching of medical X-ray images. Image stitching process includes several steps making it a slower process which is not suitable 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 the faster methods and I have analyzed and evaluated three most popular feature extractors: Harris, SIFT and SURF. Harris detects the corner points faster, 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. But for feature matching, although, it gives accurate result, SIFT is suffering from its high computational complexity, so, SURF is better option which is faster than SIFT. The exhaustive nearest neighborhood(kNN) method is modified to approximate nearest neighborhood(ANN) to get faster matching sacrificing some inaccuracies. Since, nearest neighborhood based methods gives just nearest matched point, the application of accuracy tests such as Ratio Test and Symmetry Test are quite effective to remove the false matches. The use of SURF features matching and approximate neighborhood methods are intended to make the stitching process faster, however, they give less accurate result. So, the selection of RANSAC method of robust estimation is preferable because it works on the data which consists of a lot of outliers.

I have evaluated two popular blending techniques: Alpha Blending and Pyramid Blending. Although Pyramid Blending gives smooth result, it is slower process and it loses image information. So, Alpha blending is chosen which gives better result and it is faster also. The alpha blending can be modified to work on complex alignments which occurs if the images are ro-

tated and taken in perspective view. In such a scenario, the modified Alpha Blending is effective where we use blending masks to assign the weights of the pixels of contributing images on the blended region. The method is quite effective since it successfully removes the seams and discontinuities of 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, the extension will be to carry out research on obtaining more optimized algorithms to work for very high resolution images. The X-ray image generally consists of a lot of background area which consists of no information for image stitching. So, we can implement some technique which selects bones and muscles in the image, and discard the other areas.

The blending method is resulting a seam on the join if the stitching is carried out for different intensity rotated images. 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. This works perfect if the images have similar intensity, it fails to work perfectly for different intensity images. The extension could be to use some transformation method to get the accurate pixel intensity to be filled in the unassigned pixels.

Bibliography

- [BETG06] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features(surf). Technical report, Department of Electrical and Computer Engineering, University of Windsor, 2006.
- [BL02] Matthew Brown and David Lowe. Invariant features from interest point groups. Technical report, Department of Computer Science, University of British Columbia, 2002.
- [DAC09] David L. Donoho and Ery Arias-Castro. Does median filtering truly preserve edges better than linear filtering? Technical report, Institute of Mathematical Statistics, University of California, San Diego and Standford University, 2009.
- [Dub09] Elan Dubrofsky. Homography estimation. Master's thesis, The University of British Columbia, 2009.
- [Dur10] Joe Durnavich. Making sense of the head x-rays.
<http://mcadams.posc.mu.edu/xray/reading/reading.htm>, apr 2010. Accessed: 20/07/2012.
- [Eva09] 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>.
- [FB81] 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.
- [Gav98] D.M. Gavrilla. Multi-feature hierarchical template matching using distance transforms. *Daimler-Benz AG, Research and Technology*, 1998.
- [GDNB10] 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.
- [GP99] 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.
- [HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [KBR⁺10] 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.
- [Lag11] Robert Laganiere. *OpenCV 2 Computer Vision Application Programming Cookbook*. 2011 Packt Publishing, Packt Publishing Ltd., Birmingham, UK, May 2011. <http://www.packtpub.com>.
- [Li11] Shuo Li. Registration of 3D volumetric CT images. Master’s thesis, Uppsala University, Department of Information Technology, November 2011. IT series, 11 080.
- [Lin96] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30:465–470, 1996.
- [LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Computer Science Department, Carnegie-Mellon University*, pages 121–130, 1981.
- [Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, January 2004. Computer Science Department, Vancouver, B.C., Canada.
- [LZPW06] 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*. The Hebrew University of Jerusalem, 2006. <http://www.wisdom.weizmann.ac.il/~levina/papers/eccv04-blending.pdf>.
- [ML09] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Interna-*

- tional Conference on Computer Vision Theory and Application VISSAPP'09), pages 331–340. INSTICC Press, 2009.*
- [MP06] Xue Mei and Fatih Porikli. Fast image registration via joint gradient maximization: Application to multi-modal data. *MIT-SUBISHI ELECTRIC RESEARCH LABORATORIES*, September 2006. <http://www.merl.com>.
- [Ond01] A. Ozlem Onder. Least median squares: A robust regression technique. Technical report, EGE AKADEMİK BAKIS, 2001. <http://eab.ege.edu.tr/pdf/1/c1-s1-m8.pdf>.
- [PG11] 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>.
- [RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *European Conference on Computer Vision*, pages 430–443, 2006. Department of Engineering, Cambridge University.
- [RLE⁺05] 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.
- [Rou84] Peter J. Rousseeuw. Least median of squares regression. *The American Statistical Association*, 79(388), dec 1984. <http://web.ipac.caltech.edu>.
- [SHB08] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Library of Congress Control, 3rd edition, 2008.
- [Sin10] Utkarsha Sinha. Sift:Scale Invariant Feature Transform. [http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/7/](http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/), May 2010. Accessed: 17/09/2012.
- [SK99] Harpreet S. Sawhney and Rakesh Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 235–243, 1999.
- [Spr] Perry Sprawls. X-ray Image Formation and Contrast. <http://www.sprawls.org/ppmi2/XRAYCON/>. Accessed: 17/09/2012.

- [SS97] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environmental maps. *Microsoft Research*, 1997.
- [Sze06] Richard Szeliski. Image alignment and stitching: A tutorial. *Microsoft Research*, dec 2006. <http://research.microsoft.com/pubs/70092/tr-2004-92.pdf>.
- [UES01] Matthew Uyttendaele, Ashley Eden, and Richard Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. pages 509–516, Kauai, Hawaii, December 2001. IEEE Computer Society.
- [VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition*, 2001.
- [War06] 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.
- [Wik12] Wikipedia. Google street view. http://en.wikipedia.org/wiki/Google_Street_View, sep 2012. Accessed: 16/09/2012.
- [Woe09] Harald Woeste. *Mastering Digital Panoramic Photography*. Rocky Nook, 1st edition, November 2009.
- [ZF03] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Department of Image Processing, Institute of Theory and Automation, Academy of Science of the Czech Republic*, 2003.
- [ZT98] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.