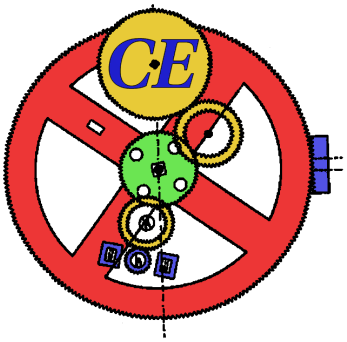


Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2005

MSc THESIS

Developing and Implementing Phase Normalization and Peak Detection for Real-Time Image Registration

Meng Ma

Abstract

Finding a known object in static pictures or real-time streaming pictures is always an interesting topic for many applications. A well known and reliable method is Symmetric Phase Only Matched Filter (SPOMF). Regular SPOMF can work well with non-rotation and scaling object but show poor performance for rotated and scaled object. A solution for this problem is to map the absolute value of the spectrum into a polar coordinate system, then detect the rotation angle and scaling factor, and finally compensate for those factors by rotating and scaling the template image. In the SPOMF operation, phase normalization in the frequency domain is required to avoid generating high peaks because of high brightness in images. In this thesis work, a phase normalization algorithm is developed, and experiments indicate that this algorithm can use limited amount of bits (even one bit) to represent phase angles but still show acceptable quality. Also, a self-adaptive peak detection algorithm is developed to detect peaks in various magnitudes. Both two algorithms are implemented on a reconfigurable and scalable platform based on PowerFFT and FPGA hardware.

TU Delft
Delft University of Technology

eonic
digitally mastering the spectrum

CE-MS-2005-02

TU Delft
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Developing and Implementing Phase Normalization and Peak Detection for Real-Time Image Registration

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Meng Ma
born in Hangzhou, China

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Developing and Implementing Phase Normalization and Peak Detection for Real-Time Image Registration

by Meng Ma

Abstract

Finding a known object in static pictures or real-time streaming pictures is always an interesting topic for many applications. A well known and reliable method is Symmetric Phase Only Matched Filter (SPOMF). Regular SPOMF can work well with non-rotation and scaling object but show poor performance for rotated and scaled object. A solution for this problem is to map the absolute value of the spectrum into a polar coordinate system, then detect the rotation angle and scaling factor, and finally compensate for those factors by rotating and scaling the template image. In the SPOMF operation, phase normalization in the frequency domain is required to avoid generating high peaks because of high brightness in images. In this thesis work, a phase normalization algorithm is developed, and experiments indicate that this algorithm can use limited amount of bits (even one bit) to represent phase angles but still show acceptable quality. Also, a self-adaptive peak detection algorithm is developed to detect peaks in various magnitudes. Both two algorithms are implemented on a reconfigurable and scalable platform based on PowerFFT and FPGA hardware.

Laboratory : Computer Engineering
Codenumber : CE-MS-2005-02

Committee Members :

Advisor: Arjan van Genderen, CE, TU Delft

Chairperson: Stamatis Vassiliadis, CE, TU Delft

Member: Patrick Dewilde, CAS, TU Delft

Advisor of Eonic : Peter Beukelman, Eonic B.V. Delft

I dedicate this thesis to my parents for their love and support.

Contents

List of Figures viii

List of Tables ix

Acknowledgements xi

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Rotation and Scale Invariant Template Matching . . . 1

2 RSI Project Description 3
- 2.1 Theory . . . 3
- 2.2 Symmetric Phase Only Matched Filter . . . 7
- 2.3 Log-Polar Mapping and Interpolation . . . 7
- 2.4 Rotation and Scale . . . 8
- 2.5 Reconfigurable Platform . . . 8

3 Phase Normalization 11
- 3.1 Why Normalization . . . 11
- 3.2 Different Solutions . . . 13
  - 3.2.1 Normal Solution . . . 13
  - 3.2.2 Lookup Table Solution . . . 13
  - 3.2.3 CORDIC Solution . . . 13
- 3.3 The CORDIC Solution . . . 14
  - 3.3.1 The Theory . . . 14
  - 3.3.2 Normalization with CORDIC . . . 15
  - 3.3.3 Lookup table and CORDIC hybrid solution . . . 16
- 3.4 Reduced Bit Number for Phase Representation . . . 17
  - 3.4.1 Sign Bit Only (SBO) solution . . . 18
  - 3.4.2 Mathematical Explanation . . . 19
- 3.5 Hardware Implementation . . . 23

4 Peak Detection 25
- 4.1 A Self-Adaptive Algorithm for Threshold Determination . . . 25
- 4.2 Sequenced Queues for Maximum Pixels . . . 28
  - 4.2.1 Sequenced queue . . . 28
  - 4.2.2 Binary Sorting . . . 29
  - 4.2.3 Parallel Sorting . . . 29
  - 4.2.4 Hardware implementation of the parallel sorting unit . . . 30
- 4.3 Adjacent Peak Removing . . . 33

4.3.1 Removing algorithm exploration 33
4.3.2 Hardware implementation of adjacent peak removing 35
4.4 Top Level Architecture of Peak Detection 37

5 Hardware Integration **39**
5.1 The PowerFFT Board 39
5.2 Control State Machine 40

6 Test Results **43**
6.1 The Cross Artifacts and the Edge-Fading Filter 43
6.2 A Satellite Image Test 44

7 Conclusion **47**
7.1 Conclusion 47
7.2 Future Work 47

Bibliography **50**

List of Figures

2.1 Image information distribution in frequency domain 4
2.2 Schematic of RSI matching 6
2.3 Polar FFT 8
2.4 Four board architecture of RSI 9

3.1 SPOMF algorithm dataflow 11
3.2 SPOMF demonstration with and without normalization 12
3.3 Phase normalization with normal method 13
3.4 Phase normalization with lookup table 14
3.5 Phase normalization with CORDIC 14
3.6 Schematic of CORDIC normalization 16
3.7 A hybrid solution of lookup table and CORDIC 17
3.8 Comparison of full CORDIC solution and hybrid solution 18
3.9 SPOMF with phase representation of fewer bit number 19
3.10 SPOMF of mig-25 using sign bit only normalization 19
3.11 Comparison of SPOMFs with full CORDIC and SBO 20
3.12 Shift information represented by phase rotation circles 21
3.13 The well kept overall characteristic with SBO normalization 21
3.14 Faulty peaks in the clean match 23
3.15 Normalizer hardware 24

4.1 Four basic classes of correlated image in 1D representation 26
4.2 Tests of threshold determination algorithm (K = 0.5) 27
4.3 Dataflow of threshold determination algorithm 28
4.4 Tree search for sequenced memory queue 30
4.5 Comparator network for parallel sorting 31
4.6 Hardware implementation of parallel sorting queue 32
4.7 Mechanism of the queue updating 32
4.8 Block based adjacent peak removing algorithms 34
4.9 Pixel based adjacent peak removing algorithm 34
4.10 Chain effect 35
4.11 The pixel format 35
4.12 Schematic of adjacent peak removing unit 36
4.13 The check unit in the adjacent peak removing 37
4.14 Schematic of detection top level 38

5.1 The PowerFFT board 39
5.2 Communication signals to the switched fabric and sequencer 40
5.3 The control state machine for detector and normalizer 42

6.1 Cross Artifact of rectangular windowing 43
6.2 Image windowing using circular filter and its frequency 44
6.3 Image windowing using Edge-Fading Circular Filter 44

6.4 Satellite search image and template image 45
6.5 Rotation and scale detection 45
6.6 Rotation and scale compensations 46
6.7 Detected location of the sport center 46

List of Tables

3.1 The value of arctan(2^{-i}) with 10-bit precision 16
3.2 Peaks' magnitude and noise levels under different bit number representations 22

5.1 Control signal definitions 41

Acknowledgements

During the different stages of this thesis project that I performed at Eonic B.V., I received a lot of kind help that guided my work heading to the right direction. In this section, I would like to thank them all.

First of all, I want to thank all the people that made this project possible. They provided me such a great opportunity to perform my thesis project in a leading IT company - Eonic B.V.

I would like to thank my supervisor at Eonic B.V. Mr. Peter Beukelman and my supervisor at TU Delft Dr. Arjan van Genderen for their valuable experience and encouragement all the time along my work. Their knowledge and guidance provided great help to my thesis.

I would also like to thank all the colleagues, especially the tea group, at Eonic. Their company brought me relief at those days of work at Eonic.

I want thank my parents for supporting my study, both financially and emotionally. Special thanks to my girlfriend Ping Lu for her support and endless love.

Again, I thank you all.

Meng Ma
Delft, The Netherlands
June 13, 2005

1 Introduction

1.1 Background

Image registration is always a hot topic for many applications. Such technologies have been widely used in face detection, object location in satellite images, virtual vision, medical applications etc. Although the development of modern computers becomes amazingly faster and faster, general purpose computers still can not apply those applications efficiently since they require large amount of calculations, especially when the quality issue is emphasized. Specially designed hardware for certain applications can be very efficient in both aspects of quality and performance. In this thesis work, a solution which can effectively, in real-time, locate known object(s) with unknown rotation and scaling in streaming pictures is introduced. This method can be subdivided into several parts of which two will be dealt with in this thesis in particular. A phase normalizer of a Symmetric Phase Only Matched Filter and a self-adaptive peak detection algorithm with a removal of adjacent peaks are developed and implemented. The whole system is based on a four-board PowerFFT platform [1].

1.2 Rotation and Scale Invariant Template Matching

Symmetric Phase Only Matched Filter (SPOMF) has proved to be a reliable method for detection of a known object in an image. To detect an object in this image (search image), a template image which contains the object itself should be prepared. Both the template image and search image will then be transformed to the frequency domain. In the frequency domain, the spectral components of search image and template image are element wise conjugated multiplied. This operation actually calculates the phase differences between spectral points in the search image and the corresponding spectral points in the template image. All the vectors in the resulting matrix of the multiplication will be normalized to 1. Then, the resulting matrix will be converted back to spatial domain and the peaks that exceed the threshold indicate the found location of this object in the search image.

SPOMF works well for exact matches even when noise is present or only part of the object is visible. It fails when the match can not be obtained by a translation and scaling and rotation are added. In that case, the rotation angle and scaling factor should be first detected and then the template image can be compensated for. After that, regular SPOMF can be used to detect the location(s) of object(s). To detect the rotation angle and scaling factor, both the search image and template image will be converted to the frequency domain by the PowerFFT platform. In this domain a rotation remains a rotation and scaling of the spatial domain is the inverse scaling of the frequency domain. The coordinate system of the images should then be converted

from rectangular coordinate system to a polar coordinate system. In a polar coordinate system, the x axis represents the angle and the y axis represents the radius. To convert a rotation and scale to a simple translation that can be detected with SPOMF, the logarithm of the radius is taken. This way the multiplication of the scaling becomes and addition in logarithmic coordinates. The rotation in the spatial domain remains rotation in the frequency domain and the scale remains scale (inverse direction). And the translation information is removed by taking the absolute value of the spectrum points. By applying SPOMF to the spectrum in the log-polar coordinate system, the rotation angle and scaling factor can be detected.

2 RSI Project Description

Real-time rotation and scale invariant (RSI) template matching in streaming images can be implemented on a reconfigurable and scalable platform based on PowerFFT and FPGA hardware. For images with resolution of 512 by 512, if there are not many detections, a frame rate of up to 18 frames per second can be reached on a four-board platform.

2.1 Theory

Image translation between two images is found perfectly with SPOMF even the noise is added to the background or to the object itself. But when it tries to find location for rotated and/or scaled object, the SPOMF becomes inefficient. So the focus is on finding rotation and scale, because if those two factors are known and compensated for, the problem is solved. To get rid of translation a transformation to a translation invariant domain is done.

Translation, rotation, and scale all have their counterpart in the Fourier domain. What we should do is to separate the spectrum in a certain way so that the information can be abstracted from the spectrum. From mathematical assistance (which will be given below), we found out that the translation information is located only in the phase (more precisely, phase difference between adjacent spectral components) of the spectrum. By conjugated multiplying the two spectra and normalizing the magnitude of the resulting numbers in the frequency domain, the translation information can be abstracted. For rotation and scale, things are more complicated because those parts of information are not only located in the magnitude of the spectrum but also partly located in the phase. See figure 2.1.

In order to take out the translation information in the phase and keep only the rotation and scale information, we can differentiate horizontally and vertically to get rid of the translation information but remain the rotation and scale information in the phase. Let $F(\xi, \eta)$ be the frequency representation of the template image and let $F(\xi, \eta) \bullet e^{j\theta(\xi x_0 + \eta y_0)}$ be the correlated frequency representation of the search image. After conjugated multiplication and normalization, only the $e^{j\theta(\xi x_0 + \eta y_0)}$ remains. See the differentiation operation below, the left part is the matrix before differentiation and the right part is afterwards. Because of the differentiation, the translation information in the exponent disappeared, but the $\theta(\xi x_0 + \eta y_0)$ function also changed to $\theta'(\xi, \eta)$. That means we also modified the rotation and scale information when applying the differentiation.

$$e^{j\theta(\xi x_0 + \eta y_0)} \Longrightarrow e^{j\theta'(\xi,\eta)}$$

In order to compensate for that, we have to integrate along the rotated x axis and y axis. But we don't know the rotation angle yet because the purpose is to detect the

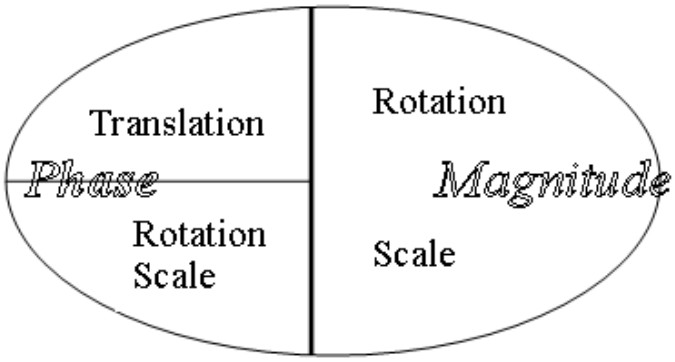


Figure 2.1: Image information distribution in frequency domain

rotation and scale! So this solution is not feasible although it's theoretically right. What we can do is throw away the rotation and scale information in the phase and take the absolute value of the complex number (the magnitude) to detect the rotation and scale. Experiments show that it still provides acceptable quality for utilities.

- **Translation**

  The translation of two images in the spatial domain is represented only by phase difference in the frequency domain. Let f_1 and f_2 be the two images that differ by a displacement of (x_0, y_0), the two images are related like this:

  $$f_2(x, y) = f_1(x - x_0, y - y_0)$$

  Their Fourier transform will then be related like this:

  $$F_2(\xi, \eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} \bullet F_1(\xi, \eta)$$

  From the above equation, it is clear that image translation in the frequency domain only matters in linear phase change because all the displacement parameters only affect the phase of the translated image in frequency domain. The phase difference can be calculated by elemental wise multiplying the spectrum of the search image by the complex conjugated spectrum of the template image or the other way around.

- **Rotation**

  Rotation in the spatial domain still results in a rotation in the frequency domain. Let $R(\theta)$ be the rotation in the spatial domain. It can be written as a 2D rotation matrix as below:

  $$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Assume the translation still remains as (x_0, y_0) , the two images are related as:

$$f_2(x, y) = f_1(x\cos\theta + y\sin\theta - x_0, -x\sin\theta + y\cos\theta - y_0)$$

After being transformed to Fourier domain, we obtain:

$$F_2(\xi, \eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} \bullet F_1(\xi\cos\theta + \eta\sin\theta, -\xi\sin\theta + \eta\cos\theta)$$

Notice that the spectrum of f_2 is actually the rotated spectrum of f_1 if the phase is taken away.

- **Scale**

  Scale in the spatial remains scale in the frequency domain but in a inverse direction. Scale factor(s) can be calculated by converting the coordinate system of the frequency domain to logarithmic scale. Let f_1 be the scaled replica of f_2 with scale factor (a, b) which stands for horizontally scale and vertically scale. The relation of f_1 and f_2 in the frequency domain is like:

  $$F_2(\xi, \eta) = \frac{1}{|ab|} F_1(\xi/a, \eta/b)$$

  Once the coordinate system is converted to logarithmic scale, scaling can be reduced to translation movement which can be caught by regular SPOMF later. If we ignore the multiplication factor $1/ab$. See below:

  $$F_2(\log\xi, \log\eta) = F_1(\log\xi - \log a, \log\eta - \log b)$$

  Then the scale factors can be easily obtained.

- **Translation, rotation and scale in log-polar representation**

  If there are translation, rotation and scale in the same search image, it is still possible to obtain those factors. The only limitation is that there should be only one scale factor instead of two in horizontal and vertical directions. That means the scale factors in horizontal and vertical direction should be the same, otherwise no rotation can be found.

  Since the rotation in the spatial domain still remains rotation in the frequency domain and the scale in the spatial domain can be converted to a shift in the frequency domain if the axes are mapped to logarithmic scale, it is convenient to map the original coordinate system to a log-polar coordinate system. In log-polar coordinates, the rotation angle is represented on the θ axis and the scale is represented on the ρ axis. Let θ_0 be the rotation angle and a be the scaling factor (notice that there is only one scaling factor). If we consider the magnitude spectra of the Fourier Transform of the two images, they are related as:

$$M_1(\log \rho, \theta) = M_2(\log \rho - \log a, \theta - \theta_0)$$

Here rotation is a shift on the θ axis and scale is a shift on the ρ axis [2].

The main algorithm schematic is shown in figure 2.2. The search image and the template image will be 2D Fourier transformed to a complex number matrix. As we explained before, there is no nice way to get rid of the translation information in the phase without changing the rotation and scale information, we have to take the absolute value to get the rotation and scale. Then, the images in the frequency domain are mapped to log-polar coordinates so the rotation and scale become translations on the angle axis and the radius axis. Those translations can be calculated by applying SPOMF to them and then rotation and scale factors are determined. They are used as parameters for a rotation and scale unit to rotate and scale the template image to the opposite directions. The modified template image now has the exact same rotation angle and scale factor as the search image. By applying the SPOMF again, the actual position(s) of the object in search image will then be located [1].

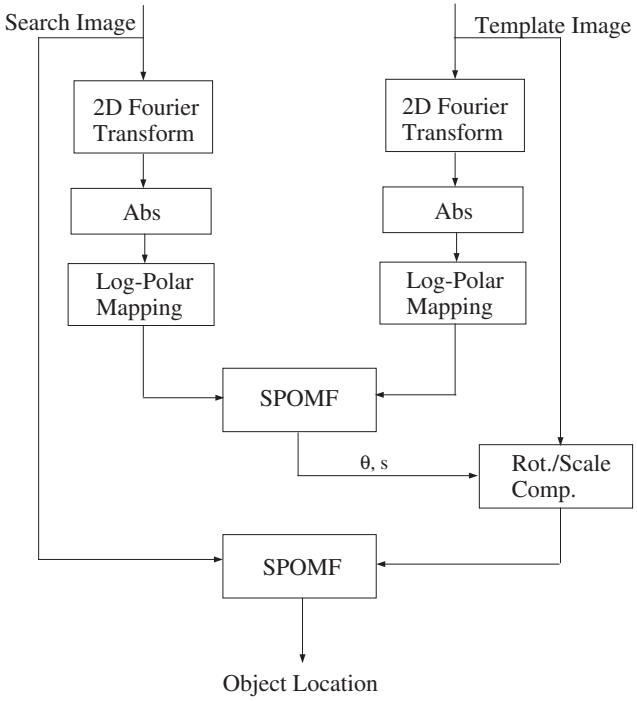


Figure 2.2: Schematic of RSI matching

2.2 Symmetric Phase Only Matched Filter

The shift information of the search image with respect to the template image is located in the phase difference of the two spectrums. Phase difference can be calculated by conjugated multiplying two spectrums. The resulting spectrum is then inverse Fourier transformed to the spatial domain. Since multiplication in the frequency domain is convolution in spatial domain, the transformed matrix has a peak in the location of the object because the energy has been summed up by convolution. The rest area of this matrix has approximately zero value because the summing up averages the value to zero if there is no shift existing.

Before inverse Fourier transform, it is necessary to normalize the radius of complex numbers to 1. The reason of doing so is that high brightness in the image causes peaks after inverse FFT. This will be discussed in detail in chapter 3.

After inverse FFT, there will be a peak in the object location. But the magnitude of the peaks in different images has a different value. The range is large enough to fail any static threshold. So a floating threshold algorithm is required to effectively detect the peak(s). The SPOMF not only finds a match in the exact location of the object, but also collect high energy in the surrounding area of the object. For that case, an algorithm that chooses the most likely peaks is developed. Details will be given in chapter 4.

2.3 Log-Polar Mapping and Interpolation

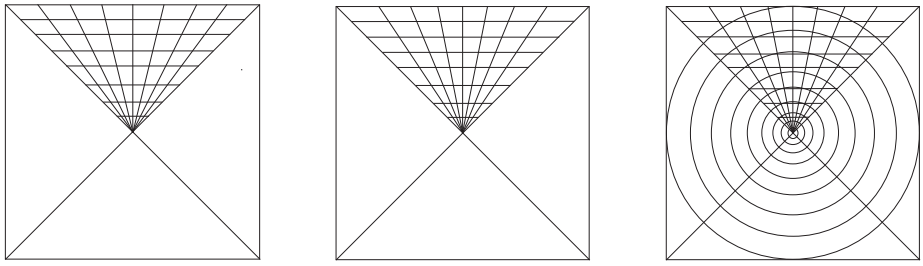
The performance of RSI template matching algorithm greatly depends on the quality of log-polar mapping of the image spectrums. If too much distortion is introduced during the interpolation of log-polar mapping procedure, it will be difficult for SPOMF to detect rotation and scale.

To map the image spectrum, one solution is to map the spectrum using 2D interpolation. Unfortunately the 2D spectrum is a highly oscillatory function that is hard to interpolate. Also, another problem is that the closer to the center of the spectrum, the denser the grid of the map is. In fact, this method introduces too many interpolation errors to be able to detect any rotation and scale. Matlab simulations using spline interpolations have shown that with floating point precision, this method still can not provide any guaranteed match [1].

Another solution which is called Polar-FFT [3] calculates the log-polar spectrum immediately from the original image in spatial domain. This algorithm provides better quality because of better interpolation. The algorithm splits the image into four wigs - top, bottom, left and right. Since the spectrum is symmetrical, only the top (or bottom) and left (or right) wigs are calculated.

For the top wig, first all the columns of the image are 1D Fourier transformed. And then Chirp-Z Transform is applied to the first row, calculating the spectrum from $-\pi$ to π. For the second row, the Chirp-Z Transform calculates the same amount of points, but with a smaller frequency range. See figure 2.3(a). Now every point in each row has the same distance to its neighbors but the angles between radiuses are not the same. The next step is to apply a 1D interpolation to each row to adjust the distances so that the angles between neighboring radiuses become the same. See figure 2.3(b). The final

step is to apply another 1D interpolation to the radiuses to fit the circle (figure 2.3(c)). The twice 1D interpolation would bring some errors and lower the quality of log-polar mapping, but the experiments show that those errors are still acceptable [1].



(a) (b) (c)

Figure 2.3: Polar FFT

2.4 Rotation and Scale

To compensate for known rotation and scale factor by rotating and scale the template image is also a critical point to the overall performance. Therefore, an interpolation algorithm providing high quality and reliability is necessary. Rotation and scale using 2D interpolation decrease the image quality too much so that detecting the object becomes too difficult.

Michael Unser presented a convolution-based interpolation for rotation using 1D convolutions based on FFT operations only. This method provides high image quality even after a great number of rotations [4].

Scale in spatial domain can be mapped to an inverse scaling in frequency, meaning that a scale up operation in an image actually causes a scale down in frequency. That makes a one on one mapping between image scale and frequency scale possible. By using zoomed-in or zoomed-out Chirp-Z Transform, scale down or scale up and be applied.

2.5 Reconfigurable Platform

The whole algorithm is implemented on a platform with four PowerFFT boards. The PowerFFT board includes a PowerFFT processor which has a 57 bit datapath (9 bits of exponent and 24 bits mantissa for real and imaginary number) and two FPGAs. Based on the dataflow schematic in figure 2.2, the system is built in a way like figure 2.4.

The log-polar mapping of the template image can be pre-calculated and stored in memory. The log-polar mapping of the search image uses one board. The conjugate multiplication can be done in the PowerFFT processor since it also contains a stand alone multiplier while the detection algorithm can be implemented on the FPGA so the SPOMF and the detection unit together use one board. The rotation and scale that use

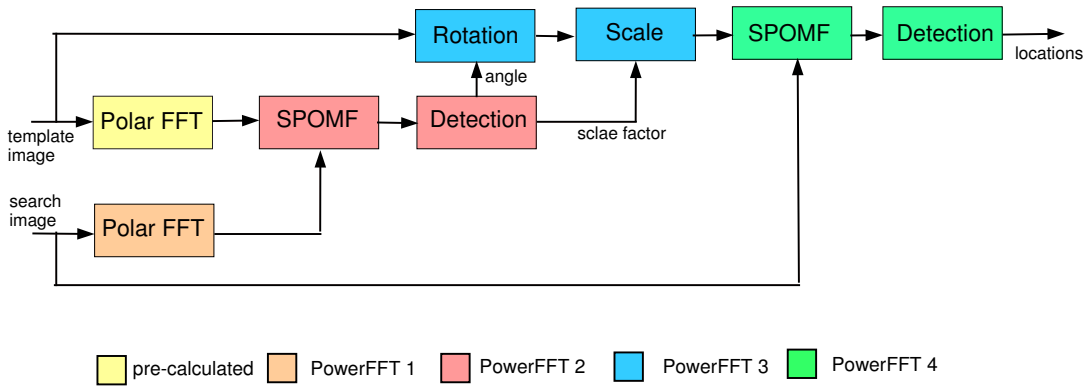


Figure 2.4: Four board architecture of RSI

Chirp-Z Transform also use another board. Finally, the SPOMF and detection for the object location needs a board.

3 Phase Normalization

Symmetric Phase Only Matched Filter has been proved a reliable method to locate the position of a known object. The typical dataflow of such an algorithm is shown in Figure 3.1.

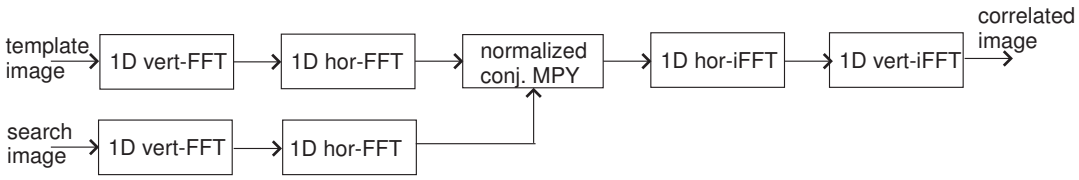


Figure 3.1: SPOMF algorithm dataflow

Both the search image and the template image are converted to the frequency domain and are conjugated multiplied. Since all the translation information is located in the phase of the resulting complex matrix, we need to normalize the magnitude of all the complex number to 1 so that only the phase information is kept. In this chapter, different normalization algorithms will be discussed and compared, and finally a simple algorithm with high performance is developed based on the experimental results.

3.1 Why Normalization

Since a multiplication in the frequency domain is a convolution in spatial (time) domain, the conjugated multiplication in the frequency domain causes pixels in the images do be convoluted. That means that the pixel value of the correlated image at a certain position is the sum of all the pixel multiplication results when pixel-wise shifting one of the images by the same position steps.

The ideal situation is that both the search image and template image have black background and there is only one object in each image but with different locations. Also assume that the object in the template image is located in the top left area, meaning that the top left pixel of the object is also the top left pixel of the template image. Since the black color is normally represented as zero in digital images, the multiplications by zeros result zeros in correlated image. Only when the two objects are matched because of the shifting, there will be positive values in the correlated image (Notice that partly matching also results positive value but with lower magnitude). The peak with positive values in the correlated image actually indicates the location of the object in the search image.

In practice, the background is not always ideal. The multiplication of none-zero pixels will also show positive values in faulty area of the correlated image. In some bad cases, there is white area in the search image or in the template image which makes the magnitude of this area in the correlated image even higher than the location peak. That makes the detection of the correct peak impossible.

Figure 3.2 gives an example of such case. Picture 3.2(a) is the search image and 3.2(b) is the template image. The task is to find the location of the template car in the search image using SPOMF. Picture 3.2(c) is the correlated image with normalization while 3.2(d) is the one without.

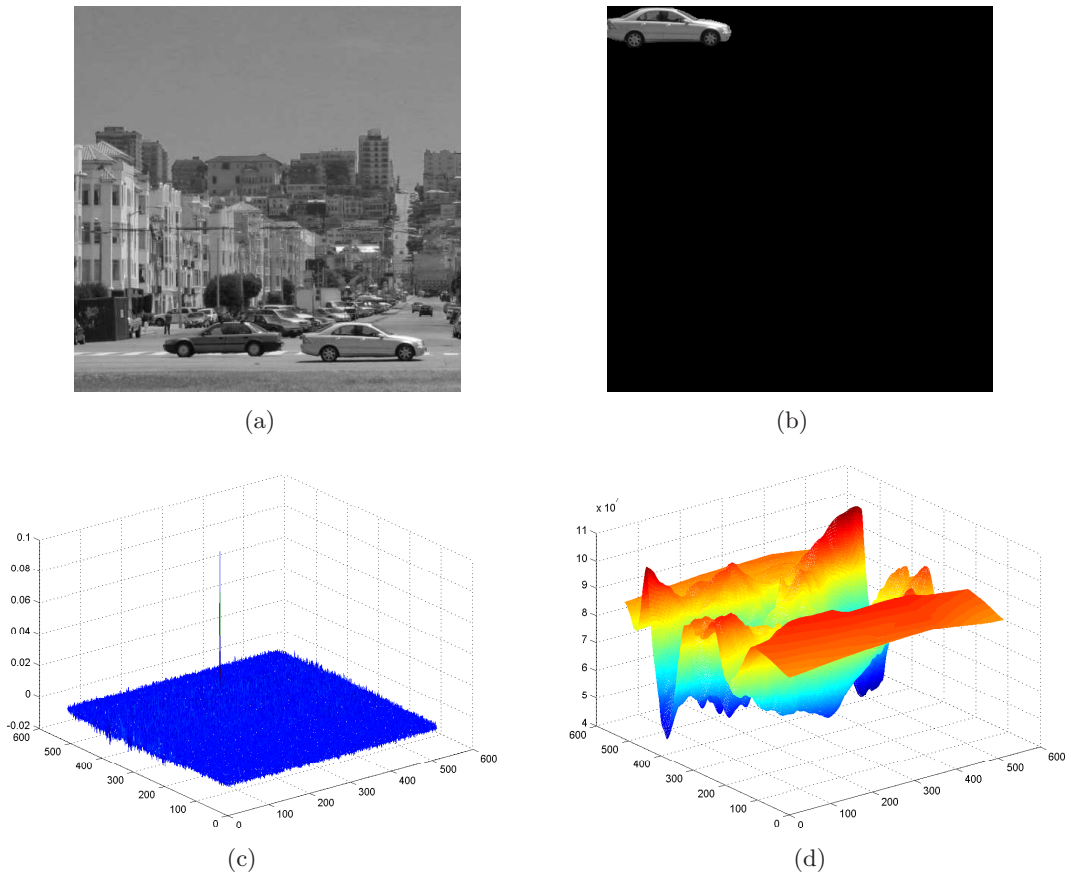


Figure 3.2: SPOMF demonstration with and without normalization

From the example we can see that the correlated image with normalization gives a nice and clean peak at the same location of the car in the search image (301, 401). But the correlated image without normalization gives no peak at all.

3.2 Different Solutions

3.2.1 Normal Solution

The intuitive method of normalizing complex numbers is to divide the radius by itself. For instance, to normalize a complex number $x + yi$, we need first calculate the vector radius: $\sqrt{x^2 + y^2}$, then divide x and y by this number, the normalized the number is $x/\sqrt{x^2 + y^2} + (y/\sqrt{x^2 + y^2})i$. See figure 3.3, here the θ is the phase of the complex number. This method involves square, square root and division which are costly and slow for hardware implementation.

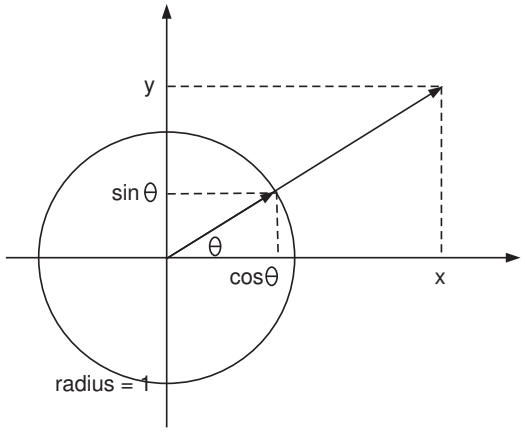


Figure 3.3: Phase normalization with normal method

3.2.2 Lookup Table Solution

In order to avoid square and square-root unit, a look up table is an option. Since the normalized the value is actually $\sin\theta$ and $\cos\theta$, as shown in figure 3.3, we can establish the lookup table based on the θ. Here, we don't have to really calculate the value of θ, but use the ratio of x and y, in fact it is $\tan\theta$, as the index of the table. In that case, it needs a match unit to find the closest matching in the index for the input. See figure 3.4.

3.2.3 CORDIC Solution

The CORDIC (Coordinate Rotation Digital Computer) algorithm was introduced in 1959 by Volder. In 1971, Walther generalized this algorithm to compute logarithms, exponentials, and square roots. CORDIC works by rotating the coordinate system through constant angles until the angle is reduced to zero. The angle offsets are selected such that the operations on x and y are only shifts and adds [5]. This method can be used here to first rotate the vector to x-axis, normalize to 1 and then rotate back to the original angle. See figure 3.5.

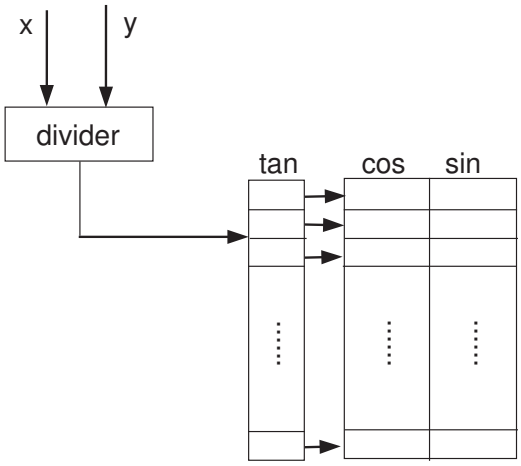


Figure 3.4: Phase normalization with lookup table

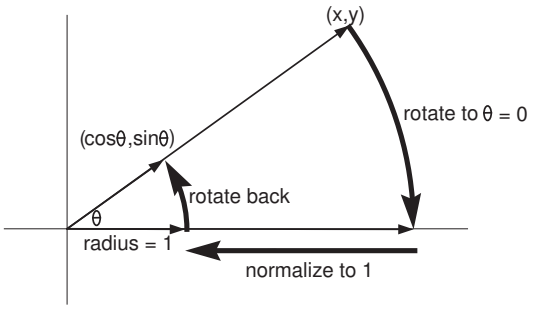


Figure 3.5: Phase normalization with CORDIC

3.3 The CORDIC Solution

An advantage of CORDIC compared to other solutions is that it only uses shifts and additions. It can be efficiently implemented on hardware without complex units. Besides, each rotation step in CORDIC is quite similar so that they can be pipelined to reach a high performance.

3.3.1 The Theory

A planar vector rotation from (x, y) to (x', y') in a 2D coordinate system can be defined as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

A single rotation can be divided into multiple small rotation steps. Each step rotates the vector by a small angle. By iteratively complete those small rotations, the full rotation can be reached. The small rotation is defined as below:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

If we take out the factor of $\cos\theta$ from this equation, we get:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

This equation still contains several multiplications. If we choose the rotation angles very nicely so that each $\tan\theta$ is a power of 2, then the multiplications become shift operations. That means: $\theta_n = \arctan\left(\frac{1}{2^n}\right)$ while all iteration angles summed up must be equal to the rotation angle θ, meaning $\sum_{n=0}^{\infty} S_n\theta_n = \theta$, where $S_n = \{-1; 1\}$ which represents the rotation direction. Then the only thing left is the $\cos\theta_n$ factor. Since the angle of each step is known, each $\cos\theta_n$ factor is actually a constant: $\cos\theta_n = \cos(\arctan(\frac{1}{2^n}))$. If the rotation is completed by N steps, the scale factor $K = \frac{1}{P} = \prod_{n=0}^{N} \cos(\arctan(\frac{1}{2^n}))$. When N is a large number, $P \approx 1.6468$, $K \approx 0.607253$ [6].

The rotation now becomes only shift operations and additions. It can be written as:

$$\begin{aligned} x_{n+1} &= x_n - S_n 2^{-2n} y_n \\ y_{n+1} &= y_n - S_n 2^{-2n} x_n, \qquad where S_n = \begin{cases} -1 & if \quad z_n < 0 \\ +1 & if \quad z_n \geq 0 \end{cases} \\ z_{n+1} &= \theta - \sum_{i=0}^{n} \theta_i \end{aligned}$$

The scale factor K is pre-calculated and can be taken into consideration in early or later stage.

3.3.2 Normalization with CORDIC

To rotate a complex number to the real axis, we can use a simplified version of CORDIC. Because the rotation direction coefficients S_n is fully depends on the sign of imaginary part, the calculation of is not necessary. Since we also want to rotate the vector back to the original angle, the rotation direction coefficients can be reused (notice that the coefficients should be inversed before using) in the rotating back operation. A simple model of CORDIC solution is given in the figure 3.6.

The input of the rotate back unit is always the same for every vector. Notice that the real part of the initial vector is 0.6073. This is the scale factor K that has been mentioned in the theory section. The rotate back unit only needs the direction coefficients from the sign bit of the imaginary part so the two rotate units can work concurrently.

The number of iteration steps for the CORDIC algorithm is basically determined by the representing bit number of the input x and y. The reason of this is that we set every

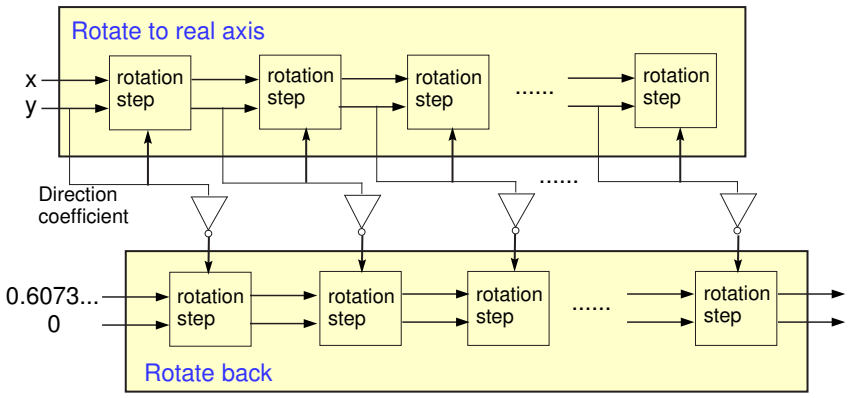


Figure 3.6: Schematic of CORDIC normalization

$\tan\theta_n$ to 2^{-n}, meaning that every step, the $\tan\theta_n$ becomes half of the previous value. And the curve of $\tan\theta$ can be considered as a linear function of $f(x) = x$ in the region near zero. That results in the fact that the θ_n itself also approximately becomes half in every step. Table 3.1 gives an example of 10 bits precision to demonstrate the linear characteristic of $\tan\theta_n$.

i	2^{-i}	$\arctan(2^{-i})$
0	1.0000000000	0.1100100100
1	0.1000000000	0.0111011011
2	0.0100000000	0.0011111011
3	0.0010000000	0.0001111111
4	0.0001000000	0.0001000000
5	0.0000100000	0.0000100000
6	0.0000010000	0.0000010000
7	0.0000001000	0.0000001000
8	0.0000000100	0.0000000100
9	0.0000000010	0.0000000010
10	0.0000000001	0.0000000001

Table 3.1: The value of $\arctan(2^{-i})$ with 10-bit precision

3.3.3 Lookup table and CORDIC hybrid solution

Another way to obtain the direction coefficients is to remember them in the lookup table since there is large memory space in the FPGA chip. The index of this table consists of several most significant bits of x and y, while the content of the table is the direction coefficient sequence. See figure 3.7. Both the real part and the imaginary part of the complex number are represented by a floating point number with 24 bits for the mantissa

and 9bits for the common exponent. Therefore, it is impossible to establish this lookup table by full accuracy since it results a table with 2^{48} entries.

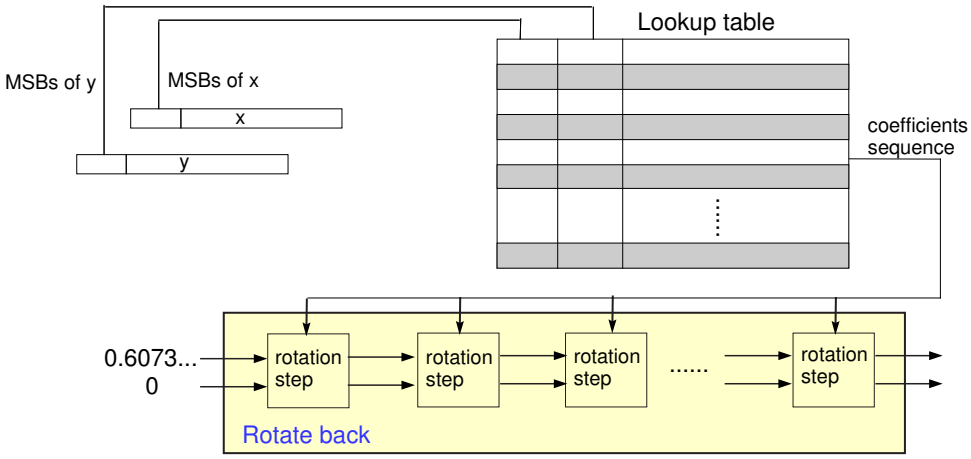


Figure 3.7: A hybrid solution of lookup table and CORDIC

Figure 3.8 gives a MatLab simulation of applying phase normalization using the full CORDIC and the hybrid solution. The search image is a mig-25 fighter in a noisy background with a high brightness area in the top left corner to test the normalization. The template image is the same mig-25 located in top left corner of the image. Figure 3.8(c) shows the correlated image after SPOMF with the full CORDIC operation while figure 3.8(d) shows the correlated image with the hybrid solution which takes the 5 most significant bits of x and y. With 5 bits of x and 5 bits of y, there are $2^{10} = 1024$ entries in the table. And the coefficients sequence contains 24 bits since there are 24 rotation steps for the CORDIC. The total size of the table is then 4352 bytes. Compared to full CORDIC, the hybrid solution has no obvious weakness. The magnitude of the peak is 99.96% of the peak in the full CORDIC and the noise level barely changes. Tests for other images also give almost the same good results. Those tests show that using few bits to represent the phase still provides good detection quality. Then, does it work with even fewer bits?

3.4 Reduced Bit Number for Phase Representation

The operation of taking MSBs of x and y actually rounds the phase to a less accurate digital representation, though the CORDIC rotation is accurate. That means that the CORDIC rotation can only rotate the vector to the rounded the phase. In that case, there is no point to have such an accurate operation after an inaccurate one. Therefore, we can totally remove the CORDIC from the algorithm and use only the lookup table. The index of the table can still be the MSBs of x and y while the contents of the table should be modified to normalized x and y with 24 bits precision.

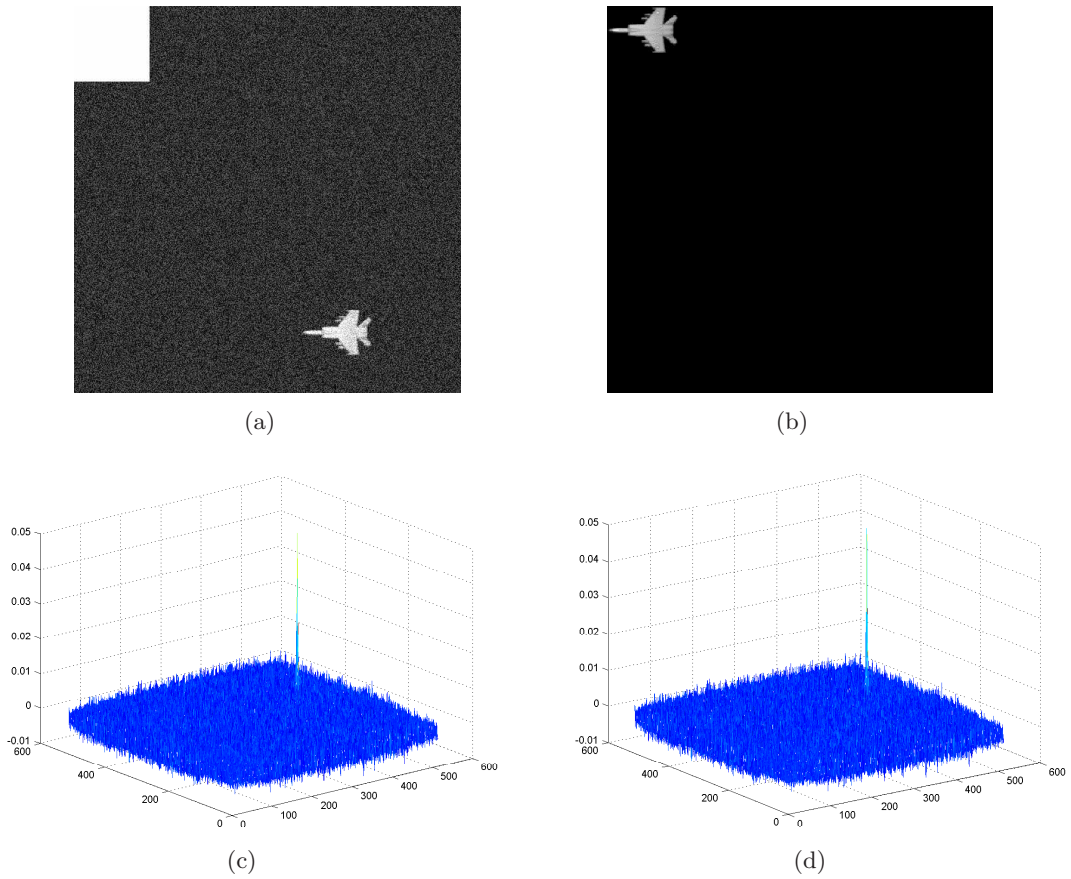


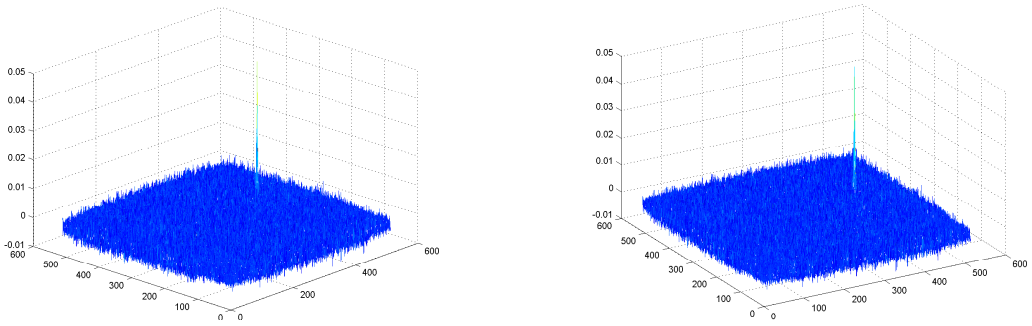
Figure 3.8: Comparison of full CORDIC solution and hybrid solution

3.4.1 Sign Bit Only (SBO) solution

In the previous section, the experimental results show that using 5 MSBs to represent x and y works well. In fact, even using fewer bits, the result can still be good enough to detect the peak. Figure 3.9 gives some examples of SPOMF using fewer bits representation. Figure 3.9(a) shows the correlated image of the mig-25 fighter in 4 bits phase representation and figure 3.9(b) shows the 3 bits phase.

Since using 4 bits or even 3 bits can still provide very good results, it is natural to try the extreme case: only 1 bit. In that case, only the sign bit of x and y are taken into consideration. For instance, if the sign bit of x is 1 (negative) and the sign bit of y is 0 (positive), the output angle is 135 degree, meaning that the normalized complex number is $-\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}i$. In fact, there are totally four possible output numbers of the normalization operation, they are $\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}i$, $-\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}i$, $-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}i$ and $\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}i$ whatever the input is.

In such extreme case, the correlated image after SPOMF still gives almost the same quality as high accuracy representation or even as the full CORDIC. MatLab simulation shows that the magnitude of the peak reduced by approximately 9.96% compared to full



(a) (b)

Figure 3.9: SPOMF with phase representation of fewer bit number

CORDIC for different images. Figure 3.10 gives the same example of the mig-25 fighter image using sign bit only (SBO) normalization. Figure 3.11 gives another example of a more complex image. The search image contains a lot of apples and the template image has an apple in the top left corner. Notice that no scale and rotation compensation is used here but only the regular SPOMF. Figure 3.11(c) is the correlated image of full CORDIC normalization while 3.11(d) uses SBO normalization.

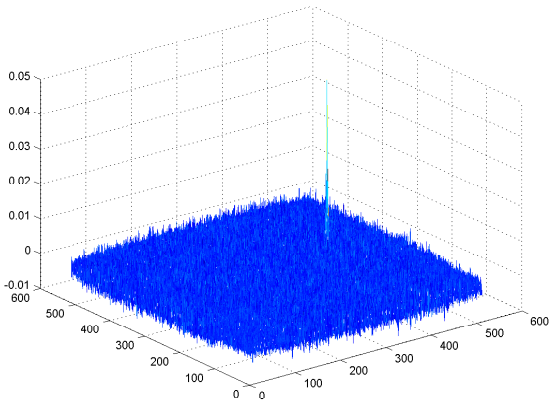


Figure 3.10: SPOMF of mig-25 using sign bit only normalization

3.4.2 Mathematical Explanation

To explain why the SBO works so well, we have to understand the actual mechanism of how phase difference represents the shift information of two images.

After Fourier transform, the search image and the template image in the frequency domain will be conjugate multiplied. In the result, that is a 2 dimensional matrix with complex numbers. The horizontal shift is represented by the number of circles that the phase rotates and the vertical shift, the same as horizontal shift, is represented by the

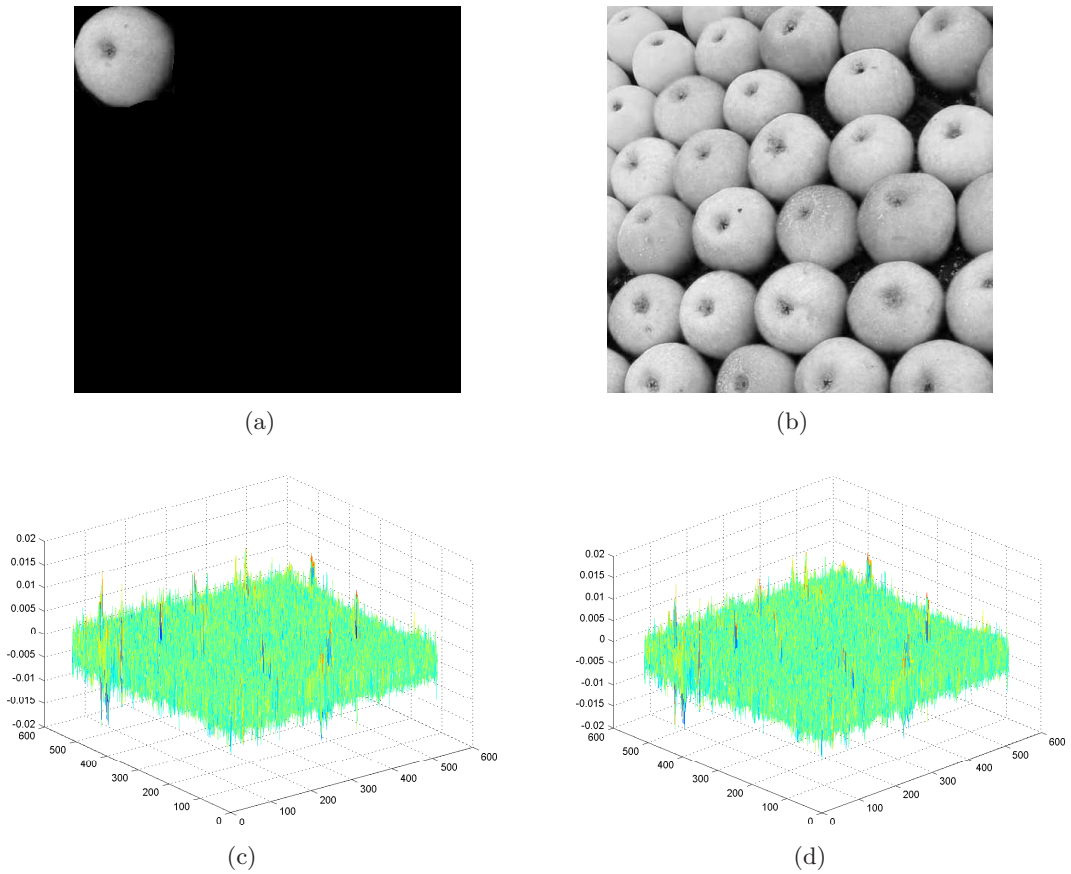


Figure 3.11: Comparison of SPOMFs with full CORDIC and SBO

number of circles that the phase rotates along the vertical direction. Figure 3.12 gives an example of the phase of such a matrix when there is a shift of 5 pixels along the horizontal direction and 9 pixels along the vertical direction. With multiple objects in the search image, the phases that represent multiple translations will be added up. For example, if there are objects located in position (5, 9) and (8, 12), the phase in the horizontal direction will be the sum of a phase function with 5 cycles and another phase function with 8 cycles.

Now, let's use the SBO method to quantize the phase angles to 4 possible angles (45, 135, 225 and 315 degree). From figure 3.13, we can see that although some individual phases lose large amount of information from their original values, the overall characteristic of all the phases is still kept well. The total number of circles that the phase has rotated can barely change from the quantization.

The SBO normalization can also be mathematically proved to be reliable. Let $F(\xi, \eta)$ be the frequency domain representation of the template images. Then the search image which differs by a displacement of (x_0, y_0) can be represented by $F(\xi, \eta) \bullet e^{-j2\pi(\xi x_0 + \eta y_0)}$. After conjugated multiplication, the correlated matrix is like:

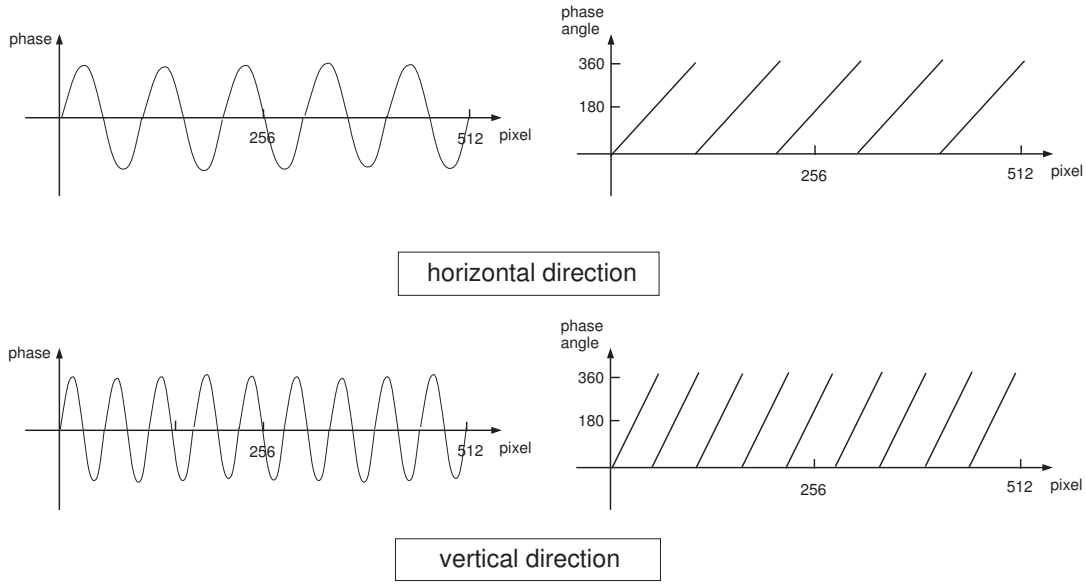


Figure 3.12: Shift information represented by phase rotation circles

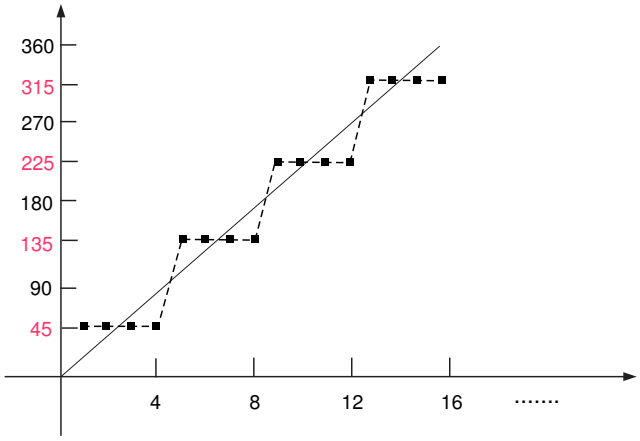


Figure 3.13: The well kept overall characteristic with SBO normalization

$$F(\xi,\eta) \bullet \overline{F(\xi,\eta)} \bullet e^{-j2\pi(\xi x_0+\eta y_0)} \Longrightarrow |F(\xi,\eta)|^2 \bullet e^{-j2\pi(\xi x_0+\eta y_0)}$$

The non-quantized normalization normalizes the $|F(\xi,\eta)|^2$ factor to 1, so the correlated matrix becomes only $e^{-j2\pi(\xi x_0+\eta y_0)}$. Let $Err(\xi,\eta)$ be the error function in the frequency domain caused by the phase quantization. Then the correlated matrix becomes $e^{-j2\pi(\xi x_0+\eta y_0)+Err(\xi,\eta)}$ after SBO normalization. It can be converted to

$e^{-j2\pi(\xi x_0+\eta y_0)} \bullet e^{Err(\xi,\eta)}$. Note that a multiplication in the frequency domain is a convolution in the spatial domain. So if we take the inverse FFT to the correlated matrix, it becomes:

$$IFFT(e^{-j2\pi(\xi x_0+\eta y_0)}) * IFFT(e^{Err(\xi,\eta)})$$

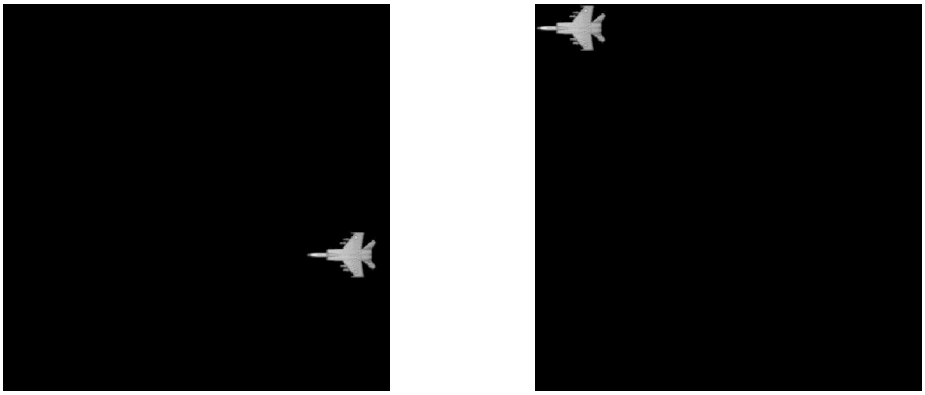
The left term in the above correlated image (note that we use correlated matrix for the frequency domain and correlated image for the spatial domain) is the correct peak. So let us see what happens when the correct peak is convoluted to the right term. When we quantize the phase to 4 angles using SBO normalization, the distortion is no more than $\pi/4$. If the errors are random distribution within the range of $-\pi/4$ to $\pi/4$, the inverse FFT of this error function is a strong DC component plus some low level noise. The convolution of the correct peak and the DC component still gives a peak in the correct position. Since the magnitude of the DC component is slightly less than 1, the magnitude of the convoluted peak will also be reduced. The more bits we use to represent the phase, the higher the convoluted peak is. Table 3.2 gives a comparison of different peaks' magnitude under different bit number representations.

	SBO	2 bits	3 bits	4 bits	5 bits
Peak magnitude percentages	90.04%	97.45%	99.36%	99.84%	99.96%
Noise level increments	0.076%	0.036%	0.019%	0.010%	0.005%

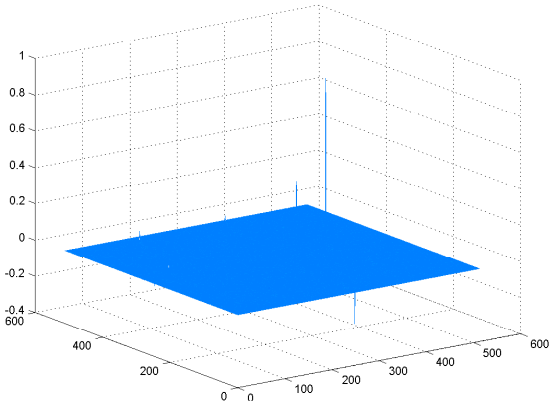
Table 3.2: Peaks' magnitude and noise levels under different bit number representations

In the above description, we assume that the error function $Err(\xi, \eta)$ is a random distribution in the range of $-\pi/4$ to $\pi/4$. For most of the applications, this model is quite close to the real error distribution because either the the noise in the image or the background information can cause the phase angle fluctuates along the linear model in the right part of figure 3.12. But in some cases when the background is very clean and the noise level is very low, meaning very nice matches, the quantization of the phase angle will be very regular like figure 3.13. This stairs-shape regularity in the frequency domain is actually a regular pulse which can cause some peaks other than the correct one in the spatial domain after inverse FFT. See figure 3.14. Both the search image and the template image have clean background. We can see that the correlated image has some faulty small peaks which actually indicate the regular pulse in the frequency domain.

Such a nice match does not happen frequently in practice. The log-polar mapping, the compensating rotation and scale all introduce interpolation errors. Experiments show that even small errors can remove this regular pulse. If there is no rotation and scale in the search image, background information will also introduce irregularity, which removes the faulty peaks eventually. The only chance, which seldom happens, is that both the search image and the template image have a clean background without rotation and scale. But still, the peak detection algorithm can remove those faulty peaks in a later stage. It is safe to say that the SBO normalization is a reliable algorithm for this application.



(a) (b)



(c)

Figure 3.14: Faulty peaks in the clean match

3.5 Hardware Implementation

The hardware implementation of the SBO is quite simple. It fetches the sign bit of both x and y and uses it as the select signals of the two multiplexers which choose the mantissa part of either $\frac{\sqrt{2}}{2}$ or $-\frac{\sqrt{2}}{2}$. The output of the normalizer are hybrid floating point numbers with fixed exponent of "-23".

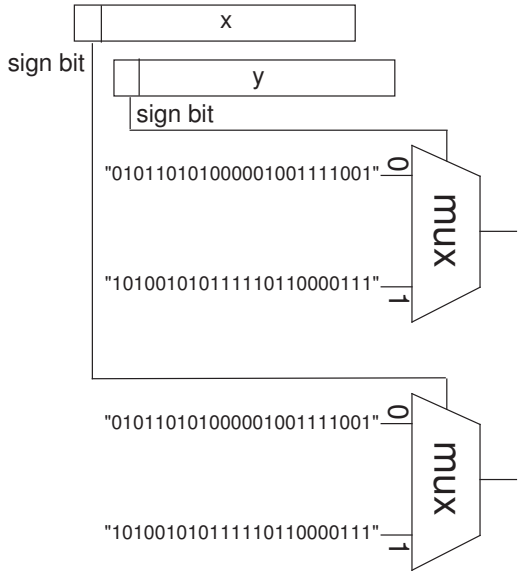


Figure 3.15: Normalizer hardware

4 Peak Detection

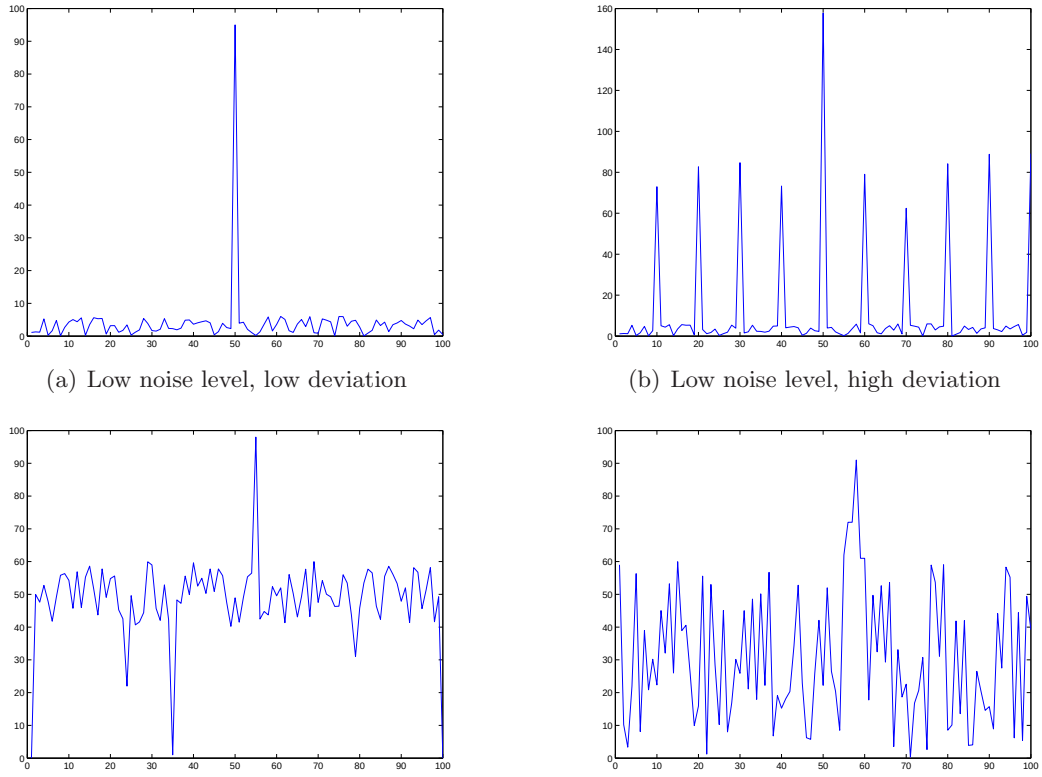
After the peaks have been generated by SPOMF, the locations of the peaks are unknown. To detect those peaks, a threshold that distinguishes peaks and non-peaks should be calculated. The fact that we should be aware of is that the average values of different correlated images (peak matrices) can be very different due to all kinds of reasons. That means a static threshold is not a good choice for detection peaks. In this chapter, a dynamic threshold determination algorithm is developed and discussed. After that, we will discuss an algorithm that maintains a real-time sorting for two sequenced queues in order to save one scan of the correlated image. Note that the threshold can possibly detect some fake peaks which are adjacent to the real one. In order to get the true position of the object, those fake peaks should be removed. A algorithm that does this job will be discussed.

4.1 A Self-Adaptive Algorithm for Threshold Determination

The peak detection algorithm should be flexible for different cases. It should be working not only for the regular SPOMF that detects translation without rotation and scale but also for SPOMF that detects the rotation and scale factor to compensate for. Many issues like noisy background or pseudo matches would possibly increase the difficulty of peak detection. Especially in rotation and scale detections, interpolation errors introduced by the mapping from the normal rectangular coordinate system to the log-polar coordinate system will increase the noise level of the correlated image. All those issues make the peak magnitude and the noise level of the correlated image have very different values in different images. Therefore, the peak detection algorithm should be adaptive to the changing environment.

For simulation, we can use 1D signal waves to demonstrate the detection results. Different correlated images can be classified to four basic categories. See figure 4.1. Here, case 4.1(a) has a peak with high magnitude in a low noise background, which is the best case for detection. Case 4.1(b) also has a relatively low noise background but the deviation is high since there are lots of faulty peaks. Case 4.1(c) has a high noise level but the level is relatively stable. Case 4.1(d) has both high noise level and high deviation.

The noise level is represented by the average value of all the pixels. When the noise level is high, we should also increase the threshold level to avoid too many faulty peaks being detected. The deviation of all the pixels should also have influence to the threshold level. A high deviation should result in a high threshold. Another important term we should take into consideration is the maximum value in the correlated image (usually



(a) Low noise level, low deviation

(b) Low noise level, high deviation

(c) High noise level, low deviation

(d) High noise level, high deviation

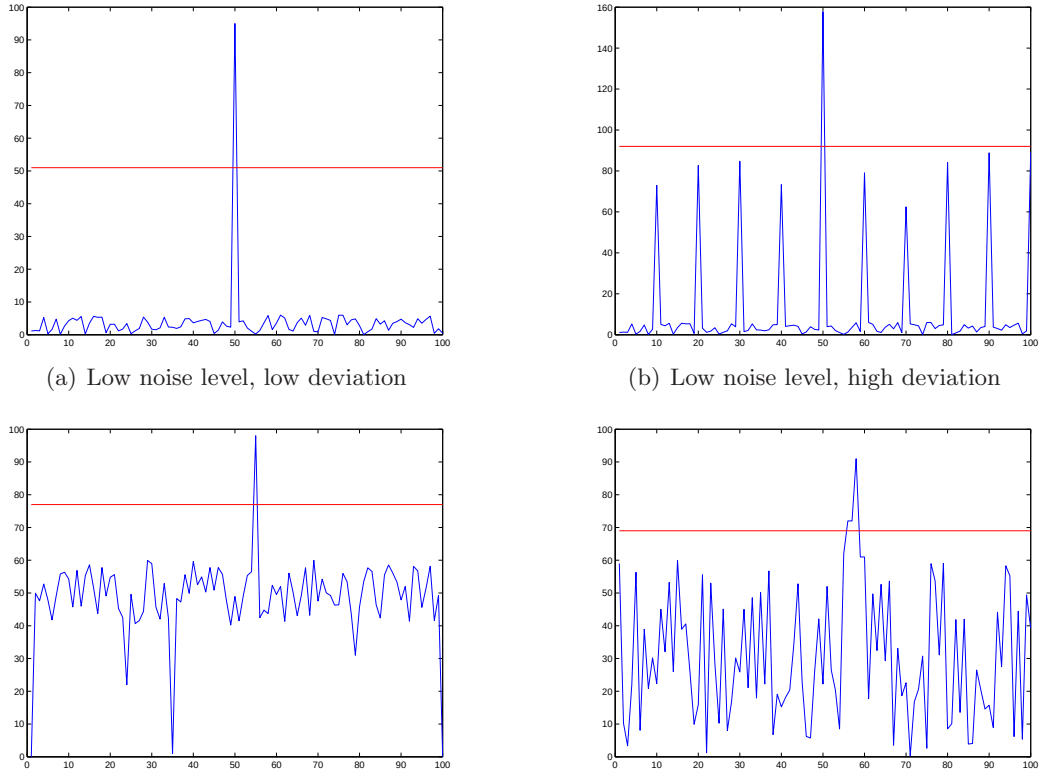
Figure 4.1: Four basic classes of correlated image in 1D representation

the true peak). Notice that the actual average of the correlated image is zero, so we should use the absolute value of the pixels to calculate the absolute average. Based on those intuitive ideas, we can write down the threshold determination equation. Let N be the sample number of correlated image and P_i be the ith pixel.

$$Threshold = (Maximum + Absolute_Average)/2 + K \bullet Deviation$$
$$where:$$
$$Absolute_Average = \frac{\sum_{i=1}^{N} |P_i|}{N}, Deviation = \frac{\sum_{i=1}^{N} |P_i - Average|}{N}$$

Here the K is the influence factor of the deviation. If the deviation is very low, the threshold will nearly be in the middle of the maximum value and the average value. If the deviation is high, the threshold should also be lifted to avoid faulty peaks exceeding it. If the influence factor K is greater than 1, there exists the possibility that the threshold will exceed the maximum value so that no peak can be detected. For example, if the signal in 1D representation is a continuous increment from 1 to 100 (although not likely to happen in practice), the average is 50.5, the deviation is 25 and the maximum is 100. That makes the threshold become 100.25 which is greater than the maximum value.

From the hardware design point of view, it is good to set the K to a power of 2 so that it only needs a shifter to complete the multiplication task. Figure 4.2 shows the same signal waves as figure 4.1 using this threshold determination equation where K is 0.5.



(a) Low noise level, low deviation

(b) Low noise level, high deviation

(c) High noise level, low deviation

(d) High noise level, high deviation

Figure 4.2: Tests of threshold determination algorithm (K = 0.5)

As mentioned before, the true average of the real part of the correlated image is zero. That means the deviation calculation equation can be simplified to:

$$Deviation = \frac{\sum_{i=1}^{N} |P_i|}{N}$$

This equation is the same as the absolute average calculation equation, so the two can be combined. Since both of them are divided by 2 in the threshold determination equation ($K = 0.5$), the combination can remove this division. So the final equation should be like this:

$$Threshold = Maximum/2 + Absolute_Average$$
$$where:$$
$$Absolute_Average = \frac{\sum_{i=1}^{N} |P_i|}{N}$$

Based on this equation, we can build up dataflow architecture to calculate the threshold. See figure 4.3. First, the real part of the complex pixels will be fetched into the system. The ABS unit calculates the absolute value of those numbers. Then an iterative adder will calculate the sum of all those numbers by holding the temporary sum in a register which is controlled by the clock signal. An "end_of_data" signal will trigger the register below the adder to release the sum of all pixels to the next stage. In order to calculate the average of those pixels, we need to divide the sum by the number of pixels. The test images we use are all 512 by 512 format, so we only need a right shifter to shift the sum to the right direction by 18 bits. The calculation of the maximum value is relatively simple. The maximum pixel will be kept in a register and it will be compared to each coming new pixel. If the new pixel is larger than the current one, it will be put into the register to replace the old one. It will also be released by the "end_of_data" signal and then shifted by 1 bit to half it. By adding the absolute average and the half of the maximum pixel, we get the threshold.

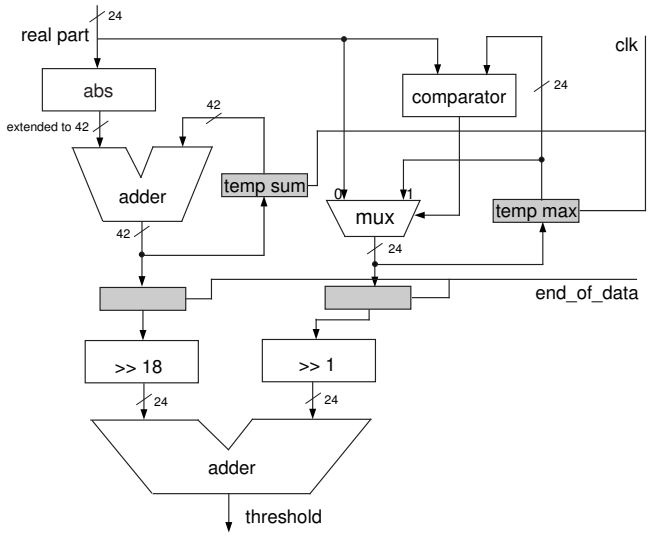


Figure 4.3: Dataflow of threshold determination algorithm

4.2 Sequenced Queues for Maximum Pixels

4.2.1 Sequenced queue

After the threshold has been determined, the peak matrix has to be scanned once again to check which pixels are greater than the threshold and which are not. Those that are greater than the threshold will be recorded as the candidate peaks for further processing. So, totally two scans are necessary here.

This second scan can be saved when the following algorithm is applied. The idea is that we can modify the right part of the dataflow which calculates the maximum value of

all the pixels. Instead of finding the maximum, it records the largest 16 pixels. And these 16 pixels are put into a memory queue in order, meaning they are sequenced from largest to smallest. All the information of the pixels is recorded, including their magnitudes, x positions and y positions. This method is safe to apply because normally there are not so many matches (peaks). After the threshold has been determined, the system looks at the smallest element of this queue. If it is smaller than the threshold, which means all the possible peaks are within the queue, the scan for the whole peak matrix is not necessary. If the smallest element is larger than the threshold, then the system will give a "notice" signal to inform the user that there might be peaks missing.

4.2.2 Binary Sorting

A question of this queue method is how to maintain this sequenced queue. Every clock cycle, a new pixel is input, by that time, the sequenced queue should be prepared for comparison. If the new pixel is larger than the smallest element of the queue, it has to find its position in the queue within one clock cycle. A naïve way is to check all the elements and make comparisons to them until the position is located. This can be improved to a binary search (tree search). The algorithm is shown in figure 4.4.

Such a binary search system requires 5 comparators, one 2 to 1 multiplexer, one 4 to 1 multiplexer and one 8 to 1 multiplexer. When a new pixel comes, it will first be compared to the 15th element of the queue, which is the smallest, and the compared result decides if the following comparisons should be continued. If the new pixel is larger than the 15th element, the queue is going to be updated. Now the question becomes what the position of this pixel should be. In order to find that out, the second comparator compares the new pixel with the 8th (the middle one) element of the queue. The compared result decides which direction the following comparison should go, either smaller direction or larger direction. By doing this operation 4 times, the position is located. Then the new pixel is inserted at that position and all the elements that are smaller than this pixel will be moved to the next position.

Since the comparators are clocked, this method requires 6 clock cycles for a single pixel: 5 cycles for 5 comparisons and 1 cycle for updating the queue. But a new pixel comes at every clock cycle. To solve this problem, a solution is to employ 6 such queues and one queue deals with only 1/6 of the pixels. Each of them maintains its own 16 largest values and finally the system compares all 96 pixels to the threshold.

Another more efficient solution which simultaneously compares the new pixel to all the elements in the queue and only uses two sequenced queues will be given in section 4.2.3.

4.2.3 Parallel Sorting

A faster search method can be obtained in the cost of more comparators. The method is to compare the new coming pixel to all the elements in the queue concurrently. The compared results are either 0 (if less) or 1 (if greater). Since the original queue itself is sorted, the sequence of compared results is either all 0s or a queue of 1s followed by a queue of 0s, which means the new pixel should be put into the position where the 0

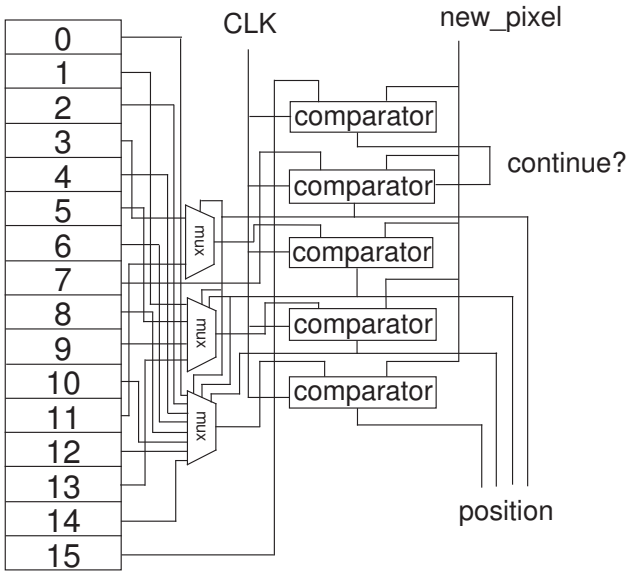


Figure 4.4: Tree search for sequenced memory queue

becomes 1. A decoder can translate the resulting sequence to binary representation of the position, which is a 4 bit number. See figure 4.5.

The parallel comparisons can be finished in one clock cycle and the updating of the queue requires one cycle. Totally, this parallel sorting unit requires two cycles to finish the job.

4.2.4 Hardware implementation of the parallel sorting unit

The hardware implementation of the queue is shown in figure 4.6. It needs a 16-comparators network, a 16 bits shift register, 16 multiplexers and 16 memory locations. The stage signal can be either 0 or 1 and it changes every clock cycle. At the first stage (stage = 0), the comparators simultaneously compare the incoming pixel to each element of the memory locations. The compared result called "greater_flag" will be sent to the shift register. The 16 bits of the "greater_flag" are also the write enable signals for the 16 memory locations of the queue. At the second stage, the shift register shifts the "greater_flag" to left by one bit. The "shifted_flag" then will be used as the select signals for the multiplexers. The multiplexers choose either the incoming pixels value or the upper position of the memory locations. In this implementation, no decoder is required to translate the 16 bits flag to a 4 bits position. The mechanism of how the flags work will be given below.

The position where 0 becomes 1 in the "greater_flag" represents the position where the incoming pixel should be placed. Each bit of the flag is the result of the comparison

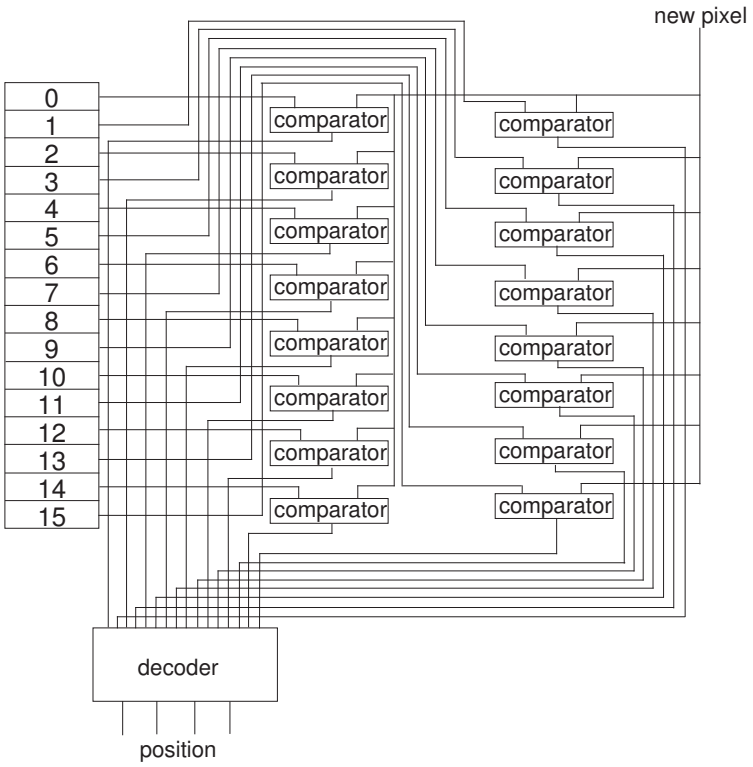


Figure 4.5: Comparator network for parallel sorting

between the pixel value and the current value at the memory location. For example, a flag of "111111111110000" means that the new pixel is smaller than the first to forth elements of the queue but greater than the fifth to sixteenth. The "greater_flag" will then be used as the write enable signal of memory locations. In this example, the 0 to 3 position of memory will be disabled so that no value can be written to those positions. Old values which are actually greater than the new pixel will be kept. At the second stage, the flag will be shifted left by one bit and used as select signals of multiplexes. In this example, the "shifted_flag" is "1111111111100000", meaning that mux_0 to mux_4 choose the new pixel while the mux_5 to mux_15 choose the values from memory_4 to memory_14 (the content of memory_15 is discarded). Then, the outputs of multiplexers are sent to the memory locations. Since position_0 to position_3 are write disabled, only position_4 are updated to pixels value and position_5 to position_15 are updated to the previous values in position_4 to position_14.

Since one queue requires two clock cycles, two such queues have to be maintained in real time to record 16 maximum numbers respectively. These two queues can work concurrently with the threshold calculation unit. The odd pixels are input to the comparator network associated with the left sequenced queue and the even pixels are input to the right one.

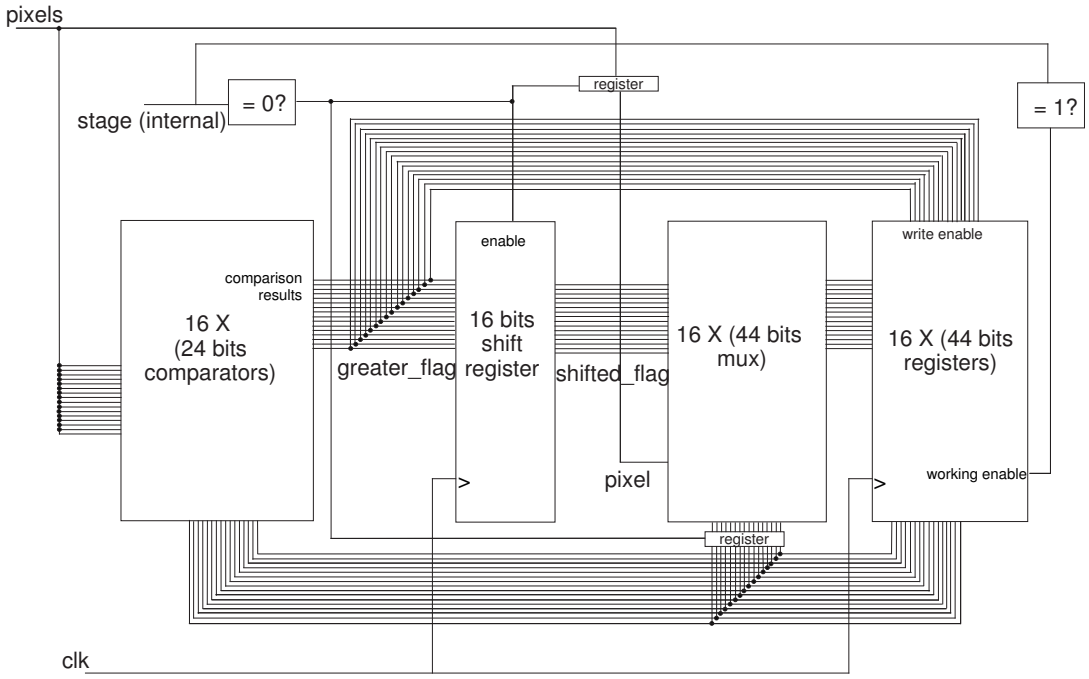


Figure 4.6: Hardware implementation of parallel sorting queue

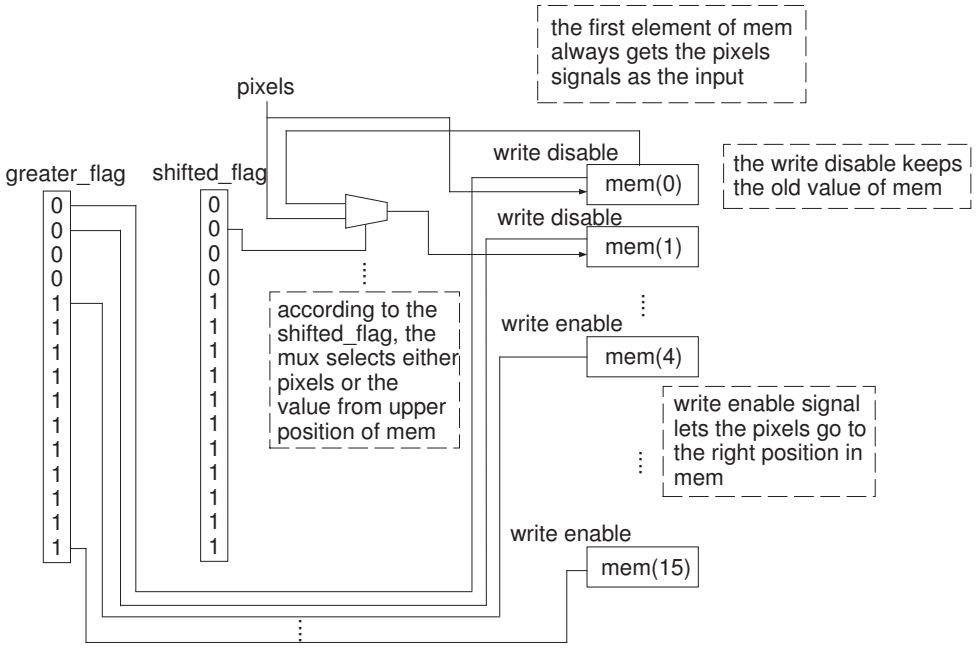


Figure 4.7: Mechanism of the queue updating

4.3 Adjacent Peak Removing

Often, the peaks that exceed the threshold are not just precisely one pixel position. The nearby (surrounding) pixels also have a high energy (value) that may be higher than the threshold. But these adjacent positions should not be considered as additional translation matches in the search image. The surrounding pixels can be considered as "almost" matches near to the true object position. An algorithm needs to be developed to remove those fake peaks and find the true position (usually the highest one). In order to locate the precise position of the object, the highest pixel among them should be found. (Sometimes the highest one is not the actual position of the object, but it is very close to the real position. Such error is acceptable.

4.3.1 Removing algorithm exploration

Different removing algorithm will be introduced and compared in the following sections. A suitable one will then be chosen and be implemented.

- **Block based algorithm**

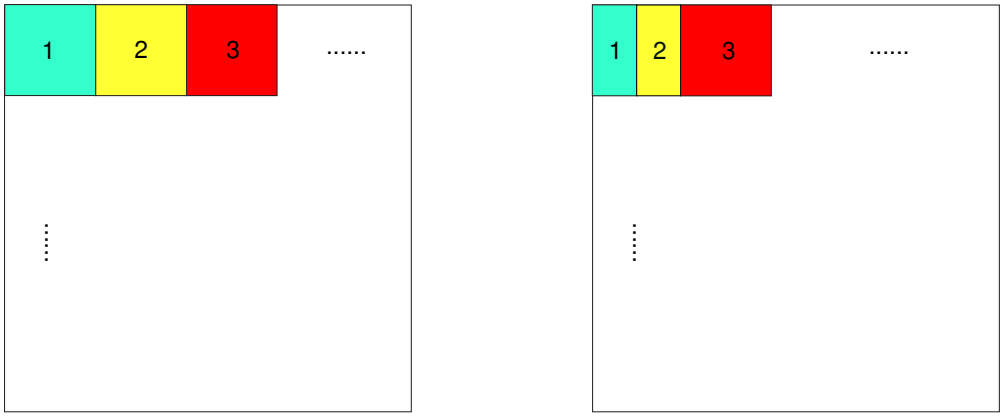
  The idea of block based algorithm is to search the peak matrix block by block while all the pixels that exceed the threshold in the same block will be considered adjacent. That means only one pixel (the highest one) can survive within one block. The block size is 32 by 32. The found pixel will then be output as the true location of the object and all the other pixels are discarded. A naïve way is to move the searching scope one block after another, which is called **non-overlapping scheme**. The disadvantage of this scheme is that two close pixels at two sides of the block borders will not be compared and removed if they are the highest pixels within their own blocks. A method to avoid this is called **overlapping scheme** which overlaps half area of the blocks. In that way, it covers the disadvantage of the non-overlapping scheme in a cost of double checking steps. But this scheme causes another less important problem. It will be discussed in the **Chain effect** section below.

- **Pixel based algorithm**

  Another solution is to detect adjacent pixels in a way of pixel by pixel. In this solution, not the whole matrix is scanned, but only the surrounding areas of the pixels that are exceed the threshold. See figure 4.9, a square area of 33 by 33 (distance 16) whose center is a detected pixel will be checked to find out if there are other detected pixels within this area. If it finds another pixel, they will be compared to each other and the pixel with smaller magnitude will be removed. This method can be much faster than the overlapping block scheme. The total number of areas check for pixel based algorithm is 32 (16 for each queue) while the total number of blocks that the overlapped block algorithm has is $916((32-1)\bullet(32-1))$.

- **Chain effect**

  Although the overlapping block algorithm or pixel based algorithm work quite well in most of the cases, they fail their task in some extreme cases. An example of such



(a) Non-overlapping block scheme (b) Overlapping block scheme

Figure 4.8: Block based adjacent peak removing algorithms

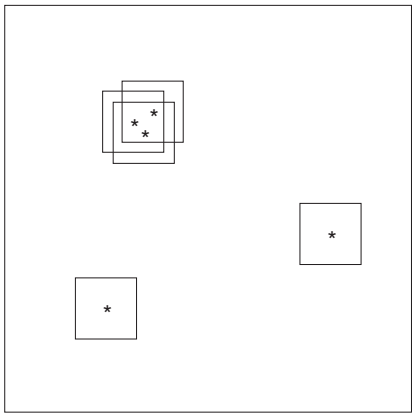


Figure 4.9: Pixel based adjacent peak removing algorithm

a situation in the overlapping block scheme is shown below (figure 4.10). Here the odd blocks and even blocks are intentionally placed with a slight displacement just to make the figure clearer. Suppose there are pixels in the right parts of blocks 1 to 6, and they are, at the same time, in the left parts of block 2 to 7 (7 is not shown here). Also suppose that the height of those pixels is increasing from block 1 to block 6. Then a problem occurs! The first pixel will be removed compared to the second one within block 2, the second one will be removed since the third one is greater than it within block 3.... Finally, all previous 5 pixels will be removed, while only the sixth pixel remains. But the distance between the sixth and first pixel is far more than 16 and they probably indicate different objects. The same case can also happen in the pixel based removing algorithm. Although this kind

of situation is hardly possible to happen in practice, at least it deserves some attention.

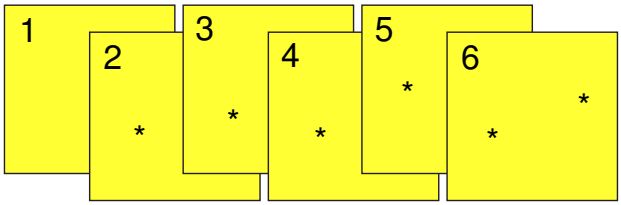


Figure 4.10: Chain effect

4.3.2 Hardware implementation of adjacent peak removing

- **The pixel format**

  The pixel in the adjacent peak removing unit contains the information of x position, y position and the value (the height) of the pixel. The range of x and y positions are from 1 to 512 for regular SPOMF and 1 to 1024 for rotation and scale SPOMF because of the oversampling, therefore 10 bits of each are needed for the representation of each coordinate. Because the exponent part of the pixel value is not needed for the calculation of the threshold and for the detection, only the 24 bit mantissa part is kept for the calculation, as shown in figure 4.11.

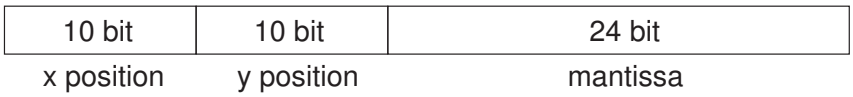


Figure 4.11: The pixel format

- **Algorithm implementation**

  Before the checking of adjacent peaks start, a memory queue of candidate peaks should be prepared. For each peak, it not really necessary to compare it to all the other candidate peaks, but only the peaks lower than (with larger memory index number) the current one should be checked. The basic idea of this implementation is that it compares each pair of candidate peaks in the memory. If two peaks are located in the same 33 x 33 block, they will be considered adjacent and their magnitude (mantissa part) will be compared. The smaller one will be marked as "removed" so that it will not be output in the later stage. To implement such a search, two registers are needed to remember the current index and search index. The current index points to the current candidate pixel in the candidate queue while the search index points to the search pixel that will be compared to the current

pixel. Each register has a counter associated with it to increase the index value. When the check unit finished its job, it gives the counter associated with the search index a "finished" signal to make this counter increment. When the search counter reaches the bottom of the candidate queue, it gives the counter associated with the current index a "reach_bottom" signal. Then the current counter increases itself by 1 and copies the index value to the search counter and the search counter makes this copy plus 1 as the initial position of searching. The checking unit checks the two elements that the current index register and the search register point to. If the two elements are not adjacent, it makes the found signal "0". If they are adjacent, it sets the "found" signal to 1, makes decision about which candidate should be removed from the queue and gives the address of that candidate to removing unit. See figure 4.12.

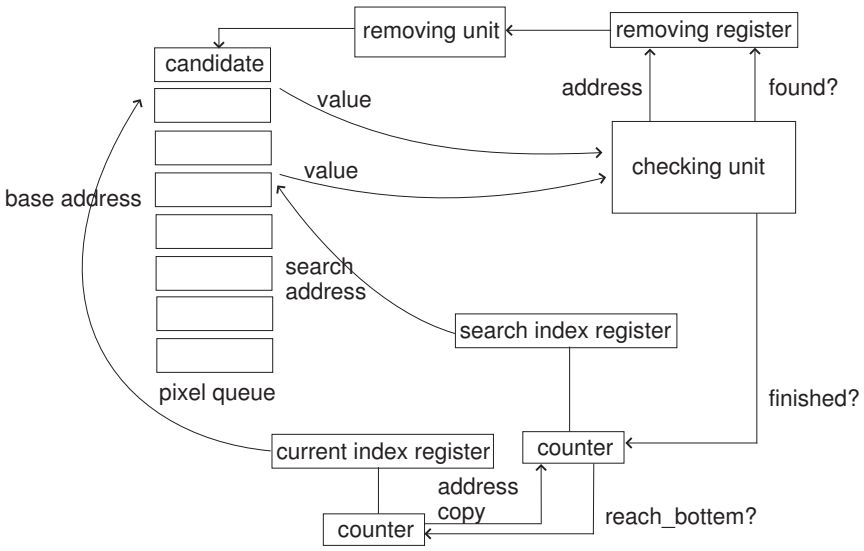


Figure 4.12: Schematic of adjacent peak removing unit

- **The checking unit**

  The checking unit shown above does two basic functions. First, it checks whether the two input pixels are close to each other, meaning both their x and y distances are within 16. Second, if the two pixels are confirmed to be close to each other, it picks up the smaller one and returns the queue address of it. See figure 4.13.

  If both of the x and y position of two pixels are within 16, they will be considered as adjacent pixels. Also, if the distance of two pixels is greater than 496, they will also be considered as adjacent pixels. Because the image itself is 512 by 512, if two pixels have their distance greater than 496, that means they are within $16(512 - 496)$ in the other direction. Situations where the x distance being less than 16 and the y distance being greater than 496, or the other way around, should also be considered as adjacent.

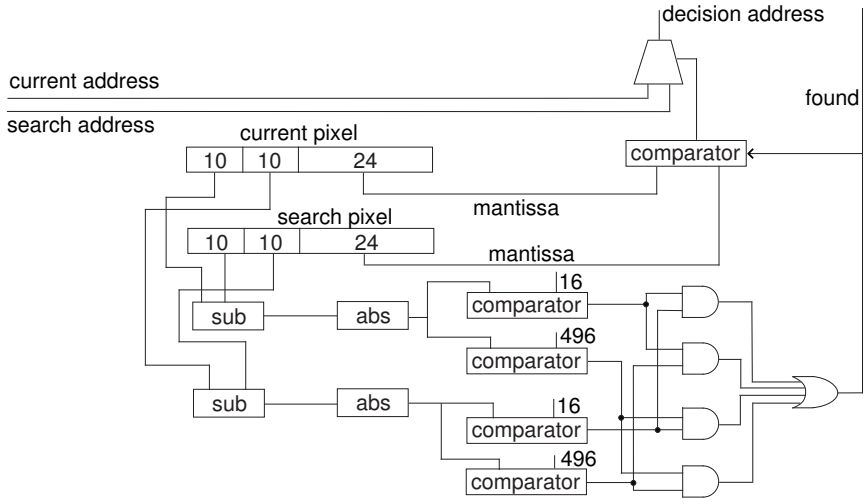


Figure 4.13: The check unit in the adjacent peak removing

4.4 Top Level Architecture of Peak Detection

The top level architecture of peak detection includes three basic parts: threshold determination, dual queues and adjacent peak removing unit. The correlated image (peak matrix) comes pixel by pixel in a 44 bits data bus with 10 bits for x position, 10 bits for y position and 24 bit for peak magnitude. The 24 bits of pixel value, in fact, is the mantissa part of a floating point representation of the pixel value. Simultaneously, the data goes to threshold calculation unit and dual queues. Each queue needs two clock cycles to update the sequenced maximum queue so two of them are required. After the threshold has been calculated, the minimum value of the two queues will be compared to the threshold to check if all the possible candidate peaks are within the queues. If not, it gives a "notice" signal to the user just to remind. The following operations continue. When the threshold calculation is done, it gives a signal to adjacent peak removing unit to indicate that the candidate peaks are ready. Then all the candidate peaks in the dual queues will be sent to the peak removing unit to remove adjacent peaks. When the threshold is being calculated, it requires the maximum value of all the pixels. This can be obtained by taking the larger one of the top elements in both queues. The schematic of the top level design is shown in figure 4.14.

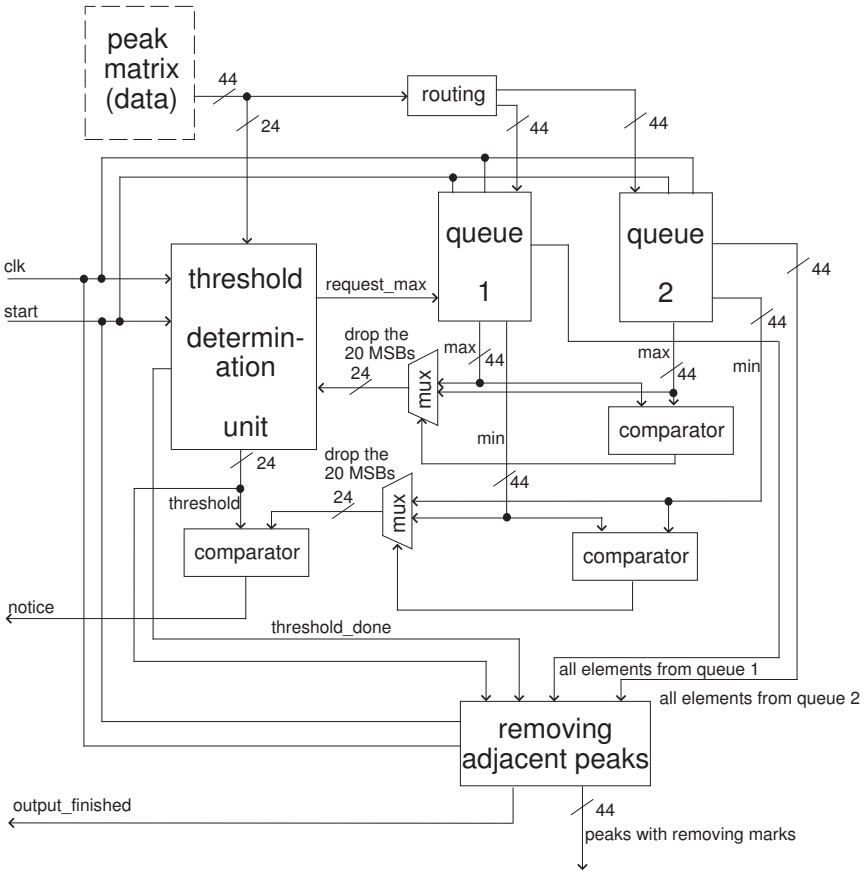


Figure 4.14: Schematic of detection top level

5 Hardware Integration

5.1 The PowerFFT Board

We implement the realtime image registration algorithm on to the PowerFFT board. The PowerFFT board contains a stand alone FFT chip that is capable of executing sustained FFT processing, vector multiplication convolutions and correlations on 1D complex data sets of up to 1K samples. The FFT chip has additional data ports for 4 SDRAM banks for long FFT processing or multidimensional FFT-based processing. Port 0 is the 64 bit primary input port, Port 5 is the 64 bit primary output port, and Port 1...4 can be connected to SDRAM banks to handle corner turning operations or act as double buffers.

Figure 5.1: The PowerFFT board

An addressing FPGA takes care of the SDRAM addressing (including refresh if necessary), such that the FFT processor is less dependent of external memory type. Another FPGA on board contains the switched fabric which can also contain customer functional units to perform functions like normalization and detection.

5.2 Control State Machine

The ND_controller (Normalizer and Detector) deals with the communication tasks for both the normalizer and the detector. The PowerFFT board will be connected to a host through Compact PCI port (CPCI). Software commands will be transmitted to the sequencer that is also located in the same FPGA as the ND_controller. The sequencer has three communication channels to the ND_controller. It gives the ND_controller a 32 bit "src_settings" signal which contains the software command and settings. At the same time, the "src_settings_valid" is set to indicate that the settings are valid. If the ND_controller is busy with processing data, the output signal - "src_busy" - will be raised to high (Receiving settings can not be interrupted by a src_busy. The block only gets settings when not busy.and raises src_busy directly after that). Also for the destination, the same three signals are required. The real data that will be processed, comes from the switched fabric. There are 6 channels between the switched fabric and the ND_controller. Four of them are the data input and output channels along with the data valid signals. The other two are called "src_stop" and "dest_full". With "src_stop", ND_controller can indicate it has to stop accepting data soon (it is almost full). With the "dest_full" signal, the switched fabric can indicate that the functional unit(s) cannot send any data soon.

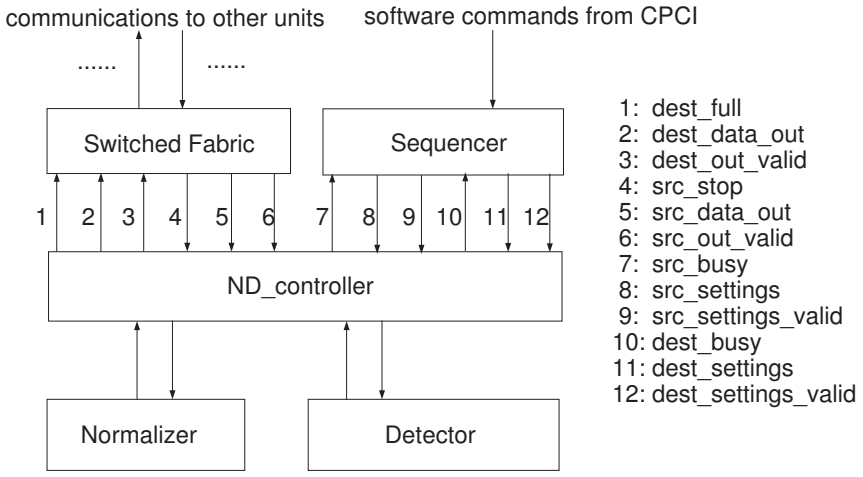


Figure 5.2: Communication signals to the switched fabric and sequencer

Before describing the state machine of the controller, it is necessary to introduce the definition of some output signals in the states. Table 5.1 gives a short description of all the output signals of the state machine.

The standard control state machine in the switched fabric always waits five clock cycles before and after the working states to insure the settings and data to be safely transmitted to the every functional unit. After the 5th input waiting state, the unit enters a formal working state, so the "src_busy" signal is raised and no other commands will be accepted from the sequencer. According to the lowest bit of the "src_settings" signal, either the normalizer or the detector will be activated.

Signals	Functions
dest_full	Stops the data transmission from switched fabric
src_busy	Stops any new operation command from sequencer
det_start	Starts the detection unit
nor_start	Starts the normalization unit
src_out_valid	Indicates the validity of the output data
t_suspend	Suspends the threshold calculation unit in the detector
q_suspend	Suspends the dual-queues unit in the detector
r_suspend	Suspends the adjacent peak removing unit in the detector
n_suspend	Suspends the normalizer
det_input_counter_start	Enables the data input counter of the detector
det_working_counter_start	Enables the working counter of the detector
det_output_counter_start	Enables the data output counter of the detector
nor_counter_start	Enables the working counter of the normalizer
det_end_of_data	Indicates the end of the output data of the detector

Table 5.1: Control signal definitions

For the normalizer, it is simple because the data input, output and processing happen in the same time. If the "src_stop" from the switched fabric becomes '1', meaning that the switched fabric is almost full, the normalizer will be suspended until it becomes '0' again. A counter called nor_counter counts the working cycles of the normalizer. When it reaches a pre-defined value (in our case, the image size is 512x512), the "nor_counter_reached" signal will become '1' so the normalizer enters the finishing state. In this state, the counter will be stopped and the data transmission should be stopped if there are any. Also, the output data from the normalizer should be indicated as not valid.

For the detector, it's more complex since it contains different working states. In the det_input states, image data is being input. At the same time, the threshold is being calculated and the queues are being built up. After that, the detector enters the det_working state, in which the adjacent peaks will be removed. The last working state of the detector is the det_outputting state where the correct peak locations will be sent out. All three states have their own counters to count the cycles of these operations and all of them can be suspended by disabling the counters. Figure 5.3 is the complete state machine diagram.

Figure 5.3: The control state machine for detector and normalizer

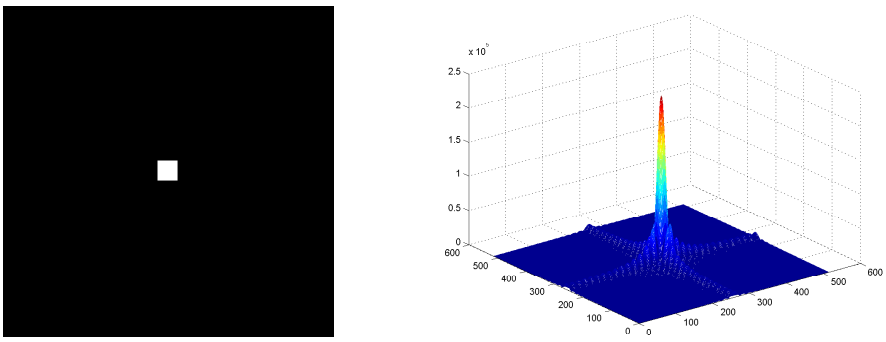
6 Test Results

6.1 The Cross Artifacts and the Edge-Fading Filter

When using SPOMF to detect the rotated and scaled object in the search image, an important issue that should be taken into consideration is the boundary of the image or the boundary of the template object. Because those boundaries are also frequencies in horizontal or vertical direction, in some cases, they will have a strong influence on the resulting frequency in the frequency domain. If the template image is a rectangular one or the object in the template image is selected by a rectangular, the edge of the rectangular can add a crossing along the horizontal and vertical directions in the frequency domain, which is called the **Cross Artifact** [7].

The reason of such an artifact is that the rectangle can be treated as the windowing operation for calculating the Fourier transform with limited amount of samples. For example, an object that is selected by a rectangle can be considered as an infinite large image with a windowing operation of square waves in the horizontal and vertical direction. The multiplication operation of the windowing in the spatial domain will cause a convolution in the frequency domain. So the actual frequency of the image feature will be convoluted by the frequency of the square waves.

Figure 6.1 shows a simple example of the Cross Artifact when a white rectangular is put in the center of the image. The right side of the figure is the frequency of this image. Notice that the DC component has been shifted to the center of the frequency matrix in order to give a clear example.

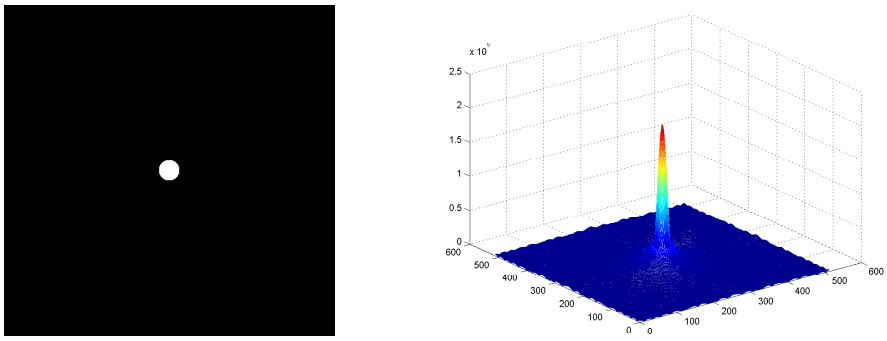


(a) (b)

Figure 6.1: Cross Artifact of rectangular windowing

To avoid the Cross Artifact, we can use other windowing solutions to lower the magnitude of frequency in the area other than DC, because the convolution to a DC

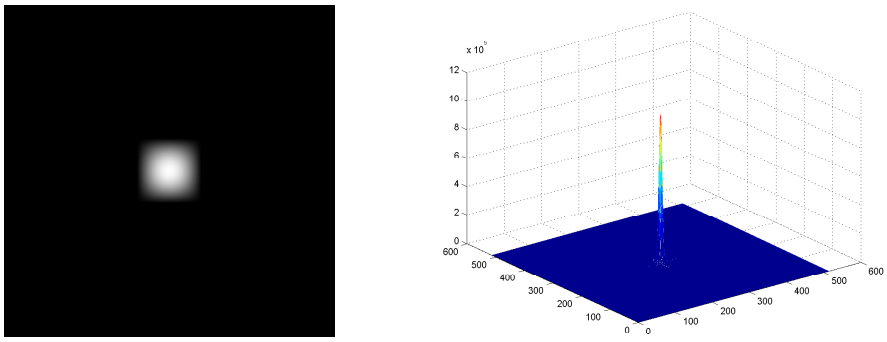
component will be harmless. Using a circular filter, that is actually a rotating square wave for windowing the object can provide a better performance than a rectangle filter but still not be good enough. See figure 6.2.



(a) (b)

Figure 6.2: Image windowing using circular filter and its frequency

The windowing technique we use for the test is called **Edge-Fading Circular Filter**. It is a circular filter with a fading edge that gradually becomes the same color as the background; in this case the background is black. The frequency of this filter is almost a DC component that has not much influence to the original frequency of the image feature. See figure 6.3.



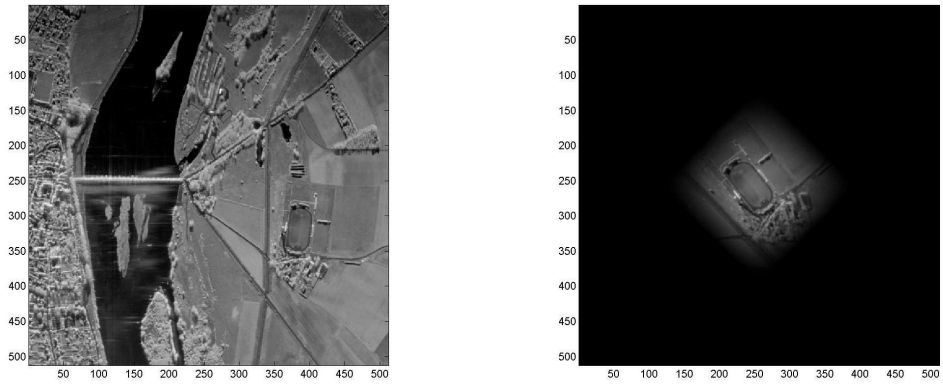
(a) (b)

Figure 6.3: Image windowing using Edge-Fading Circular Filter

6.2 A Satellite Image Test

In this section, a complete detection procedure will be given. The test image is a true satellite photo image. There is a sport center located in the search image and the template image is the rotated and scaled version of this sport center. The template image is not cut from the search image but from another photo. The template image

has been Edge-Fading Circular Filtered in order to remove the cross artifact. The task is to locate the precise position of the sport center in the search image by this filtered template. The search image and template image are shown in figure 6.4.

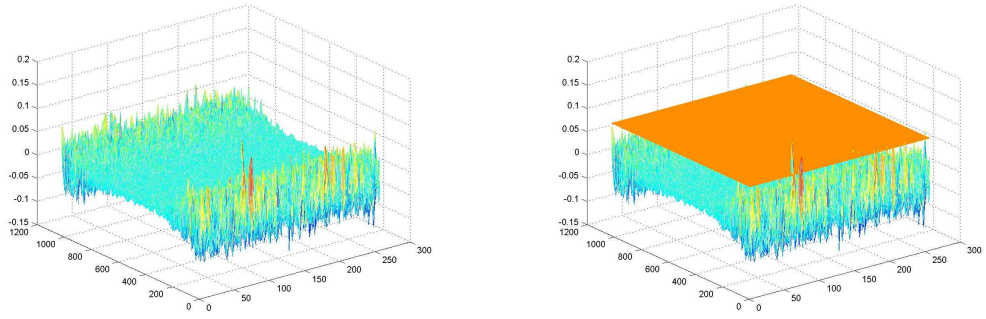


(a) (b)

Figure 6.4: Satellite search image and template image

Step 1. Detect the rotation and scaling factors

After applying the SPOMF operation to the search image and the template image, we get a correlate image shown in figure 6.5(a). By calculating the threshold (figure 6.5(b)) and removing the adjacent peaks, we obtain the rotation angle and scaling factor. The rotation angle is 45 degrees ($\pi/4$) and the scaling factor is 0.8333 (1:1.2).

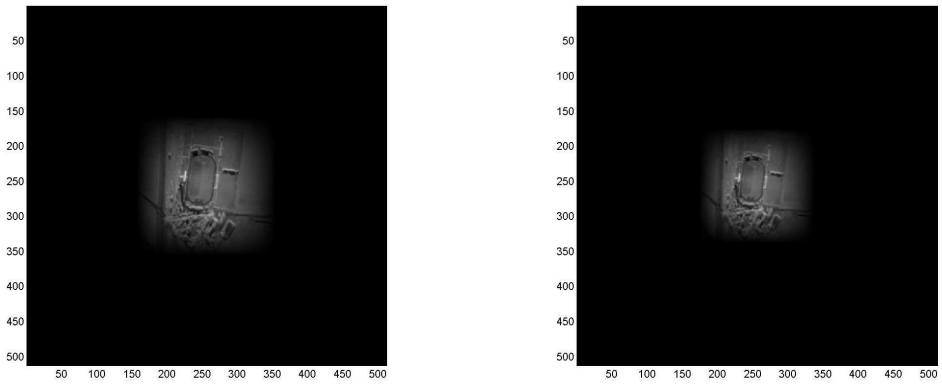


(a) (b)

Figure 6.5: Rotation and scale detection

Step 2. Rotation and scale compensation

The template image is then rotated and scaled to compensate for the detected the factors. Figure 6.6(a) is the rotation compensated template image while figure 6.6(b) is also scale compensated.



(a) (b)

Figure 6.6: Rotation and scale compensations

Step 3. Detect the actual location of the object

Now, the compensated template image contains the sport center that has the same rotation angle and scale as the sport center in the search image. By applying SPOMF again, we get a clean peak that indicates the location of the sport center. See figure 6.7.

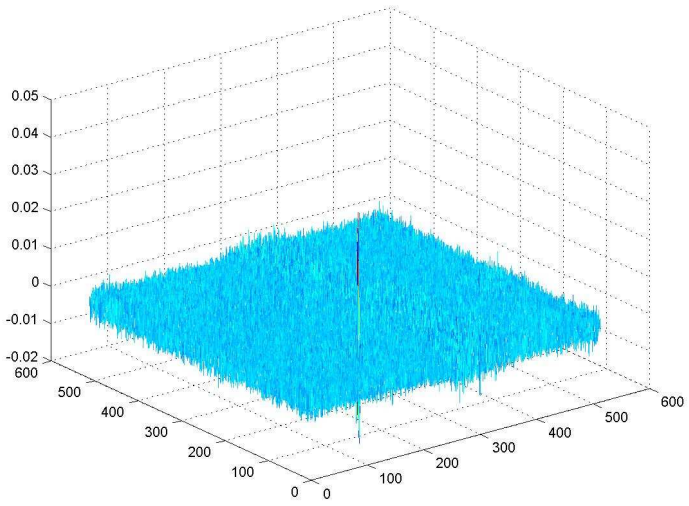


Figure 6.7: Detected location of the sport center

7 Conclusion

7.1 Conclusion

In this thesis work, a real time rotation and scale invariant object locating algorithm was introduced. Mathematical explanations of this algorithm were given for a better understanding of the theory. For two core components: phase normalization and peak detection, algorithms were developed and implemented on a reconfigurable platform that consists of PowerFFT processor and FPGAs.

For phase normalization, we explored several algorithms and found that an efficient and reliable solution was obtained by sign bit only (SBO) normalization. This solution had been mathematically proved to be reliable for different cases.

For peak detection, a self-adaptive threshold determination algorithm was developed. This algorithm can detect peaks in variant environments, independent of the average magnitude of the peak matrix. In order to save another scan of the peak matrix, a dual-queues architecture was designed and implemented. The queues can record the 32 largest candidate peaks, in which all the possible peaks are located for most practical cases. Also, an adjacent peak removing algorithm was developed to obtain the location of the detected object more precisely.

Practical test results are demonstrated to show the quality of the whole algorithm. The performance of the SPOMF in detecting non-rotation and non-scale objects is quite good. For rotated and scaled objects, the performance depends on multiple issues like the background of the objects, the scaling factor, the interpolation in the rotation and scale, the mapping the log-polar coordinate system, etc.

Generally speaking, this thesis work provides us some useful knowledge and experience in image registration techniques and it can be the foundation of future works in this area.

7.2 Future Work

- **Better information abstraction**

  As mentioned before, the information of translation, rotation and scale are not nicely distributed in the phase and magnitude of complex numbers in the frequency. The method we use is simply using the phase to detect the translation and using the magnitude to detect rotation and scale. In fact, rotation and scale information is present both in the phase and the magnitude. Therefore, in the future maybe better information abstraction techniques can be developed to greatly enhance the performance of this algorithm.

- **Peak filtering**

Further research can be done to increase the quality of the output peak in the correlated image. By gathering different kinds of image samples and studying the peak shape of different correlated images, a combination of multiple peak filters can be designed. This filter can increase the difference between the peak and the noise level, which makes the detection much easier.

- **3D detection?**

  The whole algorithm can be extended to 3 dimensional space with the same techniques (using a 3D camera). The only problem is that it requires a huge amount of calculations. We can hope that future powerful chips and systems will be available to perform the tasks.

Bibliography

[1] Peter Beukelman and Laurens Bierens, *Real-Time Rotation and Scale Invariant Template Matching in Streaming Images*, GSPx (2004)

[2] B. Srinivasa Reddy and B. N. Chatterji, *An FFT-Based Technique for Translation, Rotaion, and Scale-Invariant Image Registration*, IEEE Transaction on Image Processing, Vol 5, No. 8, page 1266-1271 August 1996.

[3] A. Averbuch, R. R. Coifman, D. L. Donoho, M. Israeli, and J. Waldn *Polar FFT, rectopolar FFT, and applications*, Stanford Univ., Stanford, CA, Tech. Rep., 2000.

[4] Michael Unser, *Convolution-Based Interpolation for Fast, High-Quality Rotation of Images*,IEEE Transactions on image processing, Vol .4, No. 10, page 1371-1381, October 1995.

[5] Javier Valls, *Evaluation of CORDIC Algorithms for FPGA Design*, Journal of VLSI signal processing 32, page 207-222, 2002.

[6] Israel Koren, *Computer Arithmetic Algorithms, 2nd Edition*, page 233-234, 2002.

[7] Morgen McGuire, *An Image Registration Technique for Recovering Rotation, Scale and Translation Parameters*, Massachusetts Institute of Technology, Cambridge, MA, 1998.

Curriculum Vitae

Meng Ma was born in Hangzhou, China, on July 5th, 1980. He graduated from Associated High School of Zhejiang University in 1999. In the same year, he got the first place of the university entry exam within the high school and was admitted to Computer Science and Technology Faculty of Zhejiang University in Hangzhou, China.

After finishing the Bachelor study and receiving the BSc degree, he was admitted to Delft University of Technology. He joined the Computer Engineering Group led by Prof. Stamatis Vassiliadis. He performed his graduation project in Eonic B.V. in Delft under the supervision of Ir. Peter Beukelman at Eonic and Dr. Arjan van Genderen in the university. His research interests are image progressing, digital signal processing, computer architecture, and FPGA based design.