Spring 2012

An Analysis of Rigid Image Alignment Computer Vision Algorithms

Rajeshree R. Joshi

AN ANALYSIS OF RIGID IMAGE ALIGNMENT COMPUTER VISION ALGORITHMS

by

RAJESHREE JOSHI

(Under the Direction of Robert Cook)

ABSTRACT

Computer vision is a field of computer science that includes methods for acquiring, processing, and analyzing images. Image registration is one of the methods used in the computer vision field to transform different sets of data into one coordinate system to align images. Registration is important in order to be able to compare or integrate the data obtained from multiple measurements. Rigid image alignment is a type of image registration technique used to align two two-dimensional images into a common coordinate system based on two transformation parameters, translation and rotation. Before any comparative studies can be performed on two images acquired at different times, it is crucial to align the two images for correct processing later on.

In our research study, we are analyzing the accuracy of registering images using two rigid image alignment algorithms, namely the Principal Axes algorithm and the Fast Fourier Transform (FFT) based phase correlation algorithm. The software for registering images using these two methods is written in MATLAB R2011a. We also compared our results with alignments achieved for the same images using an existing Statistical Parametric Mapping (SPM8) package for registration.

Image registration algorithms have been used in many applications and accordingly, algorithms are adopted to suit a particular application. Images used for registration can be derived from different capturing devices like camera, scanner, satellite sensors, etc. Our registration software is based on work with images acquired from a Magnetic Resonance Imaging (MRI) scanner and especially for images taken of a quality assurance (QA) phantom. A QA phantom is used to test the quality of images acquired by measuring different QA parameters on images acquired over a period of time. Images acquired from the MRI scanner at different times are geometrically transformed by rotation and translation. In practice, the maximum angle by which the phantom will get rotated at different times due to varying positioning in the scanner will not be greater than 50 degrees and the maximum displacement will always be less than 50 pixels based on our experience while scanning. By comparing future phantom images with the first image in the series, we can perform a series of Quality Assurance steps to measure any degradation in the MRI device. The QA results can then be used to apply inverse transformations to new customer images to improve their quality. The first step in the QA process is image registration, which is the topic of this thesis.

To test the implementations, we rotated and translated known images then we applied the two algorithms and compared the results to the known translation and rotation values. Our analysis shows that the Principal Axes method could successfully register 17 of the 22 non-aligned test images, the FFT method registered 21 test images successfully whereas SPM8 with default settings showed correct alignments for only 9 images in our case study as per our requirement. The Principal Axes algorithm performed better image

alignment when the two images were displaced by a larger distance, and the FFT based algorithm performed better for larger rotation angle differences among images. Hence, we conclude that our algorithms have the potential for inclusion in the new QA process.

INDEX WORDS: Rigid image alignment, Computer vision algorithms, Image registration, Principal axes, Fast fourier transform, FFT, Mean square error, MSE.

AN ANALYSIS OF RIGID IMAGE ALIGNMENT COMPUTER VISION ALGORITHMS

by

RAJESHREE JOSHI

(Under the Direction of Robert Cook)

B.E., Goa Engineering College, India, 1995

M.S., Georgia Southern University, 2012

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment

of the Requirements for the Degree

MASTER OF COMPUTER SCIENCE

STATESBORO, GEORGIA

2012

AN ANALYSIS OF RIGID IMAGE ALIGNMENT COMPUTER VISION ALGORITHMS

by

RAJESHREE JOSHI

Major Professor: Robert Cook
Committee: Lixin Li
Vladan Jovanovic
James Harris
Nathan Yanasak

Electronic Version Approved: May 2012

ACKNOWLEDGMENTS

I sincerely express my appreciation to my advisory committee: Dr. Robert Cook, Dr. Nathan Yanasak, Dr. Lixin Li, Dr. Vladan Jovanovic, and Dr. James Harris.

Special thanks to Dr. Robert Cook for his time, patience, and understanding. Thanks for your full cooperation during the research work.

Dr. Yanasak, thanks for giving me the opportunity to be part of this research work in the Core Imaging Facility for Small Animals (CIFSA) at Georgia Health Sciences University. Also, thanks for letting me use the facility resources for my thesis work.

Thanks to College of Graduate Studies at Georgia Southern University for making this study possible by funding it through Graduate Research Thesis Grant.

My gratitude goes to Dr. Lixin Li for being very helpful at every step of my school year. Special thanks to Dr. Jovanovic and Dr. Harris for their valuable suggestions.

I also take this opportunity to thank Dr. Wen-Ran Zhang and Dr. Wenjia Li for their cooperation and help during the school year.

The most special thanks go to my husband Dr. Sharad Purohit, my kids Neha and Raghav, my brother Pramod, my sister Shradha, my parents and my in-laws for their unconditional support and love through this long process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

"Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution."

– Albert Einstein

1.1 Motivation and Scope

Vision is the task of "seeing" with understanding. When we see things, our eyes (sensing device) capture the image, and then pass the information to the brain (an interpreting device). The brain interprets the information and gives meanings to what we see. In computer vision, a camera or scanner or any such device serves as a sensing device, and a computer acts as an interpreting device. Computer vision embeds the core technology of automated image analysis which is used in many fields. Medical computer vision or medical image processing is one of the most prominent application fields of computer vision [12].

The organization of a computer vision system is highly application dependent. There are, however, typical functions that are found in many computer vision systems [12]. They are image acquisition, pre-processing, feature-extraction, segmentation, high-level processing and decision making [12].

When multiple images are captured from different viewpoints or at different times, the images get distorted with respect to each other. Image registration or image alignment is the process of determining the optimal transformation matrix that results in the images being in spatial alignment [1].

Various image registration algorithms are designed to suit different applications. A registration algorithm designed for one application may not work or may work inefficiently because of the different image formats achieved from different capturing devices and based on application requirements. Algorithms will also vary based on the fact that the images to be aligned are from the same object (monomodal) or different. The images may need to be preprocessed or converted into formats to be supported by the existing algorithms. Even then, the algorithms may have their own limitations as to how much displacements or rotation angle differences it can handle for registration. Hence, existing algorithms need to be adapted based on the type of application it will serve.

Our application demands registering images acquired from a quality-assurance (QA) phantom using an MRI scanner. The phantom is designed to test the quality of images acquired from the scanner on a weekly basis over a time span of years. Image quality can be tested based on different QA parameters measured for the images. Before this processing can be done, all the acquired images obtained from the scanner using the same phantom at different times need to be aligned with a reference image to facilitate correct measurements for QA parameters. The images to be aligned are from the same phantom but acquired at different times. Figure 1 shows the experimental setup for acquiring images using the MRI scanner and the QA phantom.

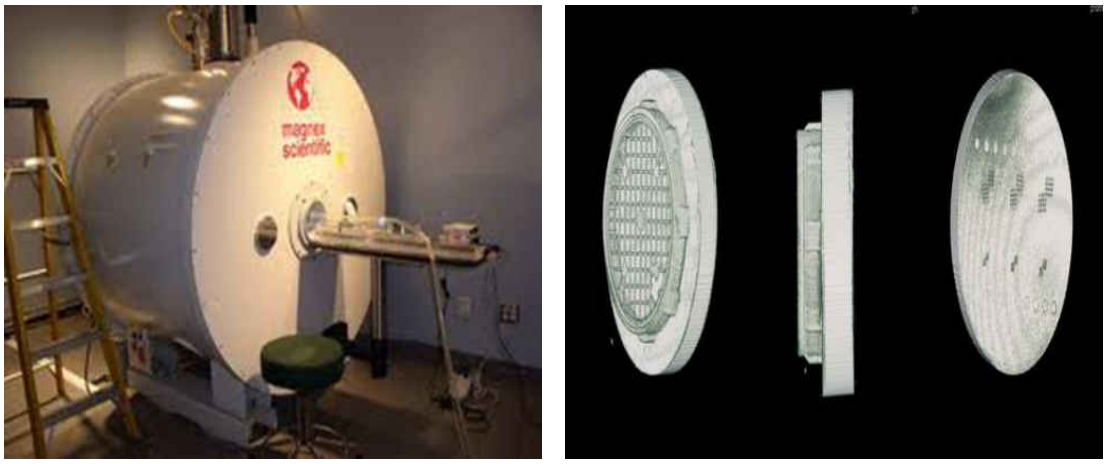


Figure 1: Experimental Setup. Left panel shows the Bruker BioSpin 7T MRI scanner used for acquiring images by placing the QA phantom as shown in the right panel on the scanner bed. The central disk in the phantom, used for resolution, contrast, and distortion measurements, is shown in the right panel. Bruker ParaVision 5.1 software is used to instruct the scanner to acquire the images.

The phantom images acquired using the scanner at different times are geometrically distorted by rotation and translation. This is due to different positioning of the phantom in the scanner each time. The circular patterns shown in the images need to be at similar coordinates for all the images for further meaningful processing. The alignment of images then is crucial in deriving correct QA parameter values to test image quality and therefore the proper functioning of the scanner. Misalignments will result in wrong measurements for QA parameters providing us with incorrect results on the scan quality of images and the functioning of the scanner.

Since the scanner functioning may deteriorate with time in the long run, it is possible that the images acquired can get distorted with more noise introduced in the future images. To tackle this problem, we have simulated two of our target images by introducing Gaussian noise enough to pixelate the images using the ImageJ (NIH, MD) [14] software.

Before designing our registration software, we have used the existing SPM8 software package [13] for our application to align these images. For this, we had to first convert our raw phantom images to the NIFTI file format required by SPM8 for image registration. We used ImageJ software for file conversion. We then registered images using the Realign Tool in SPM8. Default parameter settings were used for realignment as shown in the screenshot window in APPENDIX D. Of the 22 phantom images to be aligned with the reference image, SPM8 could correctly register only 9 images in our study to our satisfaction and requirement.

Considering our application requirements and the limitation of existing software to suit our needs, we have designed rigid image registration software in MATLAB R2011a [15]. No image preprocessing is required for using our software. It allows the user to choose between two algorithms for registering images. The algorithms chosen are Principal Axes method [1, 3] and FFT based phase correlation method [1, 4, 7, 8, 9, 10] for their simplicity, applicability and efficiency amongst other rigid image alignment algorithms.

There are 23 images used in our study. The target images need to be aligned with the first image in the sequence considered as a reference image. The software displays each step of transformation to the user during processing. In addition, the software also displays a commonly used similarity measure, Mean Square Error (MSE) [12]. The total time taken for image alignment is also shown to the user. The transformed aligned image is then saved as a raw file with the same name as the target file prefixed with letters ‘PA’ or ‘FR’ based on the algorithm chosen for registration.

1.2 Related Work

"Image registration is a vital problem in medical imaging" [5, 6]. In [1] and [2], "rigid body registration methods are effectively used for registering human brain images from MRI. It has many potential applications in clinical diagnosis (Diagnosis of cardiac, retinal, pelvic, renal, abdomen, liver, tissue etc disorders) [6]. Image registration finds its applications in various fields like remote sensing (multispectral classification), environmental monitoring, change detection, image mosaicing, weather forecasting, creating super-resolution images, integrating information into geographic information systems (GIS), in medicine (combining data from different modalities e.g. computer tomography (CT) and magnetic resonance imaging (MRI), to obtain more complete information about the patient, monitoring tumor growth, treatment verification, comparison of the patient's data with anatomical atlases, in cartography (map updating) and in computer vision (target localization, automatic quality control) [6]."

1.3 Research Plan

The thesis work involved implementing the following steps:

a. Acquiring multiple QA phantom images on a weekly basis by positioning the phantom in the MRI scanner. There are a total of 23 images used in our study; the first one is used as the reference image and the remaining 22 are target images to be aligned with the reference image. The target images were created in ImageJ by applying different transformation parameters on the reference image for testing our algorithms.

b. Converting the raw images to NIFTI file format using ImageJ software to be used for alignment testing using the existing SPM8 package. Use the tool 'Realign' with default settings to align the images. Check the alignment with 'Check Registration'.

c. Designing rigid image registration software in MATLAB R2011a. The user is given the choice of using either Principal Axes Algorithm or FFT based phase correlation method for registration.

d. Finding the alignment similarity between the two images by showing the user the difference between the two images. Commonly used simple similarity measure Mean Square Error (MSE) is also calculated.

e. Saving the transformed aligned image as a raw image for further processing by the user.

f. Analyzing the results and comparing the algorithm efficiency using the similarity measures.

CHAPTER 2

IMAGE REGISTRATION TECHNIQUES

"We are what we repeatedly do. Excellence then, is not an act, but a habit."

- Aristotle

2.1 Introduction

The images need to be geometrically aligned for better observation [6]. This procedure of mapping points from one image to corresponding points in another image is called Image Registration. It is a spatial transform [6]. According to [5], [6] and [12], the image registration algorithms can be classified into following different categories:

a) Intensity-based versus feature-based: Intensity-based methods compare intensity patterns in images via correlation metrics, while feature-based methods find correspondence between image features such as points, lines, and contours. Knowing the correspondence between a number of points in images, a transformation is then determined to map the target image to the reference image, thereby establishing a point-by-point correspondence between the reference and target images [11].

b) Transformation Models: Linear transformations include translation, rotation, scaling, and other affine transforms. Rigid registration is a type of affine transformation. Non-rigid transformations are capable of locally warping the target image to align with the reference image [11].

c) Spatial vs. frequency domain methods: Spatial methods operate in the image domain, matching intensity patterns or features in images [12]. Frequency-domain methods find the transformation parameters for registration of the images in the transform domain. Applying the Phase correlation method to a pair of images produces a third image which contains a single peak. The location of this peak corresponds to the relative translation between the images. The phase correlation method uses the Fast Fourier Transform to compute the cross-correlation between the two images. The rotation and scaling differences between two images can be determined by first converting the images to log-polar coordinates.

d) Single- vs. multi-modality methods: Single-modality methods register images in the same modality acquired by the same scanner/sensor type, while multi-modality registration methods register images acquired by different scanner/sensor types. Multi-modality registration methods are often used in medical imaging as images of a subject are frequently obtained from different scanners [12].

e) Automatic vs. interactive methods: Registration methods may be classified based on the level of automation they provide. Manual methods provide tools to align the images manually. Interactive methods reduce user bias by performing certain key operations automatically while still relying on the user to guide the registration. Semi-automatic methods perform more of the registration steps automatically but depend on the user to verify the correctness of a registration. Automatic methods do not allow any user interaction and perform all registration steps automatically [12].

2.2 Rigid Image Registration

Rigid body registration is one of the simplest forms of image registration. Whenever several images of the same subject have been acquired, it is very useful to have them all aligned properly. Advantages of aligning images include allowing images to be averaged in order to increase signal to noise, or to subtract one image from another to emphasize differences between the images. Rigid registration is normally used for registering images of the same subject that have been collected at different times. Even if images were acquired during the same scanning session, it is possible that the subject may have moved slightly between image acquisitions due to different positioning each time [2].

In simple terms, image registration involves estimating a mapping between a pair of images. One image is assumed to remain stationary (the reference image), whereas the other (the target image) is spatially transformed to match it. In order to transform the target to match the reference, it is necessary to determine a mapping from each pixel position in the reference to a corresponding position in the target. The target is then re-sampled at the new positions [2]. The mapping is like a function of a set of estimated transformation parameters. A rigid-body transformation in two dimensions is defined by two parameters, translation and rotation [2]. The translation is defined each along the x-axis and the y-axis.

According to [1], general steps involved in Rigid Image registration methods are

a. The Feature Space: Before registering two images, we need to decide what is it that needs to be registered. The type of algorithm chosen will depend

on the features chosen. The features chosen can be pixel intensities, edges, contours, surface volume, etc.

b. The Search Space: To align two images, we need to define a transformation. If $f(x)$ and $g(x)$ are two real-valued functions for the two images to be aligned then we need to find the transformation $T(x)$ such that $f(x) = g(T(x))$ for all x. The simple rigid-body transformation is used for translation and rotation, affine transformation can be chosen to account for scaling too, and for non-linear, non-uniform deformations, non-rigid transformation can be used. Rigid and affine transformations are global, ie. applied to the whole image whereas non-rigid transformation can be localized to sub-regions of images.

c. The Search Strategy: Once a transformation say, $T_0(x)$ is chosen, we have to figure out a strategy based on results of applying $T_0(x)$ to choose the next transformation. Some ways include linear programming methods, relaxation techniques, or energy or cost minimization methods.

d. The Similarity Metric: We need to quantify the differences between the geometrically transformed image and the reference image to measure how well $f(x)$ compares with $g(T(x))$. Using mean square error or correlation is the key.

Rigid-body transformations consist of only rotation and translation [2]. They are a subset of the more general affine transformations. If a point x is to be translated by q units, then the transformation is simply written as: $y = x + q$ [2]. In two dimensions, a

rotation is described by a single angle [2]. Consider a point at co-ordinate (x1; y1) on a two dimensional plane. A rotation of this point to new co-ordinates (x2; y2), by radians around the origin, can be generated by the transformation [2]:

$$x2 = \cos(\theta)x1 + \sin(\theta)y1 \quad \text{and}$$

$$y2 = -\sin(\theta)x1 + \cos(\theta)y1 \qquad \text{eq.(1)}$$

Image registration thus involves estimating a set of parameters describing a spatial transformation that best matches the two images. The similarity of the registration is based on a cost function, which is maximized or minimized using some optimization algorithm [2]. This is called cost optimization.

CHAPTER 3

SOFTWARE DESIGN

"As a rule, software systems do not work well until they have been used, and have failed repeatedly, in real applications."

- Dave Parnas

3.1 User Interface

MATLAB is a high-level technical computing language [15]. It provides an interactive environment for algorithm development, data visualization, data analysis, and numeric computation [15]. MATLAB R2011a can solve technical computing problems faster than the traditional programming languages, such as C, C++, and Fortran [15]. MATLAB has been used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology [15].

We have designed our software in MATLAB R2011a language. The user interface shows the QA phantom image on its main page. The left panel lets the user choose the algorithm for registration. The user is then prompted to select the raw type reference file and the target file. Our program supports alignment of two same size images as required by our application. The user is asked to enter the width of the image. The height is considered equal to the width of the image.

The software then invokes the necessary registration algorithm based on the user's choice. The right panel displays every step of transformation in the form of images. Finally, the similarity measure Mean Square Error (MSE) along with total execution time of the algorithm are displayed. Translation and rotation errors are also displayed after executing the Principal Axes algorithm.

The user can then continue registering images using the same algorithm or choose other algorithms for alignment. The user has the advantage of testing the image alignment with both algorithms and choosing the one which gives the best results. Figure 2 shows the initial user interface windows.

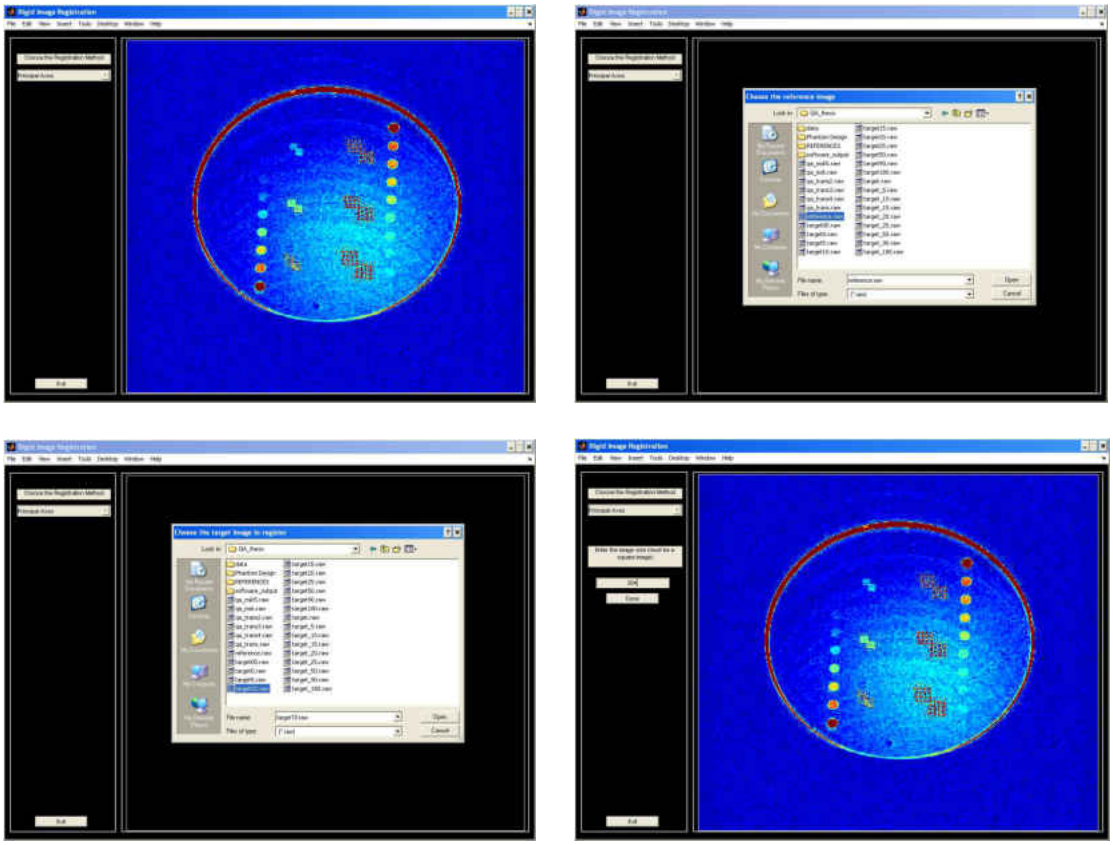


Figure 2: User interface windows. Top-left window shows the main window. Top-right window prompts user for selecting the reference file. Bottom-left window prompts user for choosing a target file. Bottom-right window asks user to input the image size.

3.2 Input

For registering two images, the user is required to choose two files. The first file is called the reference file to which the misaligned or target file will be aligned to. The file format supported is the raw binary format. The images derived from the QA phantom are unsigned 16-bit images in big-endian format. The file size is 384x384 ie. the images are square images. Each image is made up of three slices, collectively representing the volume of the phantom. The first slice shows only the boundary of the phantom. Second slice shows the varying intensity circular patterns from the phantom disk. And the third slice shows a circular disk with no patterns. Figure 3 shows the complete phantom image.

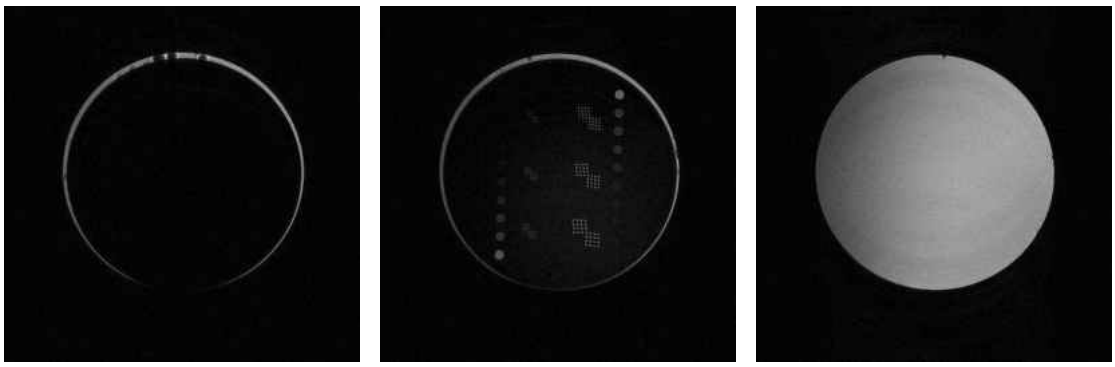


Figure 3: Input Image. Leftmost image is shows the first slice of QA phantom. Middle image shows the circular hole patterns in the second slice of the phantom disk. The rightmost image shows the third slice in the image.

The target images are transformed using ImageJ to provide varying displacements and rotations in images for testing our algorithms. The *Rotate* and *Translate* tools help perform these transformations on the target files. Gaussian noise is introduced in two of the target images to pixelate them to test for noise sensitivity of our algorithms. The *Noise-Add Specific Noise* tool in ImageJ was used to introduce Gaussian noise of 500 and 1000 standard deviation in target files target10n and target25n respectively.

The data files and their initial transformations are chosen based on practical applicability. The dataset is categorized into three based on the transformation values. The first category of dataset has files with lower values of transformations ie. with images rotated by smaller angles ranging from 0 to 25 degrees with 5 degree incremental steps compared to the reference image. The second category of dataset has files transformed between 50 degrees to 90 degress and the third dataset has transformation values greater than 90 degrees. The files with transformations other than those mentioned in the given dataset are also tested with our registration algorithms. The translation values are chosen to lie between 0 to 50 pixels as per our practical requirements. Table 1 shows the file names and their initial transformation parameters.

Table 1: *Data file names and the initial rotation and translation values*

File Name	Rotation angle (in degrees)	Translation along X-axis (in pixels)	Translation along Y-axis (in pixels)
Reference.raw	-	-	-
Target.raw	0	0	0
Target0.raw	0	15	10
Target00.raw	0	50	-25
Target5.raw	5	5	5
Target10.raw	10	0	5
Target15.raw	15	15	10
Target20.raw	20	10	20
Target25.raw	25	20	20
Target50.raw	50	25	-25
Target90.raw	90	5	10
Target150.raw	150	0	0
Target180.raw	180	0	0
Target_5.raw	-5	-5	-5
Target_10.raw	-10	0	5
Target_15.raw	-15	15	10
Target_20.raw	-20	-5	5
Target_25.raw	25	10	10
Target_50.raw	-50	10	10
Target_90.raw	-90	5	10
Target_180.raw	-180	0	0
Target10n.raw (noise=500 std)	10	0	5
Target25n.raw (noise=1000 std)	25	-20	20

3.3 Processing

The software processes the images for registration based on the algorithm chosen. Every step of transformation is then displayed to the user in the right panel of the interface window. Figure 4 shows the processing window after executing the Principal Axes algorithm. The images displayed are explained as below:

1. The first image shows the reference image.
2. Second image is the target image.
3. Third image shows the initial alignment difference between the two images. The reference image is shown in red and the target image in blue.
4. The fourth image shows the features selected for the reference image for alignment.
5. The fifth image shows the features selected for the target image for alignment.
6. The sixth image is the rotated features of target image for alignment.
7. The seventh image is the rotated target image.
8. The eighth image is translated features of target image for alignment.
9. The ninth window is the transformed or aligned target image.
10. The tenth window shows the difference between the reference image and the aligned target image. If the alignment is correct, the image difference shows up as white and the regions of image that couldn't be aligned show up in the respective red or blue color for misalignment.

11. Finally, the program displays the similarity measure MSE values. Translation error and rotation error measure are also displayed. Algorithm execution speed is shown in seconds.

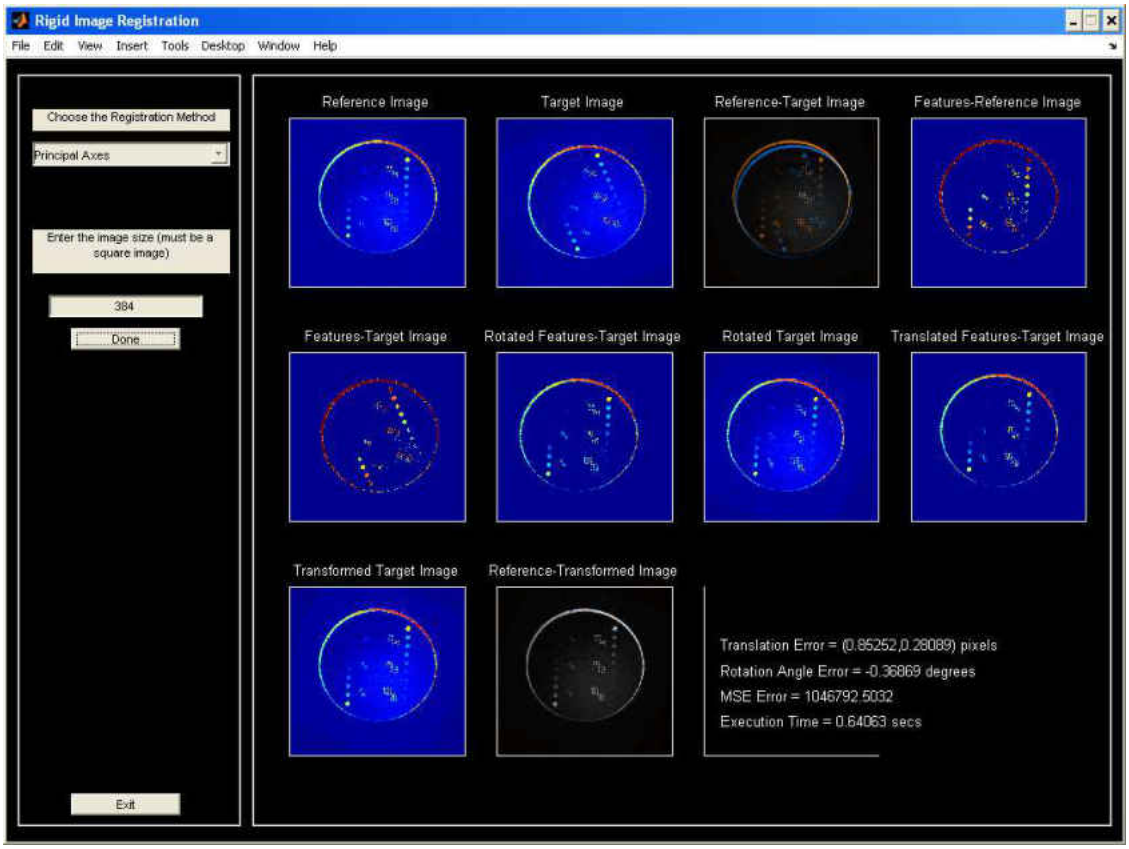


Figure 4: Software Processing Window - Principal Axes Algorithm. The figure shows each processing step after executing the Principal Axes Algorithm.

Figure 5 shows the processing window after executing the FFT algorithm. The following is the explanation for each image displayed.

1. The first subplot shows the reference image.
2. Second subplot is the target image.
3. Third subplot shows the initial alignment difference between the two images. The reference image is shown in red and the target image in blue.
4. The fourth and fifth subplots show the frequency domains of reference and target images respectively.
5. The sixth and seventh subplots show the frequency domains of reference and target images respectively represented in log-polar coordinate system.
6. Eighth image is the rotated version of the target image.
7. Ninth image is the translated target image.
8. Tenth window shows the difference between the reference image and the aligned target image. If the alignment is correct, the image difference shows up as white and the regions of image that couldn't be aligned show up in the respective red or blue color for misalignment.
9. Finally, the program displays the similarity measure MSE values. Algorithm execution speed is shown in seconds.

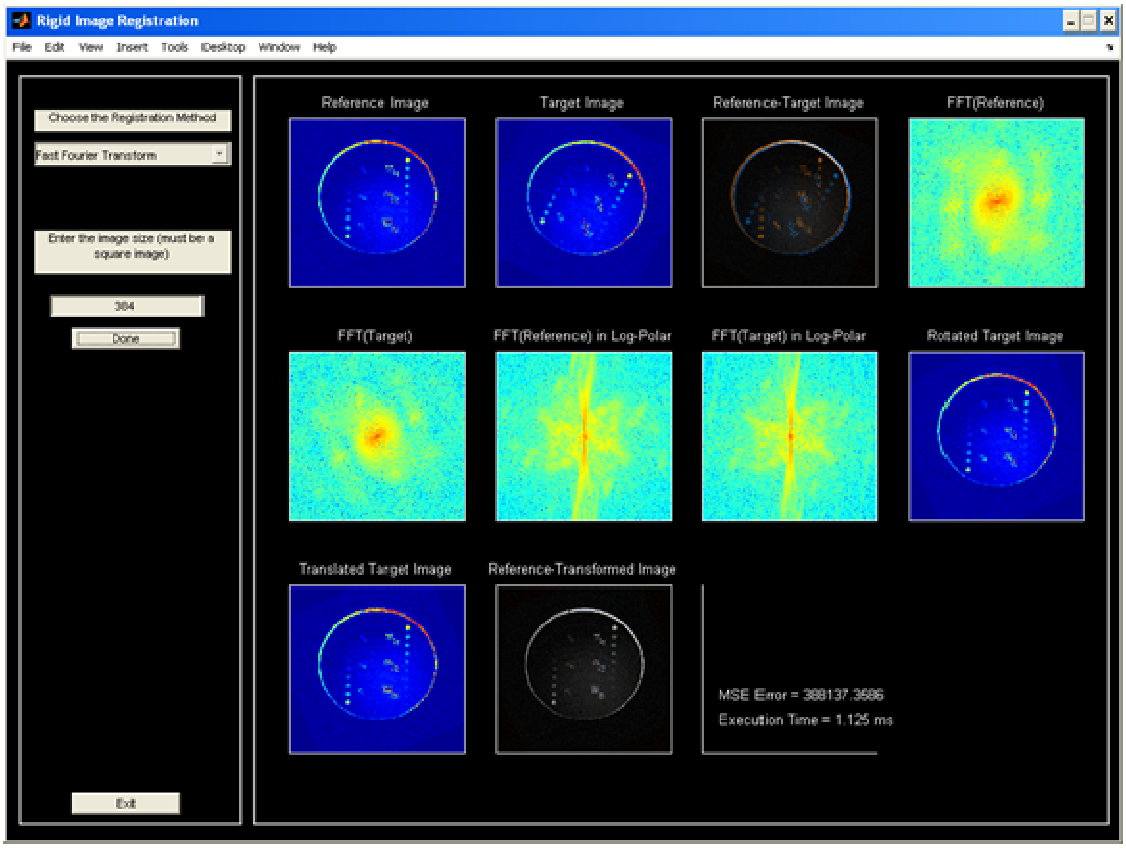


Figure 5: Software Processing Window – FFT Algorithm. The figure shows each processing step after executing FFT Algorithm.

3.4 Output

The aligned or the transformed target image is then saved in the ‘raw’ file format as the input files. Based on the algorithm chosen by the user for registration, the output file name is prefixed with either “PA” or “FR” for using the Principal Axes algorithm or the FFT method respectively. The output file name is kept same as target file name except for the prefix attached as above. The size and format of the output file is kept same as the input target file for any further processing. Examples of two of the output files are as shown in Figure 6 below.

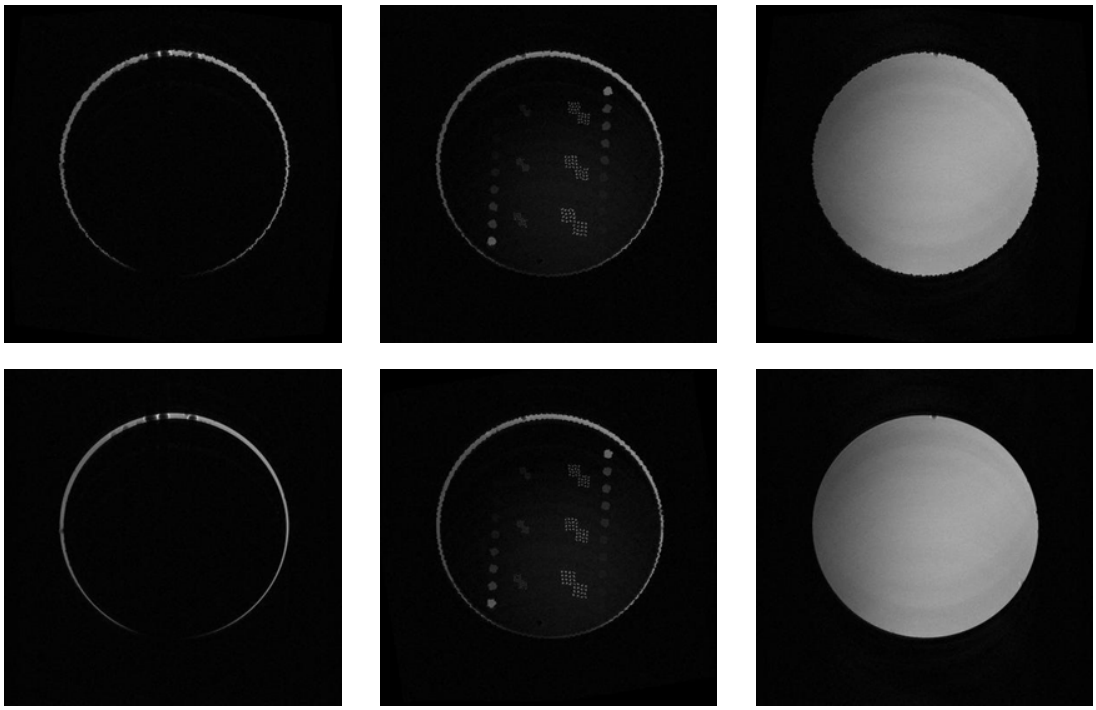


Figure 6: Output image files. Top panel shows the output file PAtarget5.raw and the bottom panel shows FRtarget10.raw file. The leftmost image is the first image slice, middle image is the second image slice and the rightmost image is the third image slice.

CHAPTER 4

PRINCIPAL AXES METHOD

"Discontent is the first necessity of progress."

- Thomas A. Edison

4.1 Introduction

Principal Axes method is a spatial, feature-based monomodal rigid image registration method for aligning two images. The Principal Axes algorithm acts upon the features of the images, such as edges, corners, or circular patterns as its feature space. The search space consists of global translations and rotations. The search strategy is finding the closed formed solution based on the eigenvalue decomposition of a certain covariance matrix [1]. The similarity metric is the variance of the projection of the feature's location vector onto the principal axis [1].

The principal axes are the orthogonal axes about which the moments of inertia are minimized. If two objects are identical except for a translation and a rotation, then they can be registered by coinciding their principal axes [6]. The algorithm is suitable for registering images shaped like an ellipse or ellipsoid. For purposes of image registration, the critical features of an ellipse are its center of mass, and principal orientations, i.e., major and minor axes [1]. The principal axes algorithm is easy to implement, and efficient but it does have the shortcoming that it is sensitive to missing data, if any.

Two images can be aligned using the Principal Axes method by first detecting crucial features of the image and then working on them to align the two images. The features selected for the phantom images are the circular patterns in the image. And the features are detected using thresholding technique. A cutoff intensity value is calculated and all the pixels with lesser value than the cutoff intensity are made zeros in the image.

Next, find the centers of mass of the two images. The rotation angle is calculated by eigenvalue decomposition of the covariance matrices and then finding the angle the maximum eigenvector makes with the horizontal x-axis for each image [1]. The difference in the two angles is the angle by which the target image needs to be reverse rotated about its center for aligning with the reference image.

Next, compute the center of mass of the rotated target image. Find the difference in the two centroids and translate the rotated target image to get the final transformed image.

We have also tested implementing the Principal Axes method with translation as the first step in transformation and rotation being the second step as suggested in the literature. For our application, the success rate we achieved by first rotating the target image and then translating the rotated image to get the final transformed image is higher than implementing it vice-versa (see Appendix E). And hence, we have chosen to rotate the target image first and then translate it to get good alignment. Figure 7 shows the alignment results of implementing translation followed by rotation step in transformation. The comparative results for the same target files by implementing rotation followed by translation step in transformation can be seen in Appendix E.

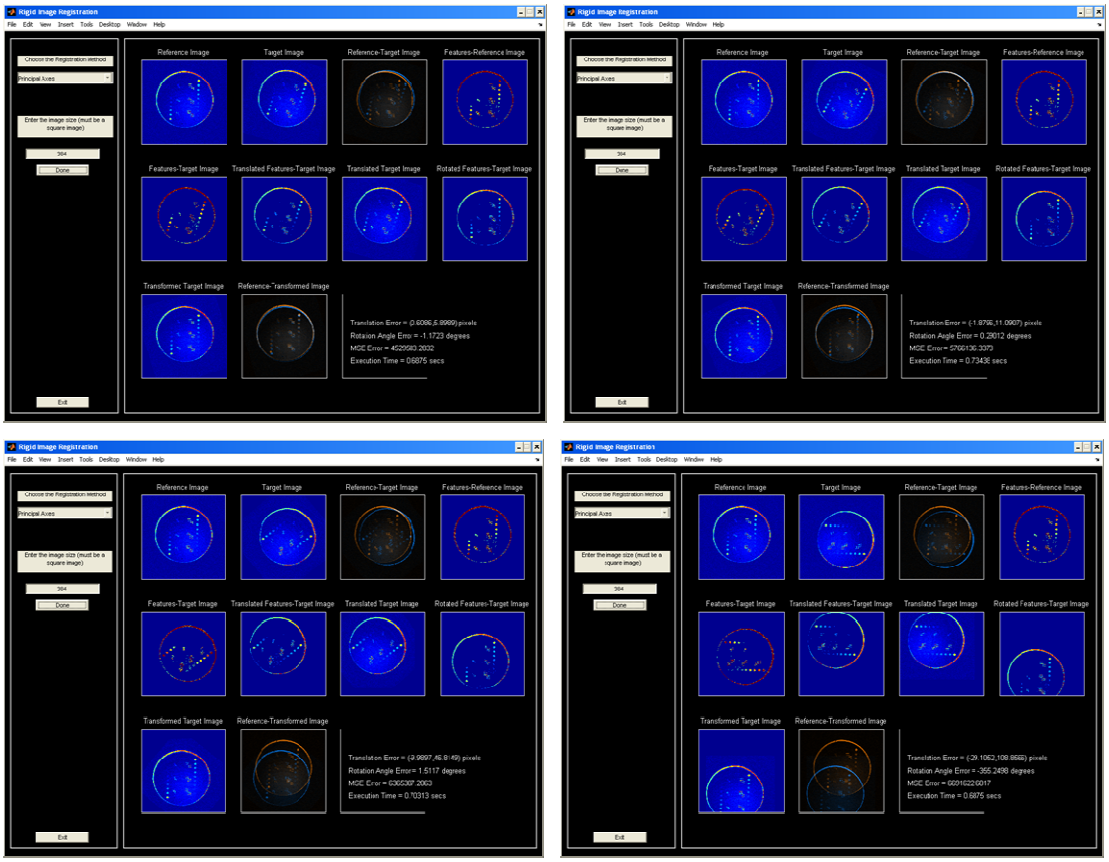


Figure 7: Aligning images by Translation followed by Rotation step for Principal Axes algorithm execution. The top left processing window shows alignment for target15 image. The top right processing window shows alignment for target25 image. The bottom left processing window shows alignment for target50 image. The bottom right processing window shows alignment for target90 image.

4.2 Algorithm

Steps involved for Principal Axes based Registration algorithm:

Let I1 and I2 denote the reference image and the target image respectively.

i) Perform feature detection by thresholding the images to only detect the circular patterns on the phantom image.

   a) Find the maximum intensity values in both images.

   b) Plot a histogram of these intensities into 10 bins.

   c) Look for the intensity for second highest number of pixels in the bins. The highest number of pixels belong to the image background and second highest to the patterns in the image.

   d) Avoid any background pixels by fixing the cutoff for feature selection to one tenth of the number in the earlier step.

   e) Set all the pixel intensities below this cutoff value to be zeros in both images. Let I1F and I2F be the two images with features selected. Use these two images for further processing.

   f) Calculate the centers of both the images with features.

ii) Calculate rotation parameter.

   a) Calculate the center of mass or centroid (X', Y') for I1F and I2F using the formulae $X' = \sum_{x, y} x * I(x, y) / \sum_{x, y} I(x, y)$ and

$$Y' = \sum_{x, y} y * I(x, y) / \sum_{x, y} I(x, y)$$

   where I(x, y) is the intensity at location (x, y).

b) Find the difference between the centroids of target image and the reference image.

c) Find the eigenvectors of the reference image and target image via an eigenvalue decomposition of the covariance matrices. Covariance matrix C can be written as

$$C = \begin{pmatrix} c11 & c12 \\ c21 & c22 \end{pmatrix},$$

where $c11 = \sum_{x, y} (x - X')^2 * I(x, y)$,

$c22 = \sum_{x, y} (y - Y')^2 * I(x, y)$,

$c12 = \sum_{x, y} (x - X') * (y - Y') * I(x, y)$ and

$c21 = c12$.

The eigenvectors of C corresponding to the largest and smallest eigenvalues indicate the direction of the major and minor axes of the eclipse, respectively.

d) For each image, determine the angle *angl*, the maximum eigenvector makes with the horizontal x-axis using the formula

$$angl = \text{atan2}(V(2,1),V(1,1))*180/\pi$$

where V is the maximum eigenvector column [cos(angl); sin(angl)].

e) Find the difference between the two angles, θ.

f) If $\theta <= -90.0$ and $\theta > -180.0$ then $\theta = \theta + 180.0$ else

if $\theta <= 180.0$ then $\theta = \theta + 360.0$

This is the rotation angle in degrees by which the target image needs to be reverse rotated about its center.

iii) Perform Rotation transformation on target image.

a) Construct the rotation matrix as

$$R = [\cos(-\theta) \sin(-\theta); -\sin(-\theta) \cos(-\theta)];$$

b) Multiply the rotation matrix with the (x, y) coordinates of the target image to get the new rotated image coordinates, also adjusting for the centers.

c) Construct the rotated image by matching the coordinates in the rotated matrix with those in the target image.

iii) Calculate translation parameters along x-axis and y-axis.

a) Find the center of mass of rotated target image.

b) Calculate the difference in locations of the two centroids. This is the displacement by which the target image needs to be shifted to align.

iv) Perform translation transformation.

a) Align the centers of mass of the two images by adding the displacements to the original points in the rotated target image.

b) Construct the translated target image by matching the coordinates in the translated image with those in the rotated image from previous step. This is the final transformed aligned target image.

4.3 Experimental Results

Twenty-two of the target images are tested for alignment with the first image in the sequence acquired from the QA phantom, here referred to as the reference image. The Principal Axes algorithm successfully registered seventeen target images with minimal translation and rotation errors.

The algorithm successfully registered the following target files:
target, target0, target00, target5, target10, target15, target20, target25, target50, target90, target_5, target_10, target_15, target_20, target_25, target_50, target_90.

The algorithm failed to align files target150, target180, and target_180. These images are the ones which are misaligned by very large angles (greater than 90 degrees) compared to reference image. In addition, the algorithm also failed to align the noisy images target10n and target25n. This shows the method's sensitivity to presence of noise in images during registration process.

The output screenshots for all the above files are shown in APPENDIX E. Figures 8 and 9 show the best and worst performance results respectively. Figure 10 shows the algorithm's performance in the presence of noise in images.

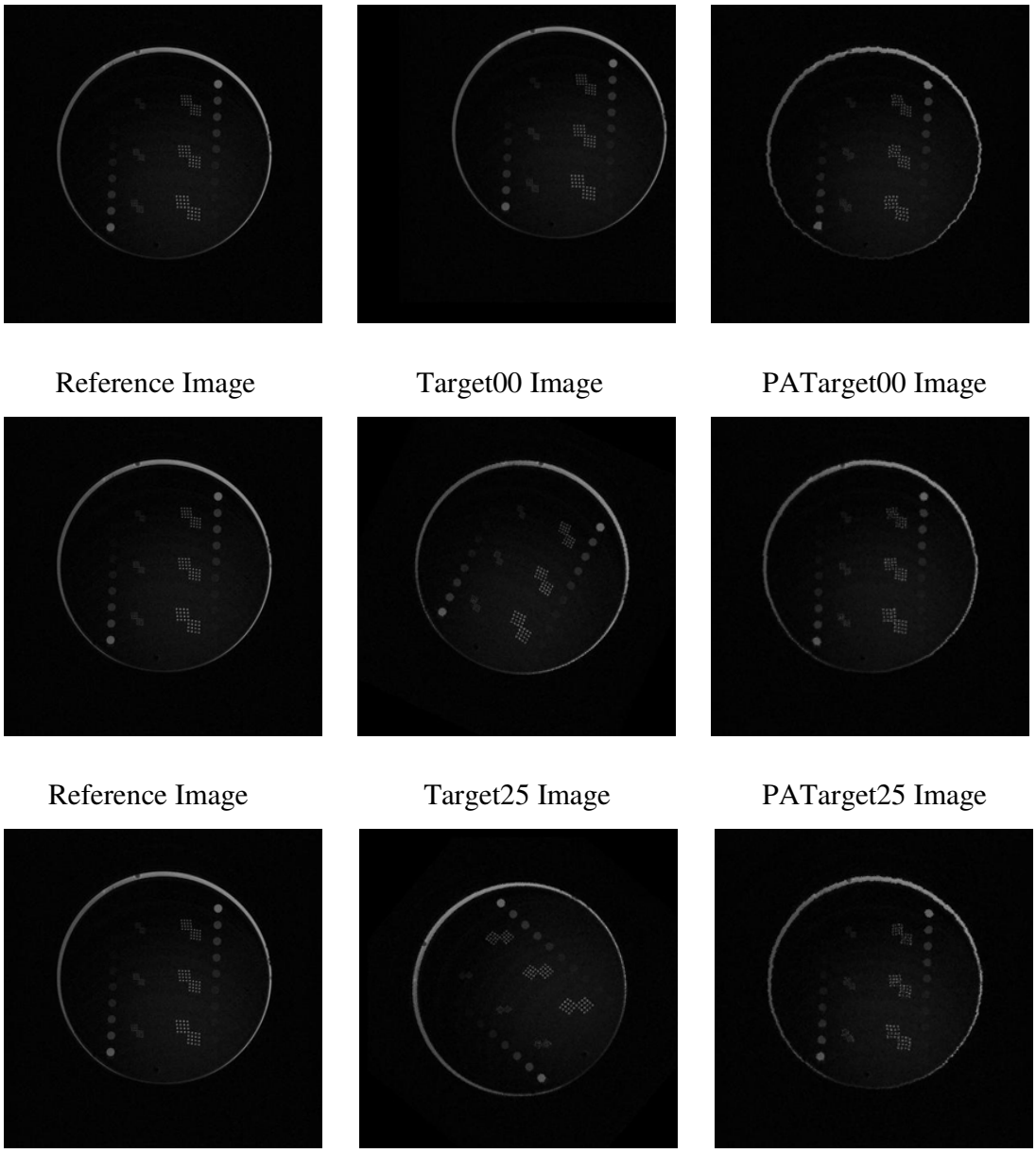
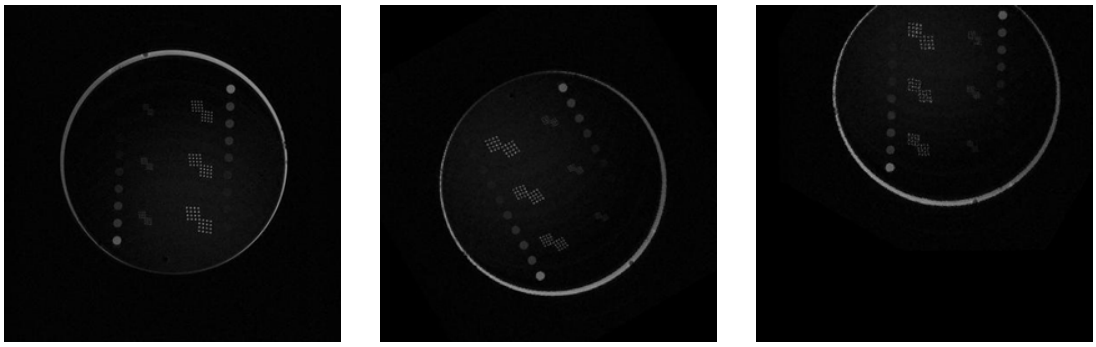
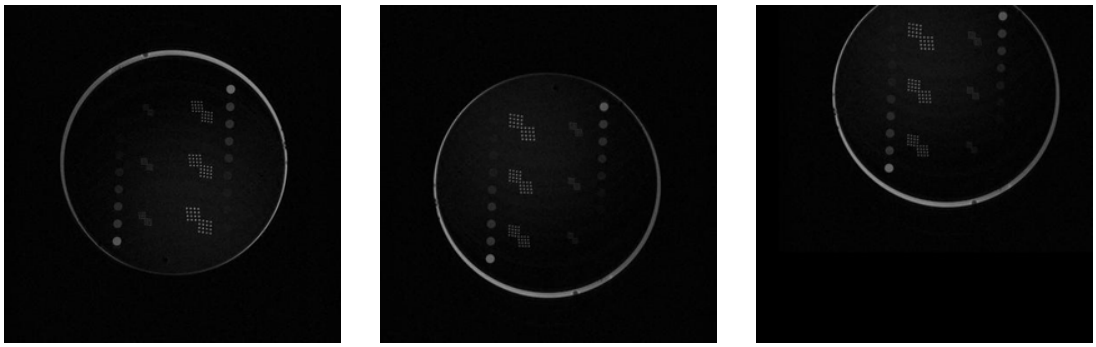


Figure 8: Best performance results from Principal Axes algorithm execution. The leftmost image shows the reference image. Middle image is the original misaligned target image. The rightmost image is the transformed target image.

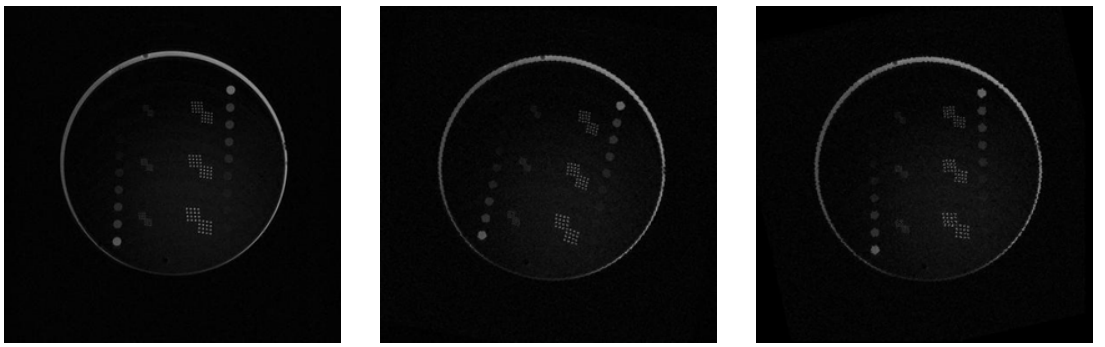


Reference Image Target150 Image PATarget150 Image

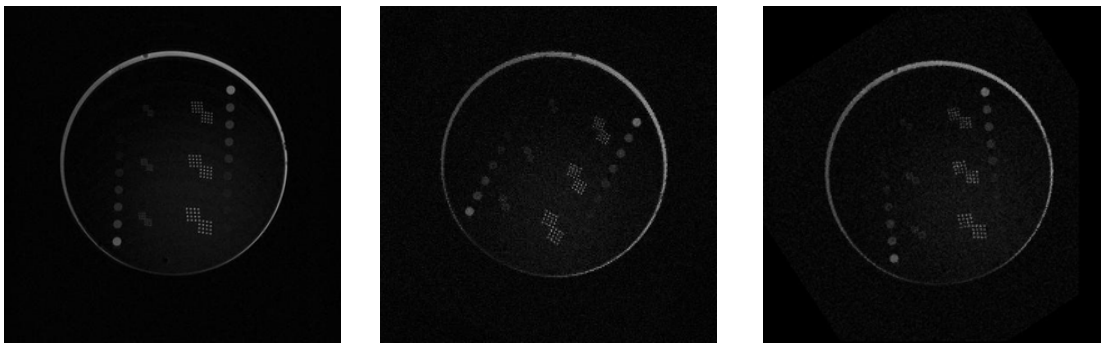


Reference Image Target_180 Image PATarget_180 Image

Figure 9: Worst performance results from Principal Axes algorithm execution. The leftmost image shows the reference image. Middle image is the original misaligned target image. The rightmost image is the transformed target image.



Reference Image Target10n Image PATarget10n Image



Reference Image Target25n Image PATarget25n Image

Figure 10: Alignment using Principal Axes algorithm in the presence of noise. The leftmost image shows the reference image. Middle image is the target image with a Gaussian noise added to the image. The rightmost image is the transformed target image.

4.4 Conclusion

The registration trend shows that the algorithm is successful in registering images displaced by smaller to larger distance and rotated up to 90 degrees with respect to the reference image. But the algorithm performs poorly when the two images differ by large rotation angle of greater than 90 degrees.

In the worst case, prior reorienting the target image with respect to the reference image using ImageJ and then using it as input will take care of the limitation of our algorithm.

The algorithm is sensitive to presence of noise in target images and does not align the images as well as compared to in the absence of noise.

CHAPTER 5

FAST FOURIER TRANSFORM METHOD

"Always bear in mind that your own resolution to succeed is more important than any other."

- Abraham Lincoln

5.1 Introduction

The Fourier Transform is an important image processing tool used for representing an image into its sine and cosine components [16]. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is its spatial domain equivalent [16]. Each point in the frequency domain represents a particular frequency contained in the spatial domain image.

The Fast Fourier Transform (FFT) based registration is a frequency-domain type automatic rigid registration method. The feature space it uses consists of all the pixels in the image, and its search space covers all global translations and rotations [1]. The search strategy is the closed form Fourier-based method, and the similarity metric is correlation, and its variants, e.g., phase only correlation [1].

The log-polar transformation is a nonlinear and non-uniform sampling of the spatial domain [9]. Nonlinearity is because of polar mapping, while non-uniform sampling is the result of logarithmic scaling [9].

Consider the log-polar (log r, θ) coordinate system, where *r* is the angle denoting radial distance from the center (x_c, y_c). Any (x, y) point can be represented in polar coordinates (r, θ) [9] as

$$r = \log\left(\sqrt{(x - x_c)^2 + (y - y_c)^2}\right)$$

$$\theta = \tan^{-1}((y - y_c) / (x - x_c))$$

The Fourier–Mellin registration method is based on the principle of phase correlation and the properties of Fourier analysis [9]. The phase correlation finds the translation between two images. The Fourier–Mellin transform extends phase correlation to handle images transformed by both translation and rotation [9].

The FFT-based automatic registration method relies on the Fourier shift theorem which guarantees that the phase of a specially defined "ratio" is equal to the phase difference between the images [7]. The Fourier-Mellin transform [7] has been implemented in our algorithm to register images that are misaligned due to translation and rotation. A Fourier transform is applied to images to recover translation [9]. Then a log-polar transformation is applied to the magnitude spectrum and the rotation angle is recovered by using phase correlation in the log-polar space [9]. By operating on the magnitude spectrum of an image, the translational differences between the images are avoided since the magnitude spectrum of an image and its translated counterpart are identical and only their phase spectrum are different [9]. The log-polar transformation causes rotation to be manifested as translation, whereby phase correlation can be applied to recover the rotation angle between the pair of input images [9].

5.2 Algorithm

Steps involved for Fourier Transform based Correlation Registration algorithm:

Let I1 and I2 are the reference and target images respectively.

i) Calculate rotation parameter.

a) Apply FFT function to images I1 and I2, resulting in F1 and F2.

b) Transform the Cartesian coordinate system points (x, y) into log-polar coordinates (log (p), θ) using formulae

Log (p) = log (sqrt($x^2 + y^2$)) and

θ = atan(y/x).

c) Find the new intensity values at the corresponding log-polar coordinates using bilinear interpolation.

d) Apply FFT to log-polar images I1 and I2, resulting in Flp1 and Flp2.

e) Compute the ratio

R1 = (Flp1 * conj(Flp2)) / (abs(Flp1 * conj(Flp2)))

where conj is the complex conjugate and abs is the absolute value. The ratio defines the cross-power spectrum. ie the phase difference between the two images.

f) Compute the inverse FFT of R1 as IR1.

g) Find the location *loc* in log-polar coordinates for maximum value of abs (IR1).

h) Calculate the corresponding point (x_o, y_o) from *loc*.

x_o = mod (loc, cols);

y_o = loc / rows;

Rotation angle θ in radians is the y-displacement in the log-polar coordinate.

ii) Perform rotation transformation on the target image.

a) Construct the rotation matrix as

R = [cos(-θ) sin(-θ); -sin(-θ) cos(-θ)];

b) Multiply the rotation matrix with the (x, y) coordinates of the target image to get the new rotated image coordinates, also adjusting for the centers.

c) Construct the rotated image I3 by matching the coordinates in the rotated matrix with those in the target image.

iii) Calculate translation parameters along x-axis and y-axis.

a) Apply FFT function to image I3, resulting in F3.

b) Compute the ratio

R1 = (F1 * conj(F3)) / (abs(F1 * conj(F3))) .

c) Compute the inverse FFT of R1 as IR1.

d) Find the location *loc* for maximum value of abs(IR1).

e) Find the translation point (x_o, y_o) using formulae

x_o = mod (loc, cols);

y_o = loc / rows;

iv) Perform translation transformation on rotated target image.

a) Align image I3 by adding the displacements to the original points in the rotated target image.

b) Construct the translated target image by matching the coordinates in the translated image with those in the rotated image from previous step. This is the final transformed aligned target image.

5.3 Experimental Results

Twenty-two of the target images are tested for alignment with the first image in the sequence acquired from the QA phantom, here referred to as the reference image. The Fast Fourier Transform algorithm successfully registered twenty-one target images with minimal translation and rotation errors between the two images.

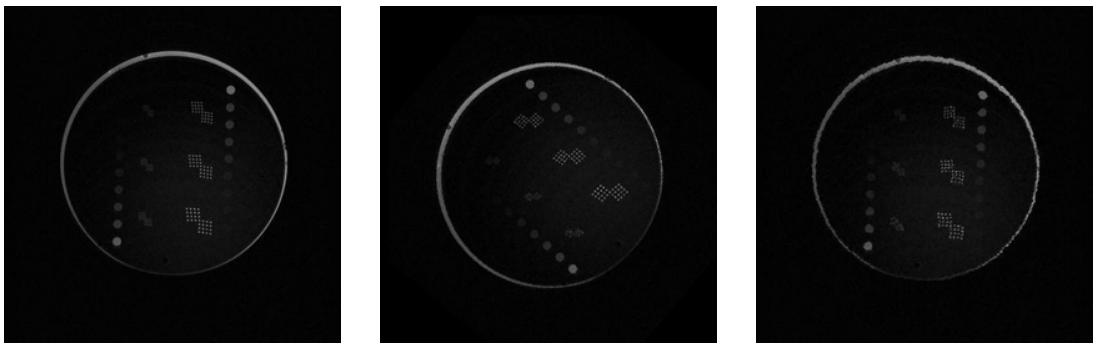
The algorithm successfully registered the following target files:

target, target0, target5, target10, target15, target20, target25, target50, target90, target150, target180, target_5, target_10, target_15, target_20, target_25, target_50, target_90, target_180, target10n, target25n.

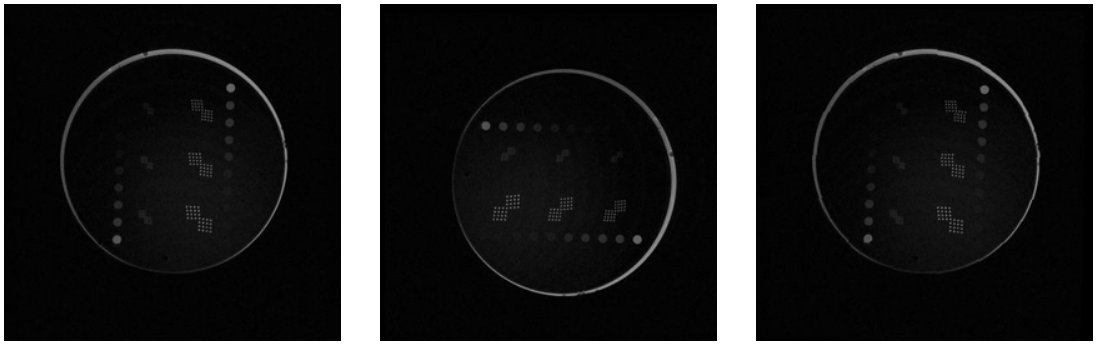
The algorithm successfully registered target images target10n and target25n in the presence of noise showing its insensitivity to the presence of noise in images during registration process.

The algorithm failed to align image target00 which was largely displaced.

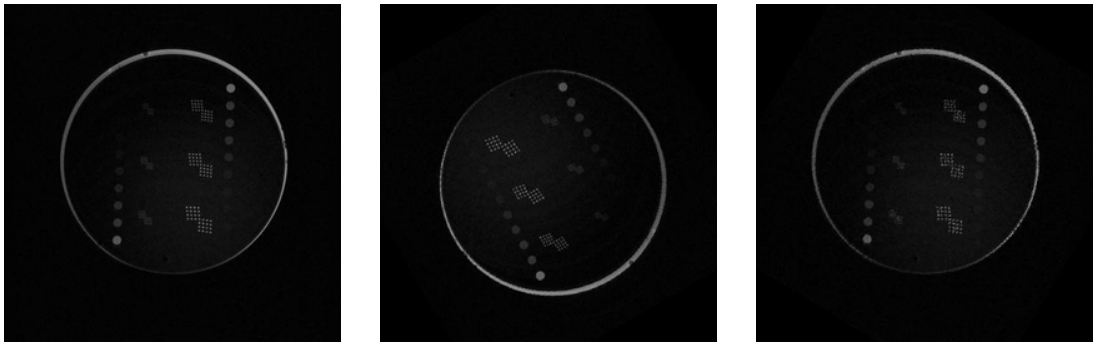
The output screenshots for all the above target files are shown in APPENDIX F. Figures 11 and 12 show the best and worst performance results respectively. Figure 13 shows the result of registering the noisy images.



Reference image Target_50 image FRTarget_50 image

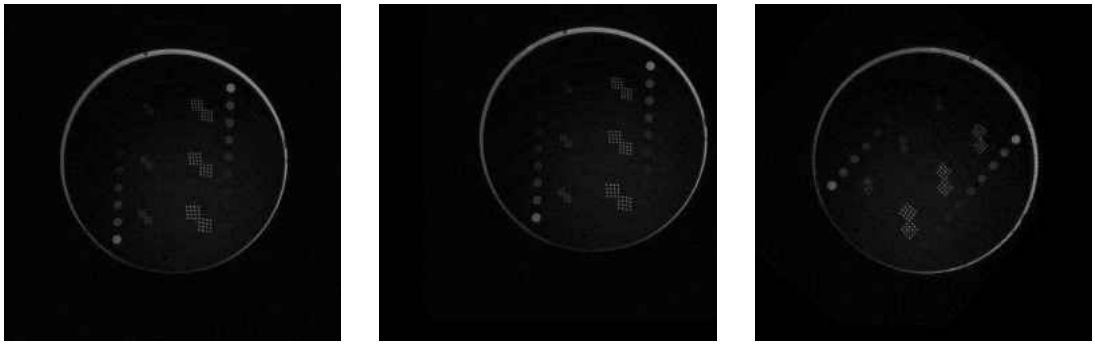


Reference image Target90 image FRTarget90 image



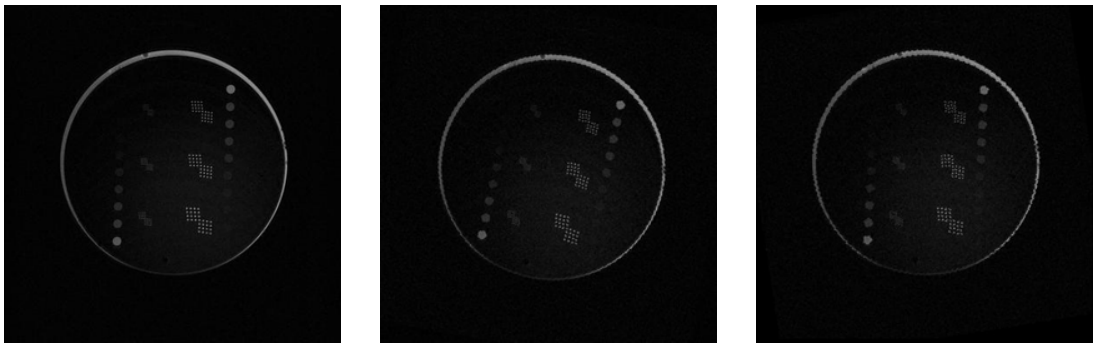
Reference image Target150 image FRTarget150 image

Figure 11: Best performance results from FFT algorithm execution. The leftmost image shows the reference image. Middle image is the original misaligned target image. The rightmost image is the transformed target image.

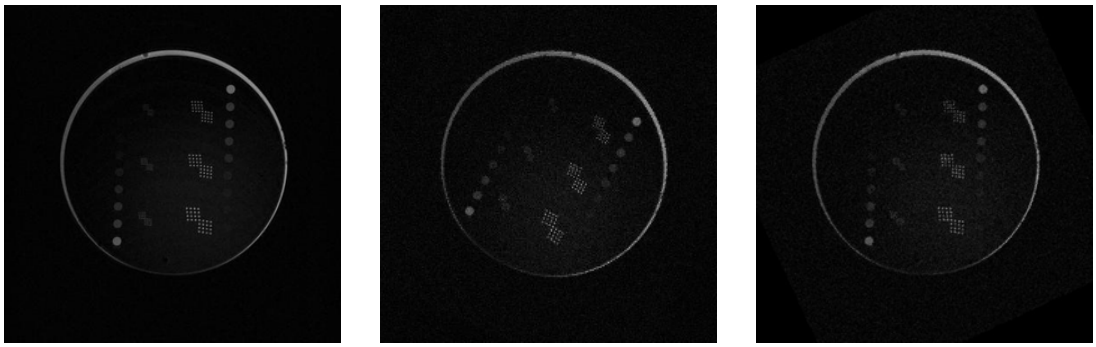


Reference Image Target00 Image FRTarget00 Image

Figure 12: Worst performance results from FFT algorithm execution. The leftmost image shows the reference image. Middle image is the original misaligned target image. The rightmost image is the transformed target image.



Reference Image Target10n Image FRTarget10n Image



Reference Image Target25n Image FRTarget25n Image

Figure 13: Alignment using FFT algorithm in the presence of noise. The leftmost image shows the reference image. Middle image is the target image with a Gaussian noise added to the image. The rightmost image is the transformed target image in the presence of noise.

5.4 Conclusion

The registrations achieved by using the FFT algorithm show that the algorithm successfully aligns images rotated by any angle difference up to 180 degrees. The algorithm performed poorly when the two images are displaced by more than 25 pixels with no rotation difference between the two images.

In the worst case, reorienting the target image with respect to the reference image either by translation or rotation using ImageJ and then using it as input will take care of the limitation of our algorithm.

The algorithm is not sensitive to presence of noise in target images and successfully aligns the images as well as compared to in the absence of noise.

CHAPTER 6

SIMILARITY MEASURES

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

- Alan Turing

6.1 Introduction

Image similarities are broadly used in medical imaging. An image similarity measure quantifies the degree of similarity between intensity patterns in two images [11] [12]. The choice of an image similarity measure depends on the modality of the images to be registered.

Common examples of image similarity measures include cross-correlation, mutual information, sum of squared intensity differences, sum of absolute differences, mean square error and ratio image uniformity [11]. Mutual information and normalized mutual information are the most popular image similarity measures for registration of multimodality images [12]. Cross-correlation, sum of squared intensity differences, sum of absolute differences, mean square error and ratio image uniformity are commonly used for registration of images in the same modality [11].

We have used the popular error metrics method Mean Square Error (MSE) for measuring similarity between the reference image and the transformed target image. The

MSE is the cumulative squared error between the transformed image and the reference image. The mathematical formula for measuring MSE is

$$MSE = 1/MN * \sum^{M}_{y=1} \sum^{N}_{x=1} [I(x, y) - I'(x, y)]^2$$

where I(x, y) is the reference image, I'(x, y) is the transformed image, and M,N are the dimensions of the images. A lower value for MSE means lower similarity error and higher similarity between the two images.

For the Principal Axes method, we have also measured translation and rotation differences between the reference image and the transformed target image to get the translation error and the rotation angle error. This gives us a realistic picture of alignment mismatch extent as can also be validated by looking at the difference between the images. Translation error between the reference image and the transformed image is measured by finding the centroids of both the images and finding its difference. An exact alignment should have both centroid locations at the same coordinates. Similarly, rotation angle error is calculated by finding the maximum eigenvectors of each image and calculating the angle it makes with the horizontal x-axis. The difference in the angles is the rotation angle error.

Measuring the translation and rotation errors for confirming the accuracy of alignment by using FFT method is computationally expensive as it requires additional computation of six Fast Fourier Transforms. Hence, MSE is used for measuring the similarity.

The difference between the reference image and the transformed image is shown to the user by subtracting the two images and displaying the difference image in the

processing window. A perfect alignment results in difference image shown in gray scale and any misalignments if exist are shown either in red or blue color in the image. This gives us a realistic picture of alignment mismatch extent as can be validated by looking at the difference image.

6.2 Analysis Results

MSE values showed a value of zero for exact alignment between images for the Principal Axes method. For similar alignments achieved, the MSE values ranged approximately from 1.5e4 to 2.0e6 and for misalignments, the MSE values approximated in the range of 5.9e6 to 6.4e6.

For the FFT method, the MSE values ranged approximately from 1.1e3 to 9.9e5 and for misalignments, the MSE values are shown to be higher in the approximate range of 1.3e6 to 2.5e6.

Figure 14 shows the similarity measure distribution for each target image registered using both methods.

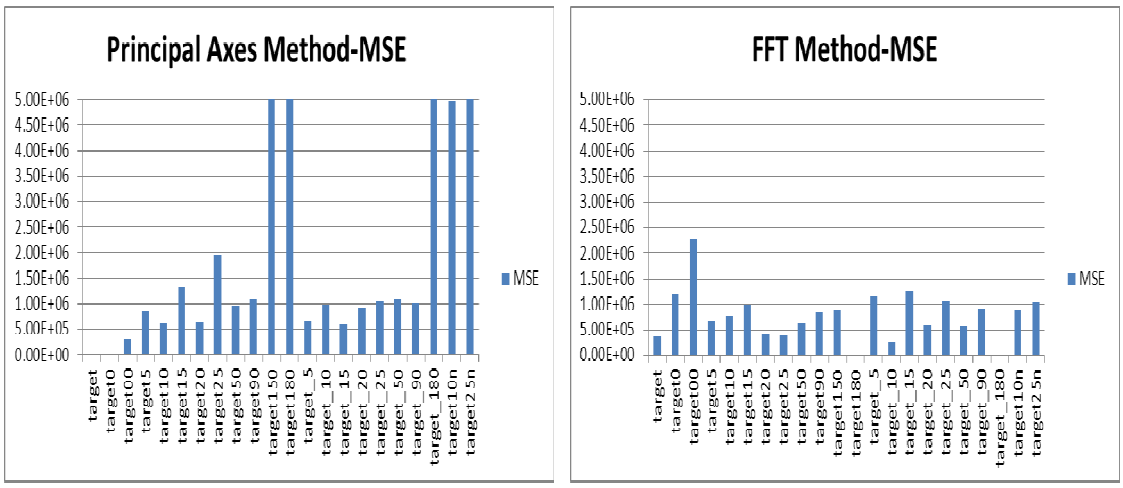


Figure 14: Similarity Measure MSE computed for image registrations. Left panel shows MSE distribution for using the Principal Axes method. Right panel shows MSE distribution for using the FFT method.

Translation and rotation errors are calculated for registering images using the Principal Axes method. Figure 15 shows their distribution for each target image. Translation errors measured along x-axis and y-axis are less than 1 pixel for all the successful registrations and rotation angle error is less than 1.5 degrees.

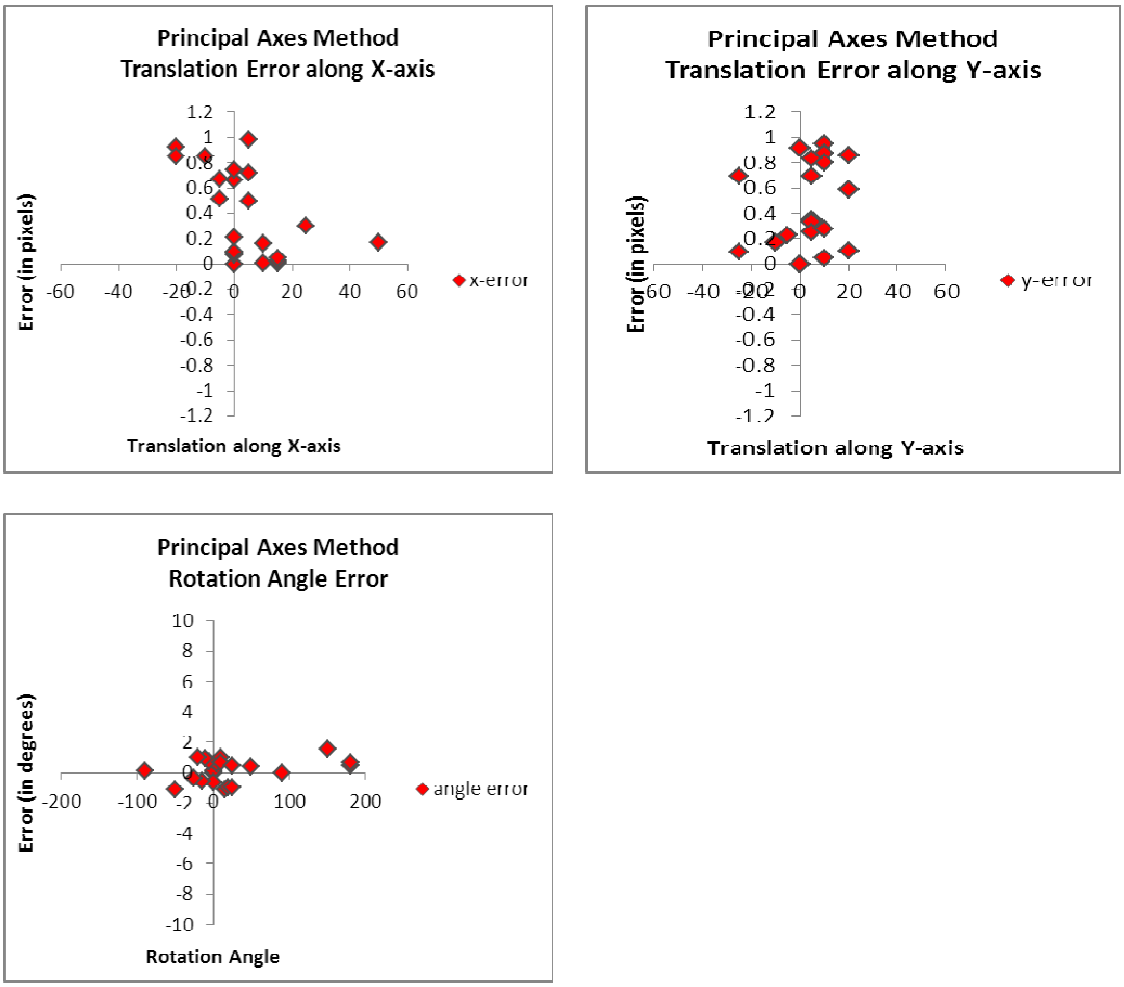


Figure 15: Translation and Rotation Errors computed for Principal Axes method. Top panels show translation error distribution along x-axis and y-axis. Bottom panel shows rotation angle error distribution.

6.3 Conclusion

Based on the Mean Square Error similarity measure analysis, the FFT method performed better in aligning the phantom images with MSE values ten times lesser than those for the Principal Axes method showing more similarity between the reference image and the transformed target image for successful registrations.

Unlike the Principal Axes method, the FFT method successfully registered the two images in the presence of noise.

Translation and rotation errors calculated for all successful registrations using the Principal Axes method give minimum errors of up to 1 pixel and 1.5 degrees respectively showing better similarities achieved in all the alignments.

The Principal Axes method successfully registered 17 of the 22 non-aligned test images, the FFT method registered 21 test images whereas using the 'Realign' tool in the existing SPM8 package with default settings showed correct alignments for only 9 images as per our requirement. Please refer to Appendices D, E and F for registration screenshots captured for registering images using SPM8, Principal Axes algorithm, and the FFT algorithm respectively.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

"A conclusion is the place where you got tired of thinking.

- Martin Fischer

QA application in MRI field requires registering images acquired from the phantom from time to time before the QA parameters can be measured to test the image quality and hence the proper functioning of the scanner. In practice, the images will be transformed no more than 50 pixels and 50 degrees angle with respect to the first image in the series. Based on our results, we conclude that the image registration software developed by us suits our application based on our requirements.

The Principal Axes method is successful in aligning images which are displaced by smaller to larger distance (tested up to 50 pixels) and rotated up to 90 degrees with respect to the reference image. This method is simple, faster and computationally inexpensive.

The FFT method successfully performs alignment of images which are rotated by an angle between 0 to 180 degrees with respect to the reference image. Registration using FFT method requires computing six Fast Fourier Transforms for both images. Hence this method is computationally expensive but very efficient.

The user can use any of the two algorithms to perform the image alignment process and choose the results with a better alignment. Also, if any algorithm gives

coarse similarity between the images after registration, then the user may realign this output image using the other algorithm to get better alignment results.

Once the phantom images are aligned using our software, the future work in our QA process involves using these aligned images to measure and compute different QA parameters such as Signal to Noise Ratio, Contrast to Noise Ratio, Ghosting Fraction, Resolution, Magnetic Homogeneity and so on. The comparative study of these QA parameters for the images acquired over time will help us in knowing how these parameters vary with time for each image thus giving us an idea about how image quality is getting affected over time. A comparative study of the image quality will also allow us to see if the change is due to malfunctioning of the scanner magnet.

REFERENCES

1. Peter J. Kostelec and Senthil Periaswamy, Image Registration for MRI, Modern Signal Processing, MSRI Publications, Volume 46, 2003.

2. J. Ashburner and K. Friston, Rigid body registration, The Wellcome Dept. of Imaging Neuroscience, 12 Queen Square, London WC1N 3BG, UK.

3. N. M. Alpert, J. F. Bradshaw, D. Kennedy, and J. A. Correia, The Principal Axes Transformation - A Method for Image Registration, J NucIMed 1990;31:1717-1722.

4. Robin Kramer, Automatic MRI Image Registration Using Phase Correlation, Department of Computer Science, University of Wisconsin, Madison, United States.

5. B. Antoine Maintz , Max A. Viergever, An Overview of Medical Image Registration Methods, Symposium of the Belgian Hospital Physicists Association, 1996.

6. Medha V. Wyawahare, Dr. Pradeep M. Patil, and Hemant K. Abhyankar, Image Registration Techniques: An overview, International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 2, No.3, September 2009.

7. Hongjie Xie, Nigel Hicks, George R. Keller, Haitao Huang, Vladik Kreinovich, An IDL/ENVI Implementation of the FFT Based Algorithm for Automatic Image Registration, Computers and Geosciences, 2003, Vol. 29, No. 8, pp. 1045-1055.

8. B. Srinivasa Reddy and B. N. Chatterji, An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration, Proc. of IEEE Transactions On Image Processing, Vol. 5, No. 8, August 1996.

9. George Wolberg, Siavash Zokai, Robust Image Registration Using Log-Polar Transform, Proc. of IEEE Intl. Conf. on Image Processing, Sep. 2000.

10. Cynthia Rodriguez, Understanding FFT- based algorithm to calculate image displacements with IDL programming language, University of Texas at San Antonio, 2007.

11. A. Ardeshir Goshtasby. 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications, Wiley Press, 2005.

12. http://en.wikipedia.org/wiki/

13. Friston KJ, Holmes AP, Worsley KJ, Poline JP, Frith CD, Frackowiak RSJ. Statistical parametric maps in functional imaging: a general linear approach. Human Brain Mapping. 1994; 2(4):189–210.

14. Abràmoff, M.D., Magalhães, P.J. and Ram, S.J. Image Processing with ImageJ. Biophotonics International, 11(7):36—42, 2004.

15. The MathWorks, Inc., MA, USA.

16. http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

APPENDIX A

USER INTERFACE SCRIPT IN MATLAB

```
% Script to design a GUI interface for image registration

function IMAGE_REGISTER(~,~)

   clc;
   clear all;
   close all;

   scrsz = get(0,'ScreenSize');
   h = figure('Name','Rigid Image Registration','NumberTitle','off',...
       'OuterPosition',[scrsz(1) scrsz(2) scrsz(3)
          scrsz(4)],'Resize','Off');
   set(h,'Color',[0 0 0]);
   clf(1);

   hp = uipanel('Parent',h,'BorderType','etchedin',...
             'BackgroundColor','black',...
             'Position',[.01 .02 .2 0.96]);
   hsp = uipanel('Parent',h,'BorderType','etchedin',...
              'BackgroundColor','black',...
              'Position',[.22 .02 .77 0.96]);
   uicontrol('Style','text',...
        'Position',[25 650 180 20],...
        'String','Choose the Registration Method');
   uicontrol('Style', 'popup',...
           'String', 'Principal Axes|Fast Fourier Transform',...
           'Position', [25 620 180 20],...
           'Callback', {@getFiles});
   uicontrol('Style', 'pushbutton',...
           'String', 'Exit',...
           'Position', [60 25 100 20],...
           'Callback', 'exit');
   ah = axes('Parent',h,'Position',[.23 .024 .75 .95],'Visible','Off');
   colordef black;
   cla;

   img=imread('f:/Qa_thesis/reference.jpg');
   image(img,'Clipping','On')
   set(gca,'xtick',[],'ytick',[]);

end

%*********************************************************************
```

```
% function to let user choose the files and image size
function getFiles(hObj,event)

    cla;
    ah = axes('Position',[.23 .024 .75 .95],'Visible','Off');
    colordef black;
    img=imread('f:/Qa_thesis/reference.jpg');
    image(img,'Clipping','On')
    set(gca,'xtick',[],'ytick',[]);

    val=get(hObj,'Value')

    [f1,pth,filter]=uigetfile({'*.raw'},'Choose the reference image');
    [f2,pth,filter]=uigetfile({'*.raw'},'Choose the target image to
            register');
    ref_file=[pth f1];
    target_file=[pth f2];

    uicontrol('Style','text',...
        'Position',[25 520 180 40],...
        'String','Enter the image size (must be a square image)');

    eb=uicontrol('Style', 'edit',...
              'Position', [40 480 140 20],...
              'String', 'Enter width of image');
    uicontrol(eb);

    uicontrol('Style', 'pushbutton',...
           'String', 'Done',...
           'Position', [60 450 100 20],...
           'Callback',{@register,ref_file,target_file,val});

%*********************************************************************

% function to open files and call registration algorithms
function register(hObj,event,ref_file,target_file,val)

    val1=get(hObj,'Value')
    if val1==1
       dd=get(eb,'String')
    end

    dim1=str2num(dd);
    dim2=dim1;
    dim=[dim1 dim2 3];

    % read the raw QA_phantom reference and target image files
    fqa = fopen(ref_file,'r');
    ref = reshape(fread(fqa,'uint16','b'),dim);
    fclose(fqa);
    fqa = fopen(target_file,'r');
    target = reshape(fread(fqa,'uint16','b'),dim);
    fclose(fqa);
```

```
    rf=ref(:,:,2);
    tg=target(:,:,2);

    subplot(3,4,1.6);
    imagesc(rf');
    set(gca,'xtick',[],'ytick',[]);
    title('Reference Image');

    subplot(3,4,2.5);
    imagesc(tg');
    set(gca,'xtick',[],'ytick',[]);
    title('Target Image');

    % Subtract images to show initial difference, aligned regions show
    % in grayscale
    image_diff(:,:,1) = (rf-min(rf(:)))/ ...
                         (max(rf(:))-min(rf(:)));
    image_diff(:,:,3) = (tg-min(tg(:)))/ ...
                         (max(tg(:))-min(tg(:)));
    image_diff(:,:,2) = (image_diff(:,:,1)+image_diff(:,:,3))/2.0;

    image_diff=permute(image_diff,[2 1 3]);

    subplot(3,4,3.4);
    imagesc(image_diff);
    set(gca,'xtick',[],'ytick',[]);
    title('Reference-Target Image');

%*********************************************************************
% CHOOSE ALGORITHM
    if(val==1)
       aligned = Principal_Axes_Algorithm(ref,target,dim);
    elseif(val==2)
       aligned = Fourier_Transform_Algorithm(ref,target,dim);
    end

%*********************************************************************
% WRITE THE ALIGNED FILE
    % Write the aligned file to a raw file with the same name as target
    % file and prefixed with 'FR' for using FFT algorithm and with 'PA'
    % for using Principal Axes algorithm
    if val == 1
        fout = [pth 'PA' f2];
    else
        fout = [pth 'FR' f2];
    end
    fid = fopen(fout,'w');
    fwrite(fid,aligned,'uint16','b');
    fclose(fid);
  end
end
%*********************************************************************
```

APPENDIX B

PRINCIPAL AXES SCRIPT IN MATLAB

```
% Script to realign images using Principal Axes Algorithm
% MATLAB function comments are taken from MATLAB documentation

function target = Principal_Axes_Algorithm(ref,target,dim)
    t=cputime
       % FEATURE DETECTION FOR REFERENCE IMAGE
       % Define a minimum threshold to detect features
       rf2=ref(:,:,2);
       tg2=target(:,:,2);

       jnk=rf2;
       maxb=max(jnk(:));
       n=hist(jnk(:),[0:maxb/10:maxb]);
       phtm_imax=find(n==max(n(2:length(n))));
       cutoff=n(phtm_imax)*1/10;
       jnk2=zeros([dim(1) dim(2)]);
       idx=find(jnk(:)<cutoff);

       % Make zero all the pixel intensities lesser than the cutoff
       rf=rf2;
       rf(idx)=0;

       idy=1:dim(1)*dim(2);
       yidx=fix((idy-1)/dim(1))+1;
       xidx=idy-(yidx-1)*dim(1);

       % Find center of mass for reference image
       ref_center=[sum(xidx(:).*rf(:))/sum(rf(:)) ...
                   sum(yidx(:).*rf(:))/sum(rf(:))]

       subplot(3,4,4.3);
       imagesc(rf',[0 8000]);
       set(gca,'xtick',[],'ytick',[]);
       title('Features-Reference Image');

       % FEATURE DETECTION FOR TARGET IMAGE
       % Define a minimum threshold to detect features
       jnk=tg2;
       maxb=max(jnk(:));
       n=hist(jnk(:),[0:maxb/10:maxb]);
       phtm_imax=find(n==max(n(2:length(n))));
       cutoff=n(phtm_imax)*1/10;
       jnk2=zeros([dim(1) dim(2)]);
       idx=find(jnk(:)<cutoff);
```

```
        % Make zero all the pixel intensities lesser than the cutoff
        tg=tg2;
        tg(idx)=0;

        idy=1:dim(1)*dim(2);
        yidx=fix((idy-1)/dim(1))+1;
        xidx=idy-(yidx-1)*dim(1);

        % Find center of mass for target image
        target_center=[sum(xidx(:).*tg(:))/sum(tg(:)) ...
                       sum(yidx(:).*tg(:))/sum(tg(:))]

        subplot(3,4,5.6);
        imagesc(tg',[0 8000]);
        set(gca,'xtick',[],'ytick',[]);
        title('Features-Target Image');


%************************************************************************
    % Form the covariance matrix ref_C
    c11=0;
    c12=0;
    c21=0;
    c22=0;
    for i = 1:dim(1)
        for j = 1:dim(2)
            c11 = c11 + (i - ref_center(1)) ^ 2 * rf(i,j);
            c22 = c22 + (j - ref_center(2)) ^ 2 * rf(i,j);
            c12 = c12 + (i - ref_center(1)) * (j - ref_center(2)) *
                                rf(i,j);
        end
    end
    c21 = c12;
    ref_C = [c11 c12; c21 c22];

    % Find the largest eigen values and the eigen vectors for
    % covariance matrix A
    % V is the modal matrix composed of eigenvectors
    % L is the diagonal matrix with eigenvalues on its diagonal
    [ref_V,ref_L]= eigs(ref_C);

    % find ref_theta, the angle between the eigenvector and the x-axis
    % for reference image,
    % first column of  ref_V=[cos(theta);sin(theta)]
    ref_theta = atan2(ref_V(2,1),ref_V(1,1))*180/pi;

    % Form the covariance matrix target_C
    c11=0;
    c12=0;
    c21=0;
    c22=0;
    k=1;
    t_xy=ones(dim(1)*dim(2),2);
```

```
    for i = 1:dim(1)
        for j = 1:dim(2)
            c11 = c11 + (i - target_center(1)) ^ 2 * tg(i,j);
            c22 = c22 + (j - target_center(2)) ^ 2 * tg(i,j);
            c12 = c12 + (i - target_center(1)) * (j - target_center(2))
                        * tg(i,j);
            t_xy(k,1)=j;
            t_xy(k,2)=i;
            k=k+1;
        end
    end
    c21 = c12;
    target_C = [c11 c12; c21 c22];

    % Find the largest eigen values and the eigen vectors for
    % covariance matrix A
    % V is the modal matrix composed of eigenvectors
    % L is the diagonal matrix with eigenvalues on its diagonal
    [target_V,target_L]= eigs(target_C);

    % find target_theta, the angle between the eigenvector and the x-
    % axis for
    % target image. %first column of target_V=[cos(theta);sin(theta)]
    target_theta = atan2(target_V(2,1),target_V(1,1))*180/pi;

    % find difference in angles in degrees
    angl=target_theta-ref_theta
    if angl <= -90.0 && angl > -180.0
        angl=angl+180.0;
    elseif angl <= 180.0
        angl=angl+360.0;
    end


% Find the difference between the rotation angles for both images in
% radians
rot_theta = angl*pi/180.0;

%*********************************************************************
     % TRANSFORMATION

     % Rotation matrix
        R = [cos(-rot_theta) sin(-rot_theta);-sin(-rot_theta) cos(-
                  rot_theta)];

        RR = ones(dim(1)*dim(2),2);
        temp = ones(2,1);

        % Apply rotation parameters on original coordinates in t_xy in
        % center
        for i = 1 :dim(1)*dim(2)
           tt=t_xy(i,:);
```

```
        tt(1)=tt(1)-target_center(1);
        tt(2)=tt(2)-target_center(2);
        temp = R * tt';
        temp(1)=temp(1)+target_center(1);
        temp(2)=temp(2)+target_center(2);
        RR(i,:) = temp';
    end

    RR = reshape(RR,[dim(1) dim(2) 2]);
    RR = round(RR); % RR has the rotated target image coordinates
    aligned_data1 = zeros(dim(1),dim(2),3);
    ad1=zeros(dim(1),dim(2));

    % construct the rotated image
    for i=1:dim(1)
        for j=1:dim(2)
            if(RR(i,j,1)>0 & RR(i,j,1)<dim(1)+1 & RR(i,j,2)>0 &
                    RR(i,j,2)<dim(2)+1)
                ad1(i,j) = tg(RR(i,j,1),RR(i,j,2));
                aligned_data1(i,j,1) =
                    target(RR(i,j,1),RR(i,j,2),1);
                aligned_data1(i,j,2) =
                    target(RR(i,j,1),RR(i,j,2),2);
                aligned_data1(i,j,3) =
                    target(RR(i,j,1),RR(i,j,2),3);
            end
        end
    end
    target=aligned_data1;
    tg=ad1;

    subplot(3,4,6.5);
    imagesc(tg');
    set(gca,'xtick',[],'ytick',[]);
    title('Rotated Features-Target Image');

    subplot(3,4,7.4);
    imagesc(target(:,:,2)');
    set(gca,'xtick',[],'ytick',[]);
    title('Rotated Target Image');

% ****************************************************************
   % Find centroid, or center of mass of rotated target image
   target_center=[sum(xidx(:).*tg(:))/sum(tg(:)) ...
                  sum(yidx(:).*tg(:))/sum(tg(:))]


    Tr=[-ref_center(1)+target_center(1)
        -ref_center(2)+target_center(2)      ]
```

```
        % TRANSLATE
        % Align the centroid of target image to that of reference image
        % by translation. Add displacement values to original points.
        aligned_data1=zeros(dim(1),dim(2),3);
        ad1=zeros(dim(1),dim(2));

        for i=1:dim(1)
            for j=1:dim(2)
                xdis=fix(i+Tr(1));
                ydis=fix(j+Tr(2));
                if(xdis>0 & xdis<dim(1)+1 & ydis>0 & ydis<dim(1)+1)
                    ad1(i,j)=tg(xdis,ydis);
                    aligned_data1(i,j,1)=target(xdis,ydis,1);
                    aligned_data1(i,j,2)=target(xdis,ydis,2);
                    aligned_data1(i,j,3)=target(xdis,ydis,3);
                end
            end
        end
        target=aligned_data1;
        tg=ad1;

    subplot(3,4,8.3);
    imagesc(tg');
    set(gca,'xtick',[],'ytick',[]);
    title('Translated Features-Target Image');

    subplot(3,4,9.6);
    imagesc(target(:,:,2)');
    set(gca,'xtick',[],'ytick',[]);
    title('Transformed Target Image');

    e=cputime-t     % Total Execution time for algorithm

%****************************************ERRORS**********************

% TRANSLATION AND ROTATION ERROR CALCULATION
% Find centroid, or center of mass target image
    target_center=[sum(xidx(:).*tg(:))/sum(tg(:)) ...
                       sum(yidx(:).*tg(:))/sum(tg(:))]

    % Translation Error
    Tr=[-ref_center(1) + target_center(1)
        -ref_center(2) + target_center(2)      ]

    % Form the covariance matrix target_C
    c11=0;
    c12=0;
    c21=0;
    c22=0;
    k=1;
    t_xy=ones(dim(1)*dim(2),2);
```

```
    for i = 1:dim(1)
        for j = 1:dim(2)
            c11 = c11 + (i - target_center(1)) ^ 2 * tg(i,j);
            c22 = c22 + (j - target_center(2)) ^ 2 * tg(i,j);
            c12 = c12 + (i - target_center(1)) * (j - target_center(2))
                                * tg(i,j);
            t_xy(k,1)=j;
            t_xy(k,2)=i;
            k=k+1;
        end
    end
    c21 = c12;
    target_C = [c11 c12; c21 c22];

    % Find the largest eigen values and the eigen vectors for
    % covariance matrix A
    [target_V,target_L]= eigs(target_C);

    % find target_theta, the angle between the eigenvector and the x-
    % axis for
    % target image. %first column of target_V=[cos(theta);sin(theta)]
    target_theta = atan2(target_V(2,1),target_V(1,1))*180/pi;

    % Rotation Error
    angl=target_theta-ref_theta

%**************************************************************************

    % Find the similarity between the two aligned images by subtracting
    % reference image from aligned image and display the difference
    tg2=target(:,:,2);

    image_diff(:,:,1) = (rf2-min(rf2(:)))/ ...
                         (max(rf2(:))-min(rf2(:)));
    image_diff(:,:,3) = (tg2-min(tg2(:)))/ ...
                         (max(tg2(:))-min(tg2(:)));
    image_diff(:,:,2) = (image_diff(:,:,1)+image_diff(:,:,3))/2.0;

    image_diff=permute(image_diff,[2 1 3]);

    subplot(3,4,10.5);
    imagesc(image_diff);
    set(gca,'xtick',[],'ytick',[]);
    title('Reference-Transformed Image');

% *************************************************************************
```

```
    % Find similarity measure between images by Mean Square Error (MSE)
    % method

    ad=0;
    for i = 1:dim(1)
        for j = 1:dim(2)
            ad= ad+ abs(rf2(i,j)-tg2(i,j))^2;
        end
    end

    MSE_err = ad / (dim(1)*dim(2))

% ************************************************************************

    % Display error parameters and execution time
    subplot(3,4,11.4);
    set(gca,'xtick',[],'ytick',[]);
    text(.1,.65,['Translation Error =
       (',num2str(Tr(1)),',',num2str(Tr(2)),') pixels'],'FontSize',10);
    text(.1,.5,['Rotation Angle Error = ',num2str(angl),'
       degrees'],'FontSize',10);
    text(.1,.35,['MSE Error = ',num2str(MSE_err)],'FontSize',10);
    text(.1,.2,['Execution Time = ',num2str(e),' secs'],'FontSize',10);

end
% ************************************************************************
```

APPENDIX C

FAST FOURIER TRANSFORM SCRIPT IN MATLAB

```
% Script to realign images using Fast Fourier Transform based
% correlation Algorithm
% MATLAB function comments are taken from MATLAB documentation

function target = Fourier_Transform_Algorithm(ref,target,dim)
    t=cputime
    % CALCULATE ANGLE OF ROTATION

    % Find Fast Fourier Transform function for each image
    % Y = fft2(X) returns the two-dimensional discrete Fourier
    % transform (DFT)
    % of X, computed with a fast Fourier transform (FFT) algorithm.
    % The result Y is the same size as X.
    % Y = fftshift(X) rearranges the outputs of fft, fft2, and fftn by
    % moving the zero-frequency component to the center of the array.
    % It is useful for visualizing a Fourier transform with the zero-
    % frequency component in the middle of the spectrum.

    F_ref = fftshift(fft2(ref(:,:,2)));
    F_target = fftshift(fft2(target(:,:,2)));

    subplot(3,4,4.3);
    imagesc(log(abs(F_ref)));
    set(gca,'xtick',[],'ytick',[]);
    title('FFT(Reference)');

    subplot(3,4,5.6);
    imagesc(log(abs(F_target)));
    set(gca,'xtick',[],'ytick',[]);
    title('FFT(Target)');

    % convert polar coordinates to cartesian coordinates
    fsize=size(ref);
    width=fsize(2);
    height=fsize(1);
    largest_dim=max(fsize);
    max_radius=sqrt(sum((fsize(1:2)/2).^2));
    scale_factor=1.0;
    npts_per_dim=scale_factor*largest_dim;
    r_range=logspace(0,log10(max_radius),npts_per_dim);
    th_range=linspace(-pi,pi,npts_per_dim);
    fill_with=0;
    interp_method='linear';
    [TH,R]=meshgrid(th_range,r_range);
    [x,y]=pol2cart(TH,R);
```

```
    % Use bilinear interpolation to interpolate image intensities from
    % cartesian coordinate system to those in logpolar coordinate
    % system
    % ZI = interp2(X,Y,Z,XI,YI) returns matrix ZI containing elements
    % corresponding to the elements of XI and YI and determined by
    % interpolation
    % within the two-dimensional function specified by matrices X, Y,
    % and Z.
    % X and Y must be monotonic, and have the same format ("plaid") as
    % if they
    % were produced by meshgrid. Matrices X and Y specify the points at
    % which
    % the data Z is given. Out of range values are returned as NaNs.
    % XI and YI can be matrices, in which case interp2 returns the
    % values of Z
    % corresponding to the points (XI(i,j),YI(i,j)).
    % Alternatively, you can pass in the row and column vectors xi and
    % yi,
    % respectively. In this case, interp2 interprets these vectors as
    % if you
    % issued the command meshgrid(xi,yi).
    % ZI = interp2(Z,XI,YI) assumes that X = 1:n and Y = 1:m, where
    %[m,n] = size(Z).


ref_interp=interp2(double(ref(:,:,2)),x+width/2,y+height/2,interp_metho
 d,fill_with);

target_interp=interp2(double(target(:,:,2)),x+width/2,y+height/2,interp
_method,fill_with);

    % Find FFT values for images in logpolar coordinate system
    F_ref_polar = fftshift(fft2(ref_interp));
    F_target_polar = fftshift(fft2(target_interp));

    subplot(3,4,6.5);
    imagesc(log(abs(F_ref_polar)));
    set(gca,'xtick',[],'ytick',[]);
    title('FFT(Reference) in Log-Polar');

    subplot(3,4,7.4);
    imagesc(log(abs(F_target_polar)));
    set(gca,'xtick',[],'ytick',[]);
    title('FFT(Target) in Log-Polar');

 % Find ratio to define cross-power spectrum of two images (Phase
 % Correlation).
 % Phase of the cross-power spectrum is equal to the phase
 % difference between the images.
 rati = (F_ref_polar .* conj(F_target_polar)) ./ (abs(F_ref_polar .*
                       conj(F_target_polar)));
```

```
    % Call inverse FFT on the representation in the frequency domain.
    % Y = ifft2(X) returns the two-dimensional inverse discrete Fourier
    % transform (DFT) of X, computed with a fast Fourier transform
    % (FFT)
    % algorithm. The result Y is the same size as X.
    % ifft2 tests X to see whether it is conjugate symmetric.
    % If so, the computation is faster and the output is real.
    % An M-by-N matrix X is conjugate symmetric if
    % X(i,j)=conj(X(mod(M-i+1, M) + 1, mod(N-j+1, N) + 1)) for each
    % element of X.
    % This returns a function that is an impulse with approximate zeros
    % everywhere except at the displacement.
    inv_ratio = fftshift(ifft2(rati));

    % Call max function to get max value and its position in the array.
    % The position is used to find the angle of rotation between the
    % two images.
    [max_val,loc] = max(abs(inv_ratio(:)));
    x_coord = fix(mod(loc,dim(1)))
    y_coord = fix(loc / dim(2))
    if x_coord==0 x_coord=1; end
    if y_coord==0 y_coord=1; end

    y_th=TH(x_coord,y_coord) % angular displacement in radians

    % find rotation angle
    angl_r = y_th;            % angle in radians

%**************************************************************************
    % ROTATE
    t_xy=ones(dim(1)*dim(2),2);
    k=1;
    for i = 1:dim(1)
       for j = 1:dim(2)
            t_xy(k)=j;
            t_xy(k,2)=i;
            k=k+1;
       end
    end

    RR = ones(dim(1)*dim(2),2);
    temp = ones(2,1);

    % Reverse Rotate the target image about its center by angl_r
    R = [cos(-angl_r) sin(-angl_r);-sin(-angl_r) cos(-angl_r)];

    % Apply rotation parameters on original coordinates in t_xy in
    % center
    for i = 1 :dim(1)*dim(2)
        temp = ((R * (t_xy(i,:)-dim(1)/2)')+dim(1)/2);
        RR(i,:) = temp';
    end
```

```
    RR = reshape(RR,[dim(1) dim(2) 2]);
    RR = round(RR); % RR has the rotated target image coordinates

    aligned_data1 = zeros(dim(1),dim(2),3);
    for i=1:dim(1)
        for j=1:dim(2)
            if(RR(i,j,1)>0 & RR(i,j,1)<dim(1)+1 & RR(i,j,2)>0 &
                    RR(i,j,2)<dim(2)+1)
               aligned_data1(i,j,1) = target(RR(i,j,1),RR(i,j,2),1);
               aligned_data1(i,j,2) = target(RR(i,j,1),RR(i,j,2),2);
               aligned_data1(i,j,3) = target(RR(i,j,1),RR(i,j,2),3);
            end
        end
    end

    target=aligned_data1;

    subplot(3,4,8.3);
    imagesc(target(:,:,2)');
    set(gca,'xtick',[],'ytick',[]);
    title('Rotated Target Image');

%**************************************************************************
    % CALCULATE DISPLACEMENT

    % Find Fast Fourier Transform function for each image
     F_target = fftshift(fft2(target(:,:,2)));

    % Find ratio to define cross-power spectrum of two images (Phase
    % Correlation).
    % Phase of the cross-power spectrum is equal to the phase
    % difference between the images.
    rati = (F_ref .* conj(F_target)) ./ (abs(F_ref .* conj(F_target)));

    % Call inverse FFT on the representation in the frequency domain.
    % This returns a function that is an impulse with approximate zeros
    % everywhere except at the displacement.
    inv_ratio = ifft2(rati);

    % Call max function to get max value and its position in the array.
    % This value is the translation between the two images.
    % Use MOD and DIVIDE to get the x and y coordinates of the
    % displacement position.
    [max_val,loc] = max(abs(inv_ratio(:)));
    x_coord = mod(loc,dim(1));
    y_coord = loc / dim(2);
    Tx = x_coord;
    Ty = y_coord;

    if (x_coord < (dim(1) / 2))
     Tx = -(Tx-1);
    else
     Tx = dim(1) -(Tx-1);
```

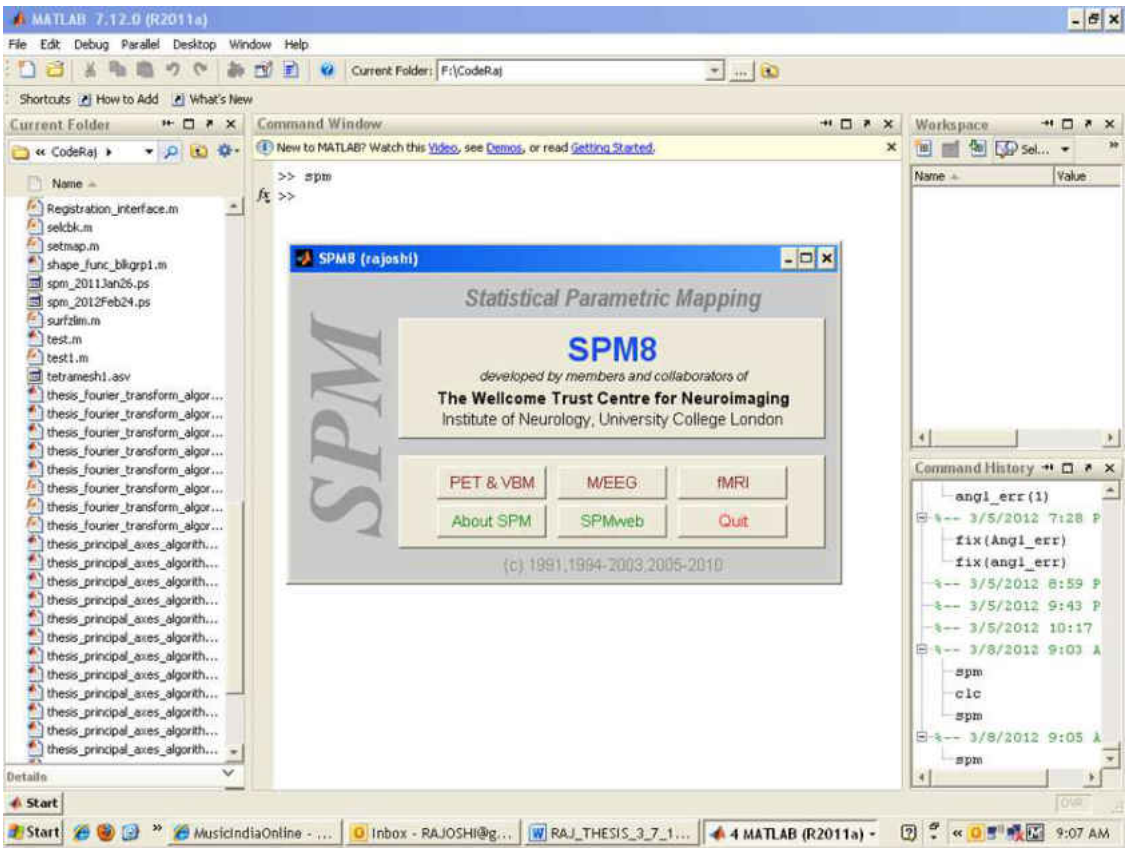
```
    end
    if (y_coord < (dim(2) / 2))
      Ty = -(Ty-1);
    else
      Ty = dim(2) - (Ty-1);
    end

%**********************************************************************
    %TRANSLATE
    % Align the two images by adding displacement values
    aligned_data2=zeros(dim(1),dim(2),3);
    for i=1:dim(1)
      for j=1:dim(2)
         xdis=fix(i+Tx);
         ydis=fix(j+Ty);
         if(xdis>0 & xdis<dim(1)+1 & ydis>0 & ydis<dim(1)+1)
            aligned_data2(i,j,1)=target(xdis,ydis,1);
            aligned_data2(i,j,2)=target(xdis,ydis,2);
            aligned_data2(i,j,3)=target(xdis,ydis,3);
         end
      end
    end
    target=aligned_data2;

    subplot(3,4,9.6);
    imagesc(target(:,:,2)');
    set(gca,'xtick',[],'ytick',[]);
    title('Translated Target Image');

    e=cputime-t

% *********************SIMILARITY MEASURES*****************************

    % Find the similarity between the two aligned images by subtracting
    % reference image from aligned image and display the difference
    rf=ref(:,:,2);
    tg=target(:,:,2);
    image_diff(:,:,1) = (rf-min(rf(:)))/ ...
                         (max(rf(:))-min(rf(:)));
    image_diff(:,:,3) = (tg-min(tg(:)))/ ...
                        (max(tg(:))-min(tg(:)));
    image_diff(:,:,2) = (image_diff(:,:,1)+image_diff(:,:,3))/2.0;

    image_diff=permute(image_diff,[2 1 3]);

    subplot(3,4,10.5);
    imagesc(image_diff);
    set(gca,'xtick',[],'ytick',[]);
    title('Reference-Transformed Image');

%**********************************************************************
```

```
% Find similarity measure between images by Mean Square Error (MSE)
% method

    ad=0;
    for i = 1:dim(1)
     for j = 1:dim(2)
        ad= ad+ abs(ref(i,j,2)-target(i,j,2))^2;
     end
    end

    MSE_err = ad / (dim(1)*dim(2))

%*********************************************************************

    % Display error parameter and execution time
    subplot(3,4,11.4);
    set(gca,'xtick',[],'ytick',[]);
    text(.1,.35,['MSE Error = ',num2str(MSE_err)],'FontSize',10);
    text(.1,.2,['Execution Time = ',num2str(e),' secs'],'FontSize',10);

%*********************************************************************
end
```

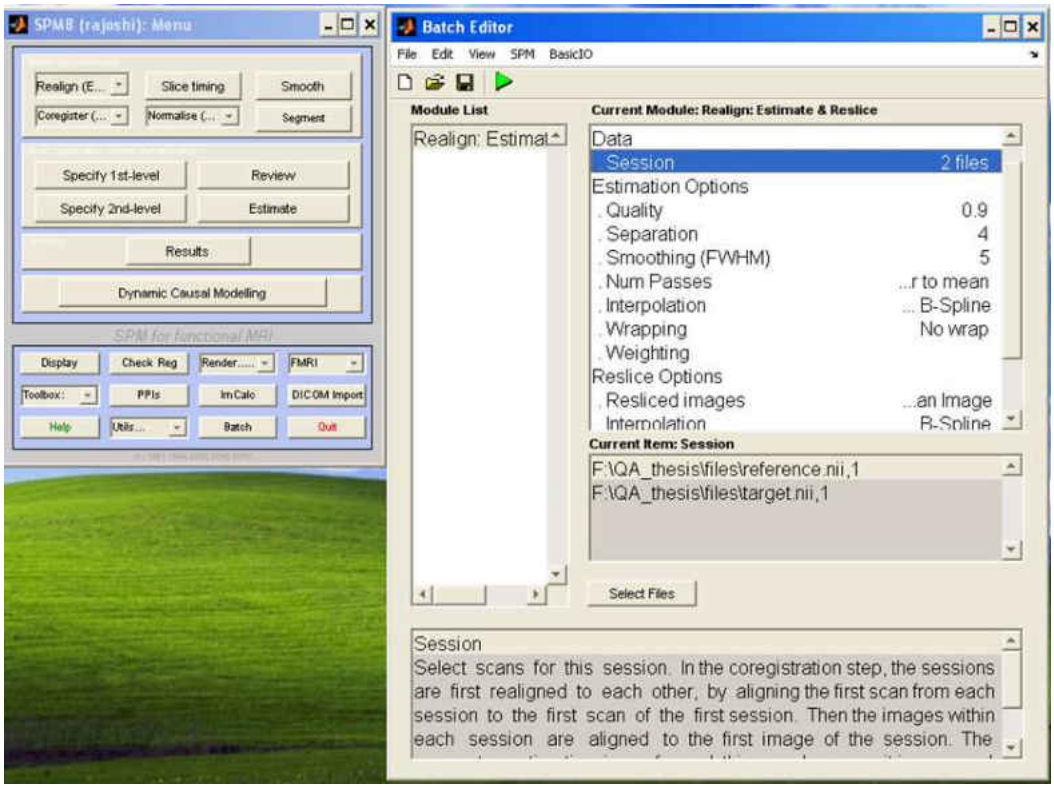
APPENDIX D

REGISTRATION OUTPUT WINDOWS IN SPM8

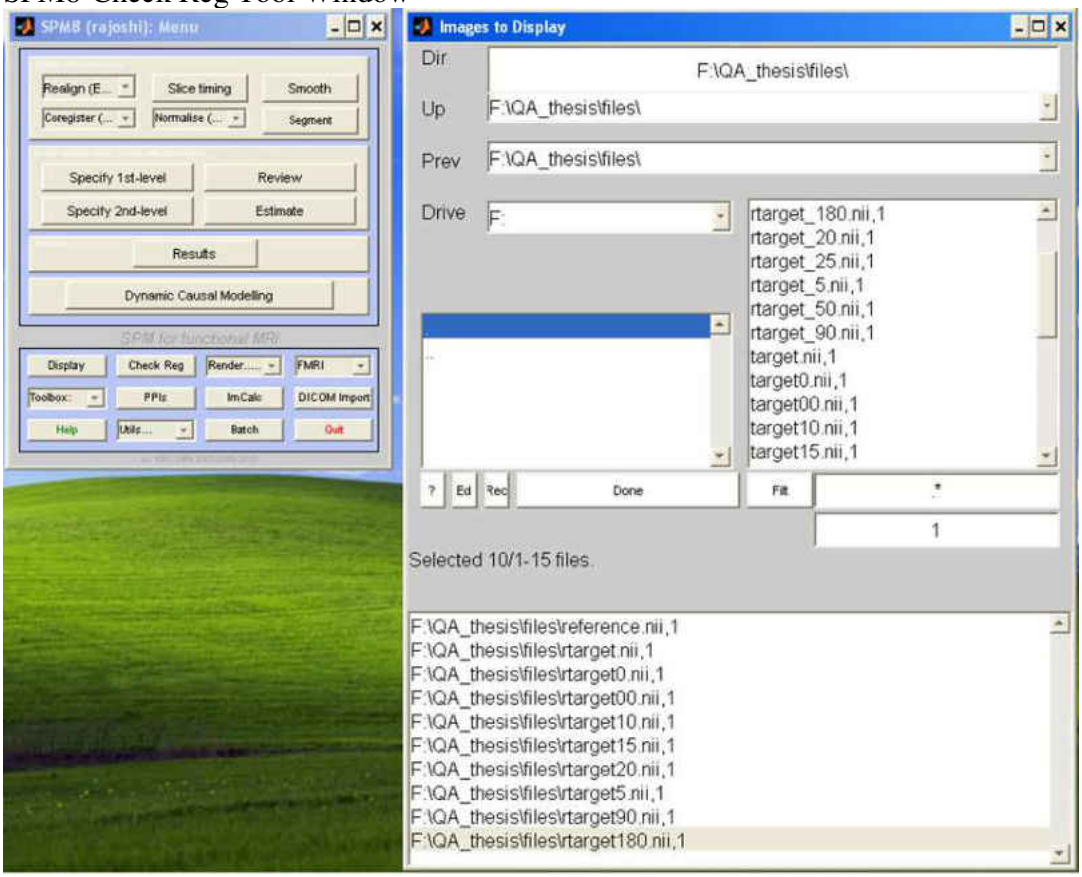
SPM8 Main Window



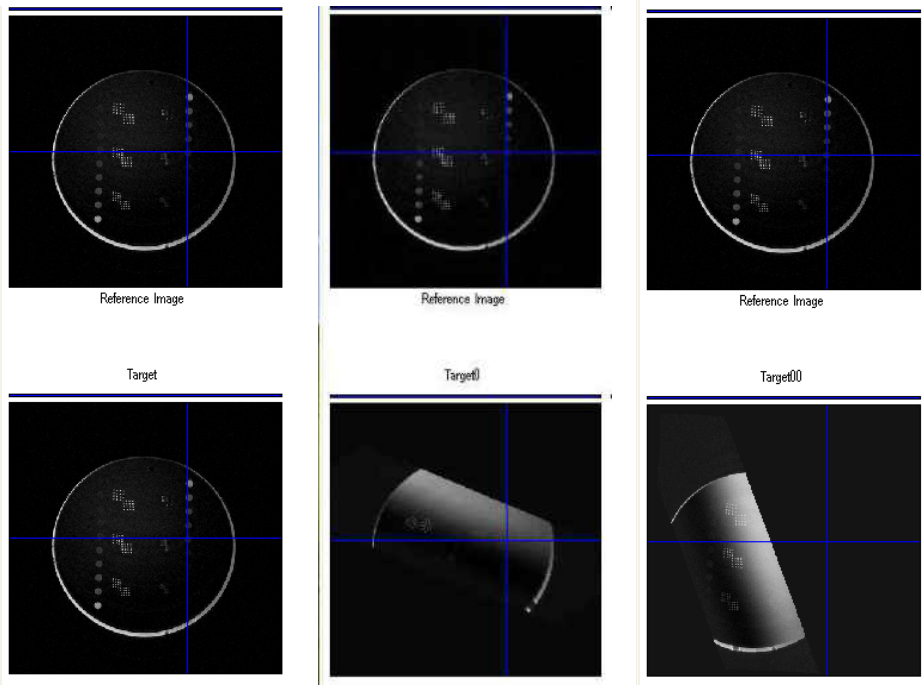
SPM8-Realign Tool Window

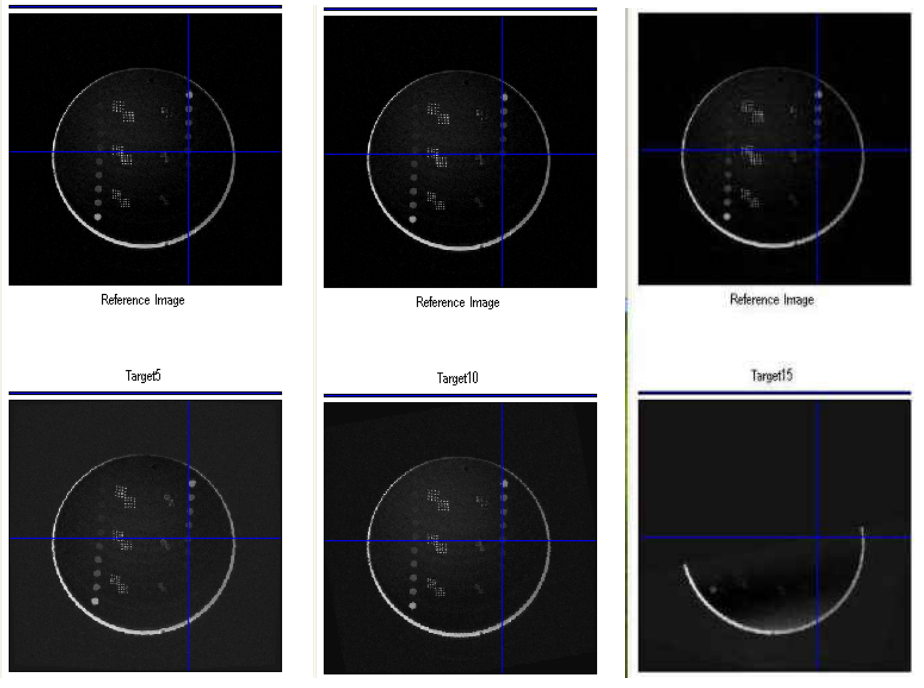


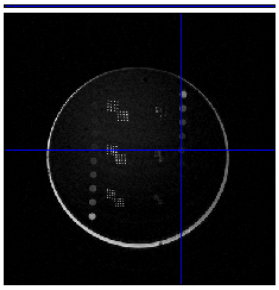
SPM8-Check Reg Tool Window



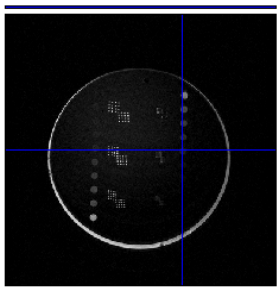
Registration Windows



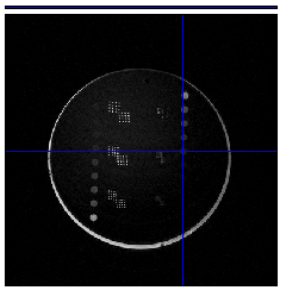




Reference Image

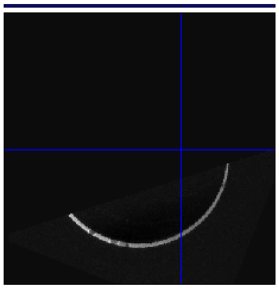


Reference Image

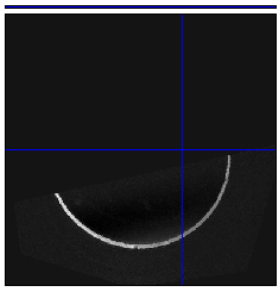


Reference Image

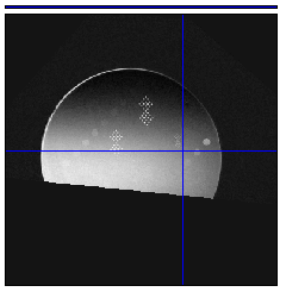
Target20

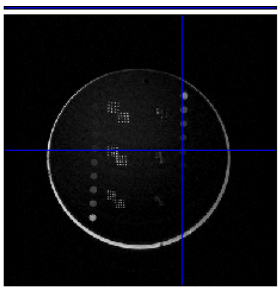


Target25

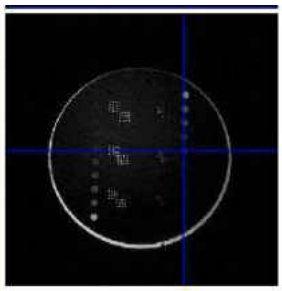


Target90

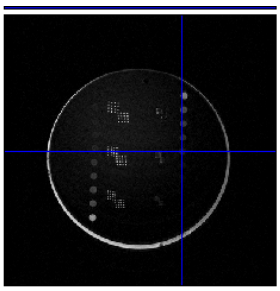




Reference Image

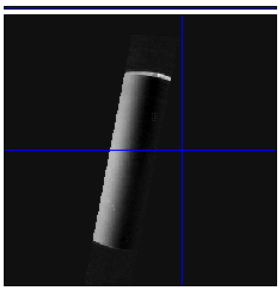


Reference Image

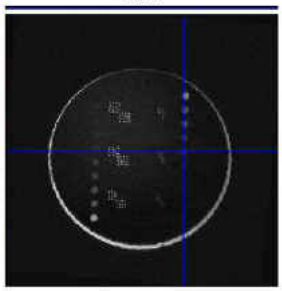


Reference Image

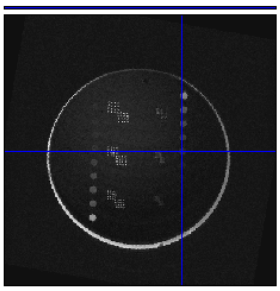
Target180

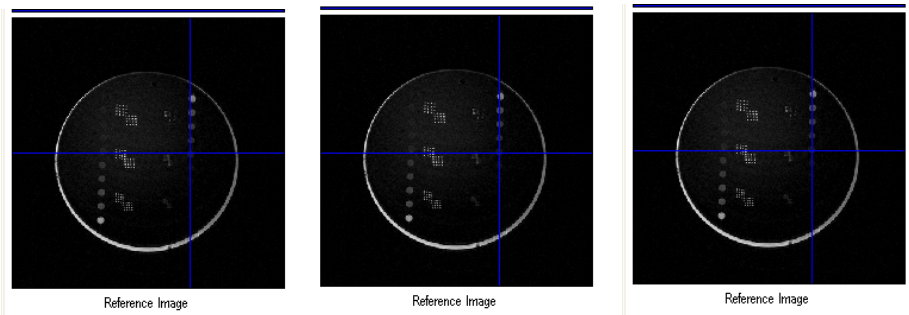


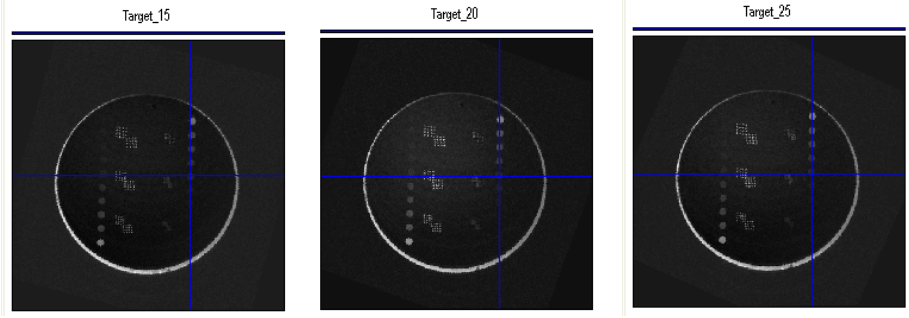
Target_5

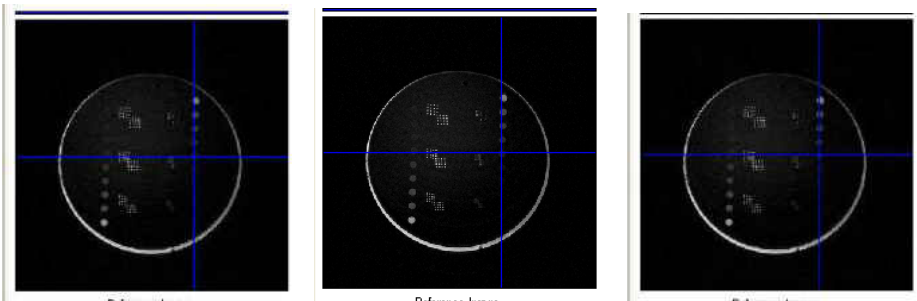


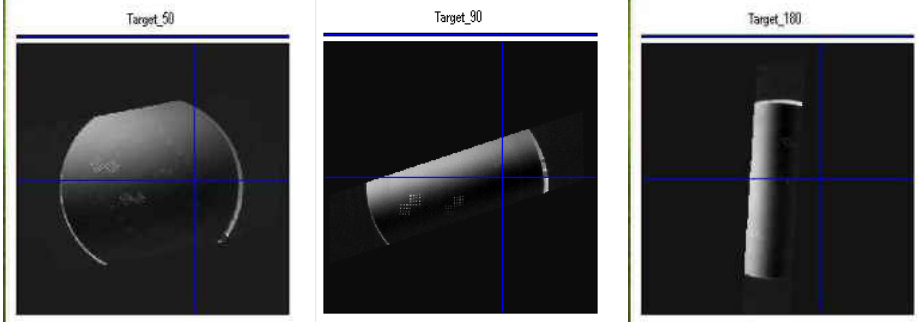
Target_10

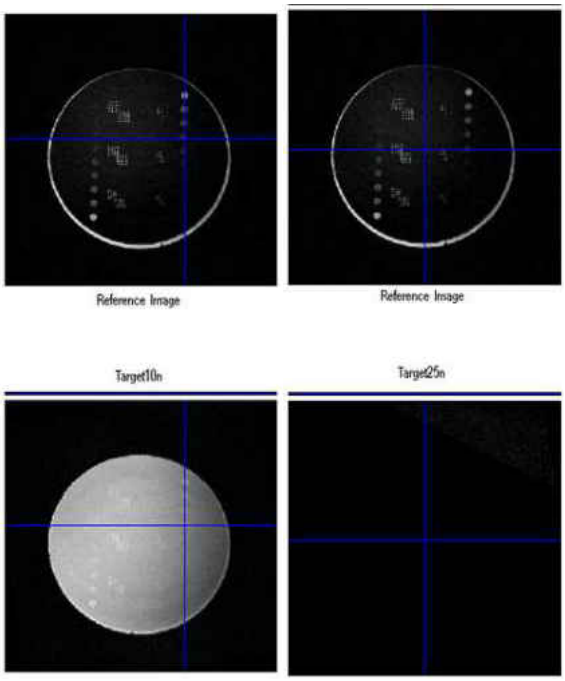



Reference Image
Reference Image
Reference Image

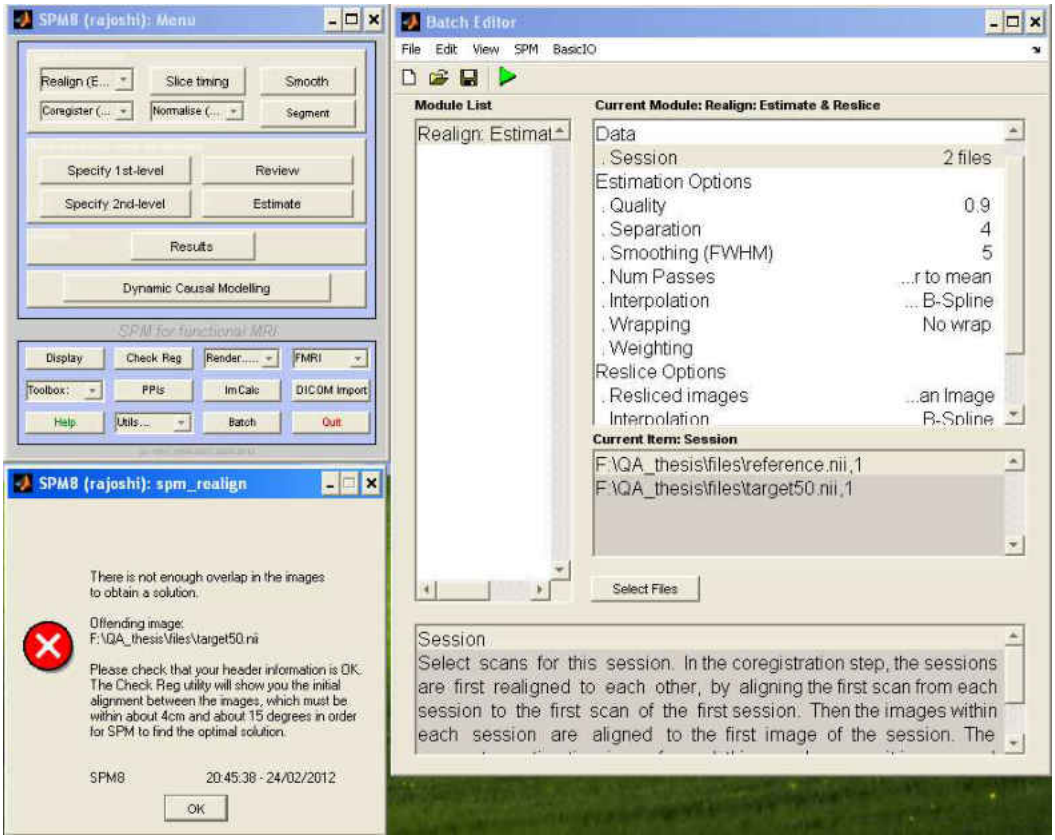

Target_15
Target_20
Target_25


Reference Image
Reference Image
Reference Image


Target_50
Target_90
Target_180



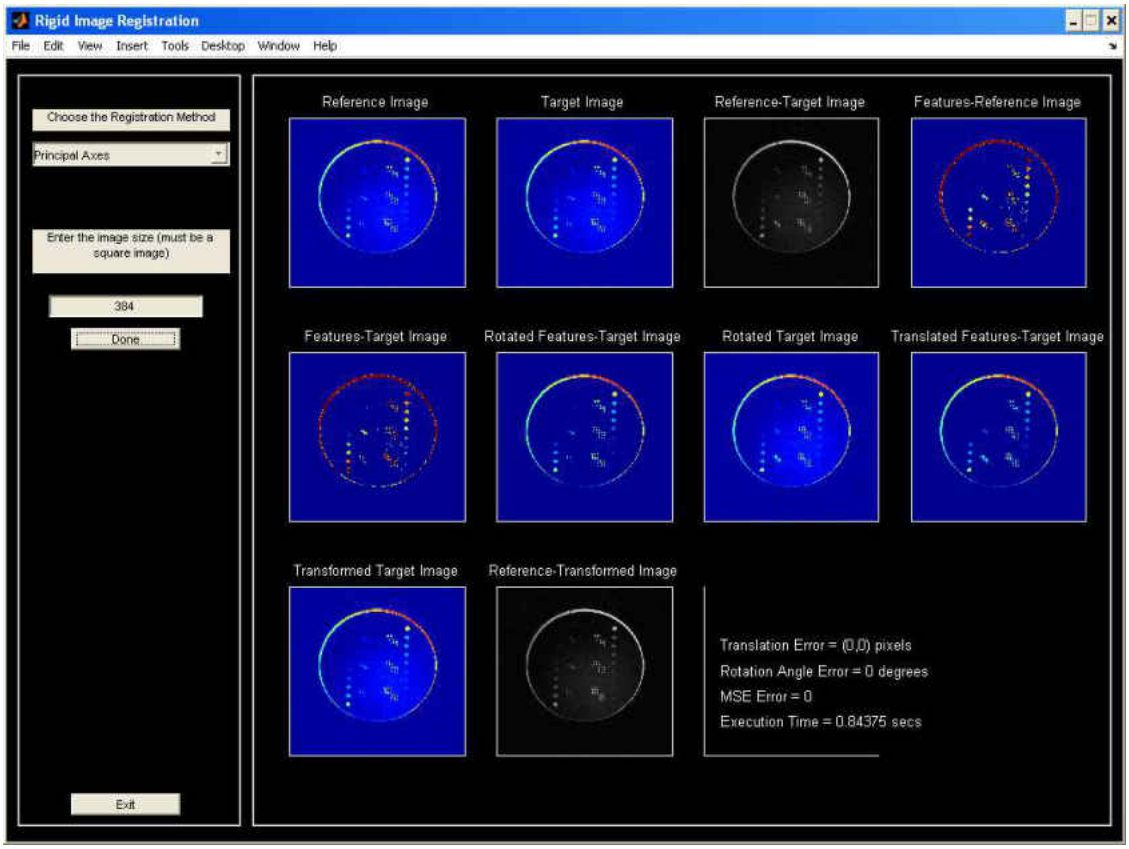
Error aligning target50.raw file with 50 degree rotation angle difference



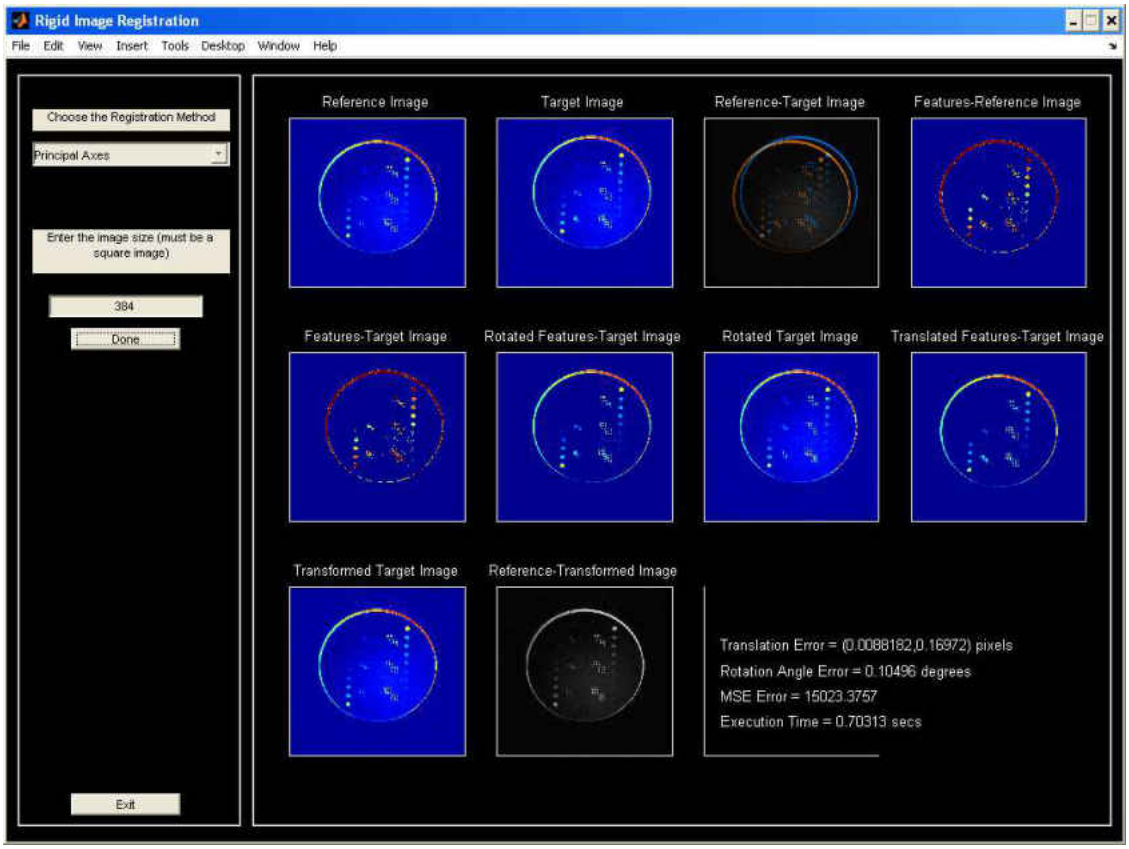
APPENDIX E

REGISTRATION OUTPUT WINDOWS USING PRINCIPAL AXES ALGORITHM

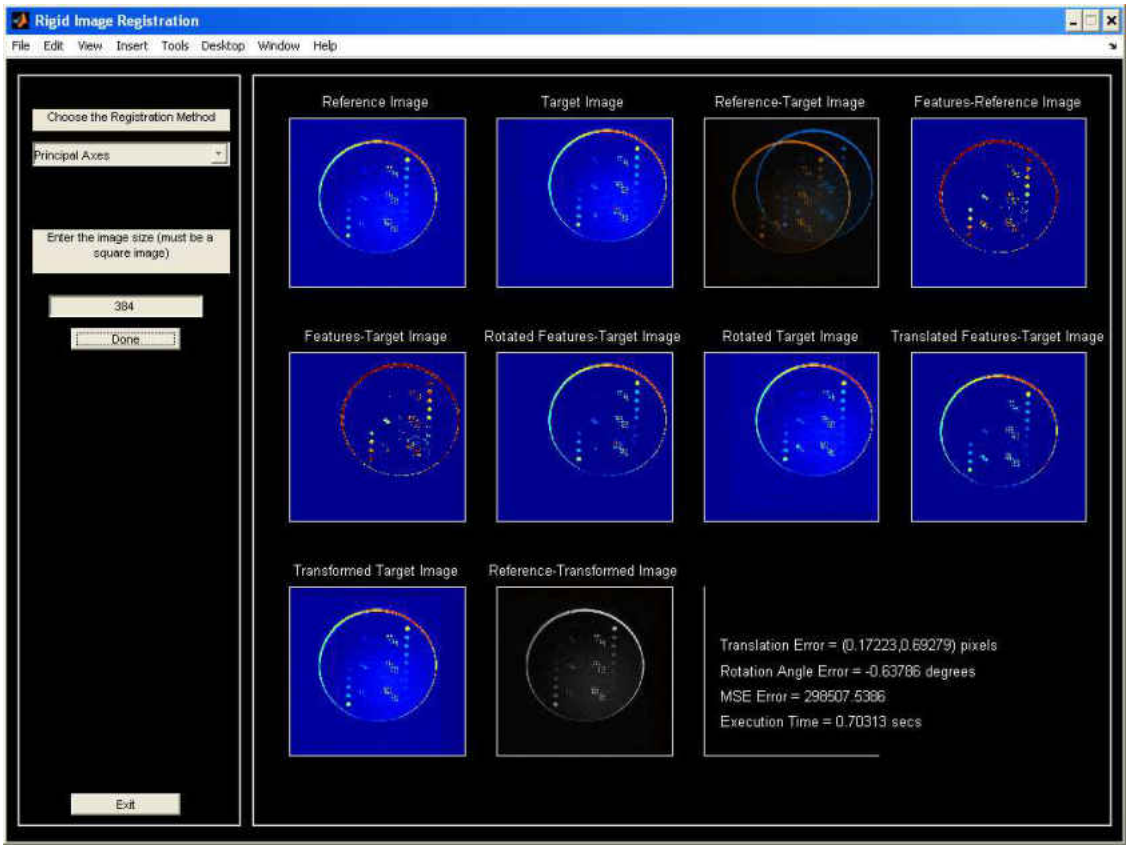
Target.raw : Translation = (0, 0) Rotation=0



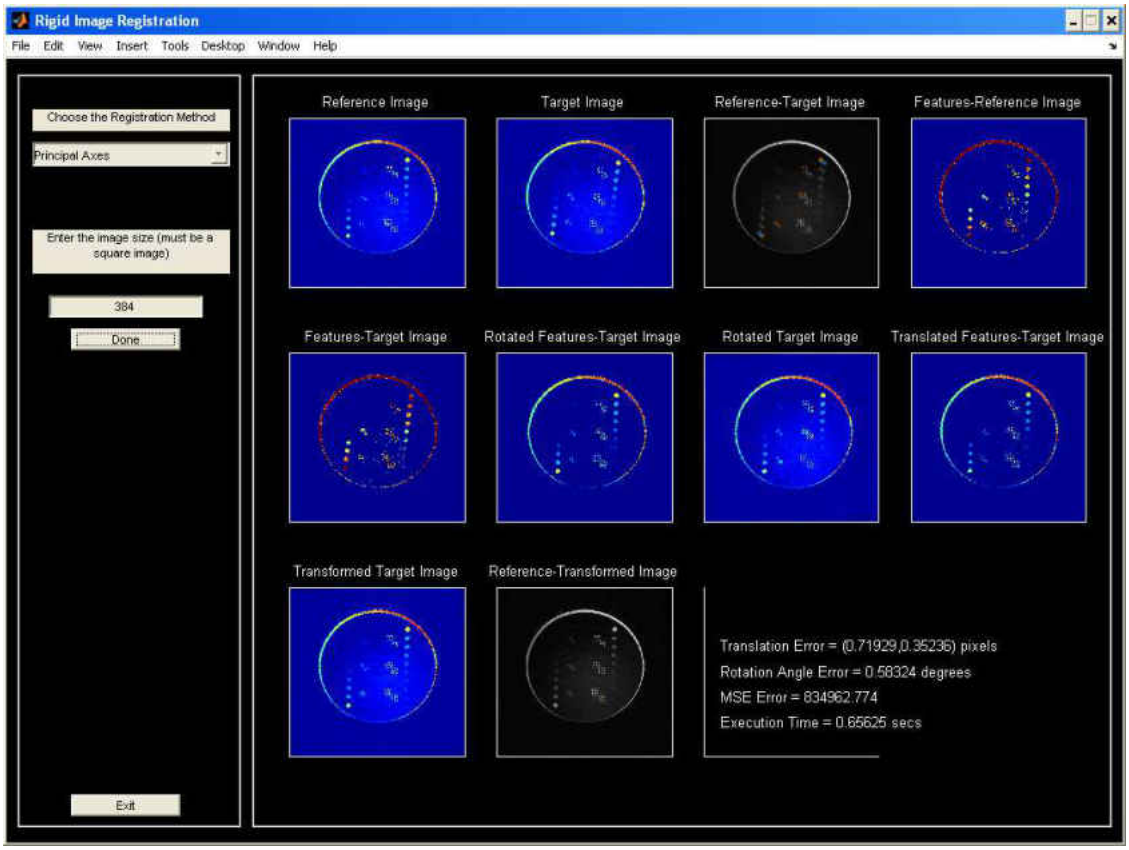
Target0.raw : Translation = (15,-10) Rotation=0



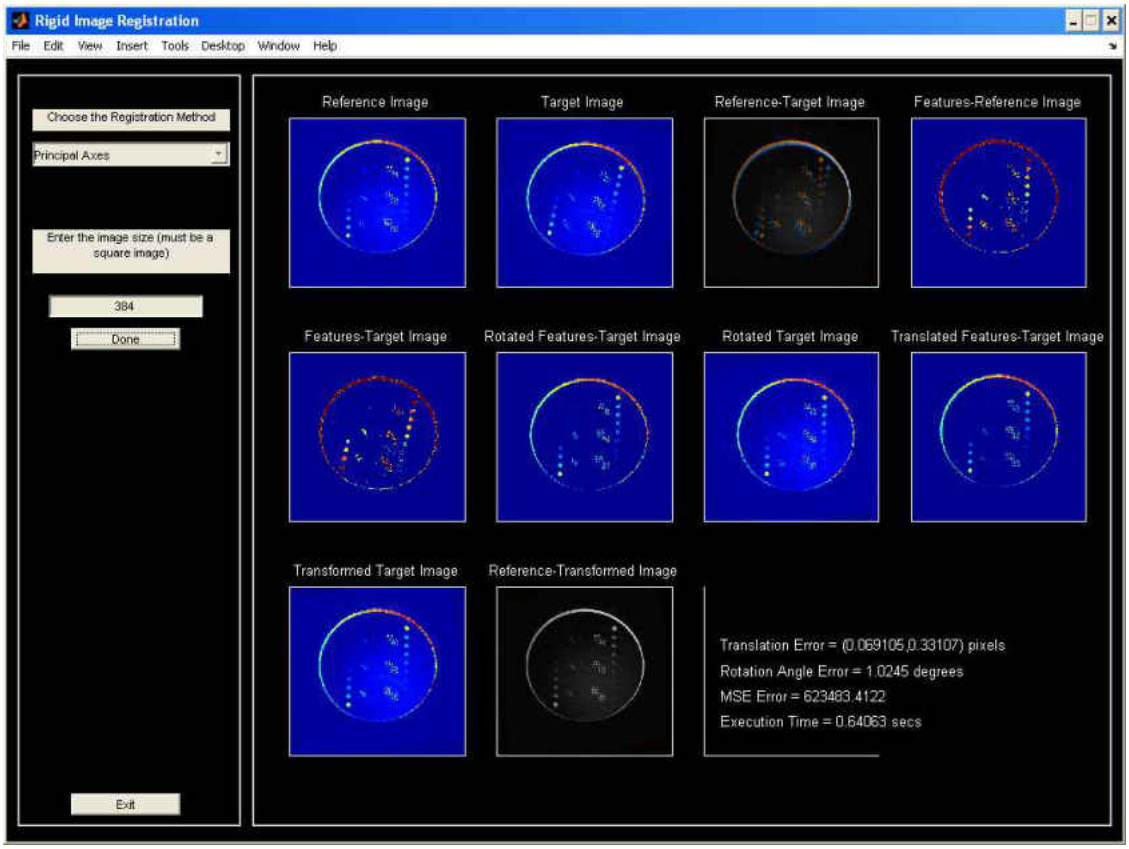
Target00.raw : Translation = (50,-25) Rotation=0



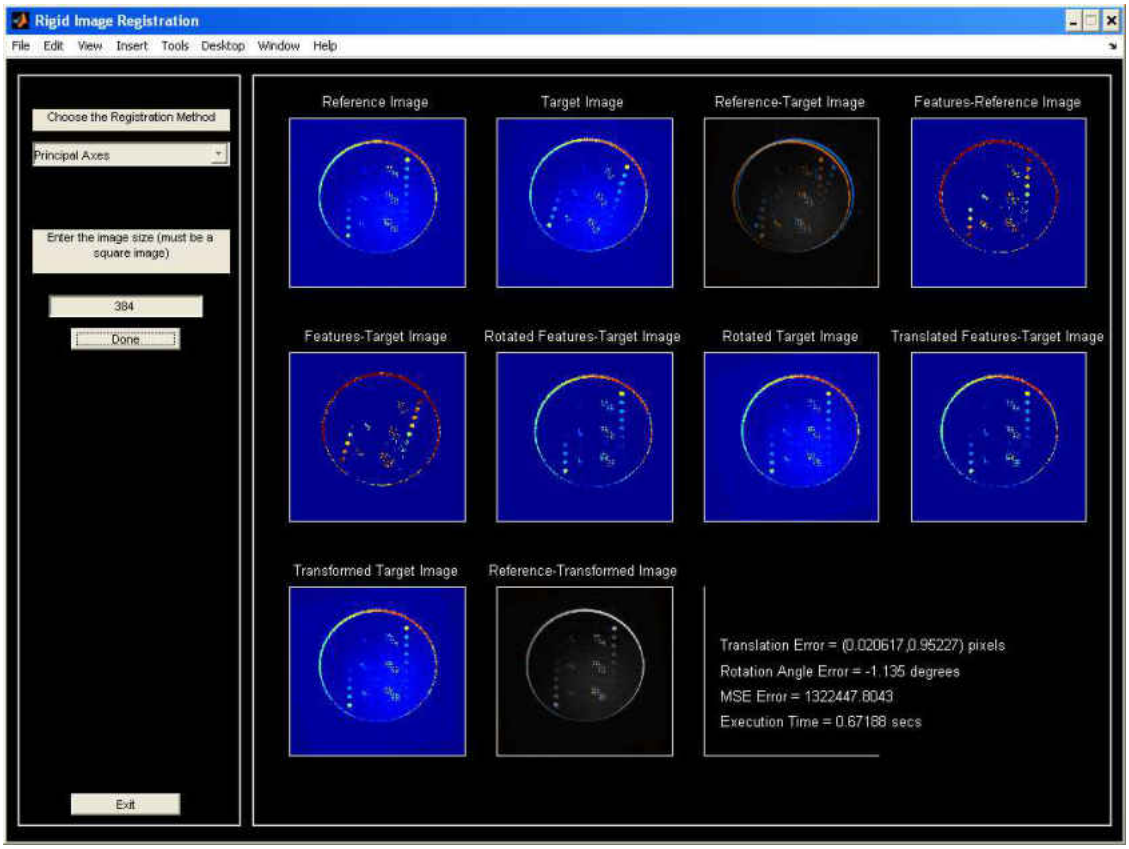
Target5.raw : Translation = (5, 5) Rotation=5



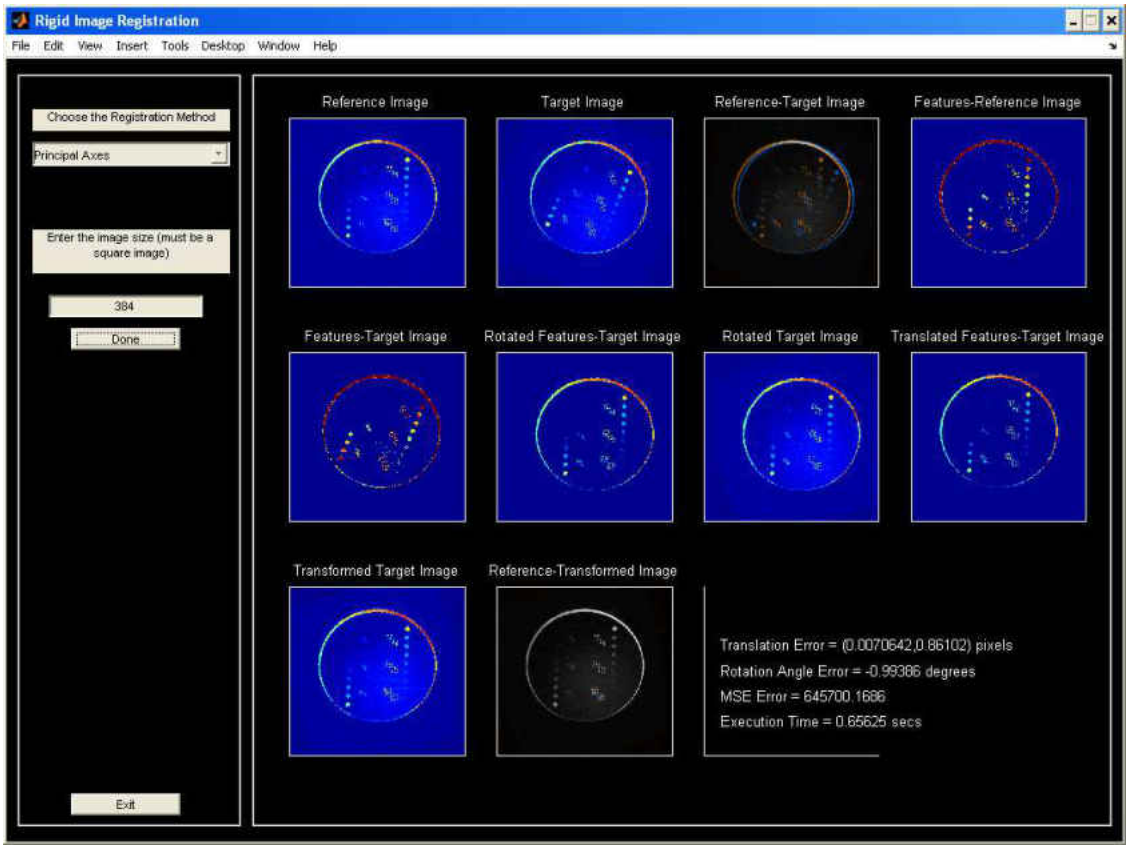
Target10.raw : Translation = (0, 5) Rotation=10



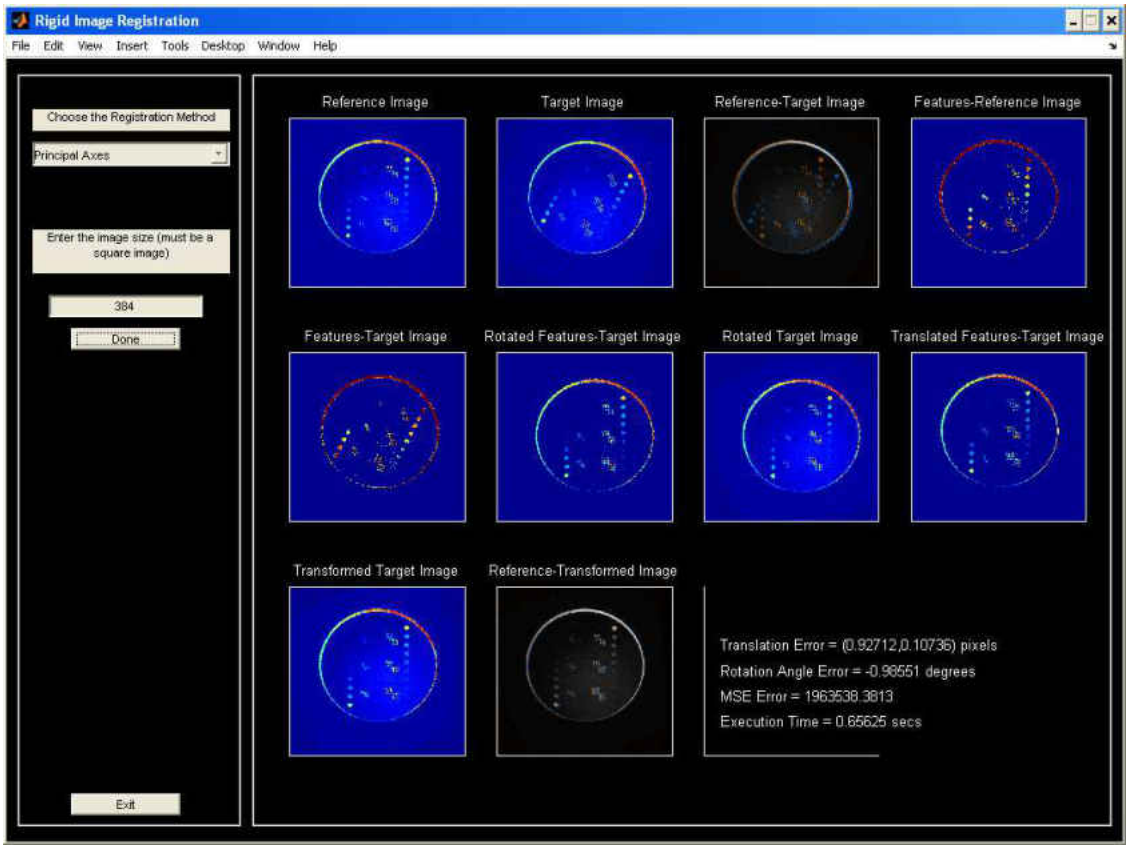
Target15.raw : Translation = (15, 10) Rotation=15



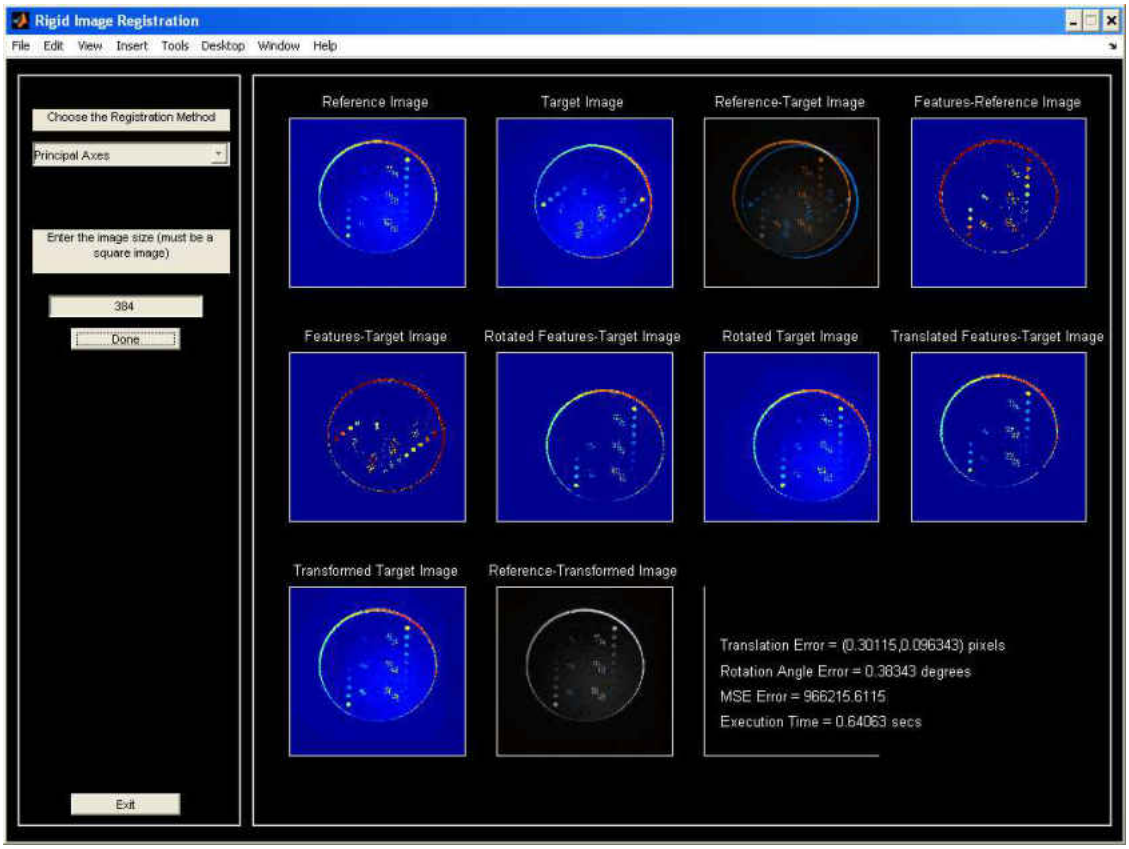
Target20.raw : Translation = (10, 20) Rotation=20



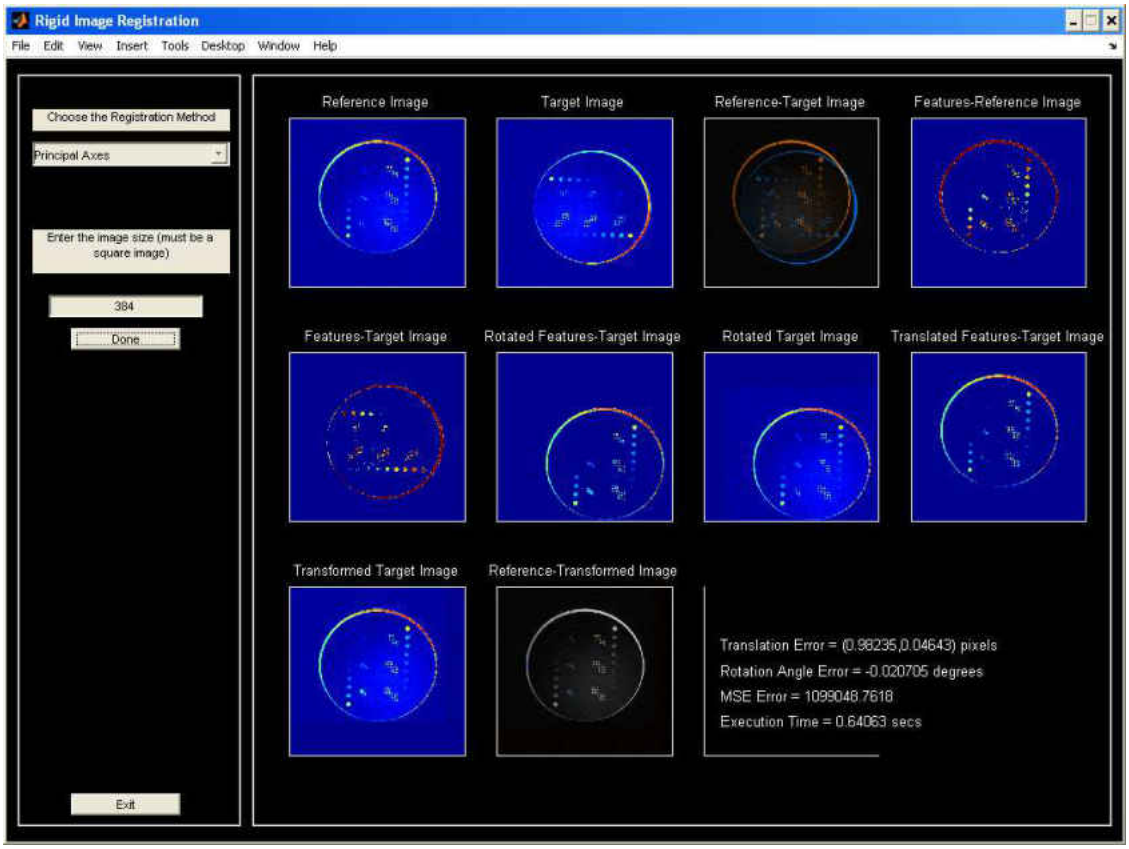
Target25.raw : Translation = (-20, 20) Rotation=25



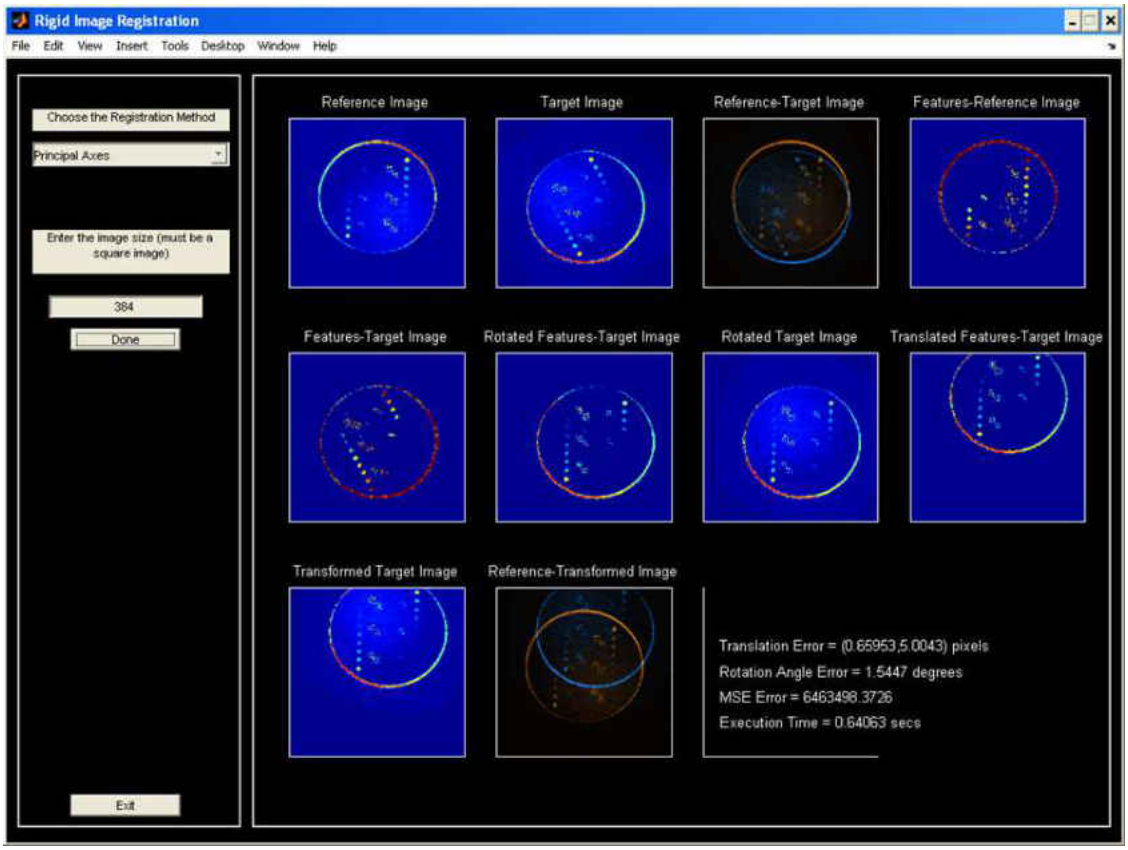
Target50.raw : Translation = (25,-25) Rotation=50



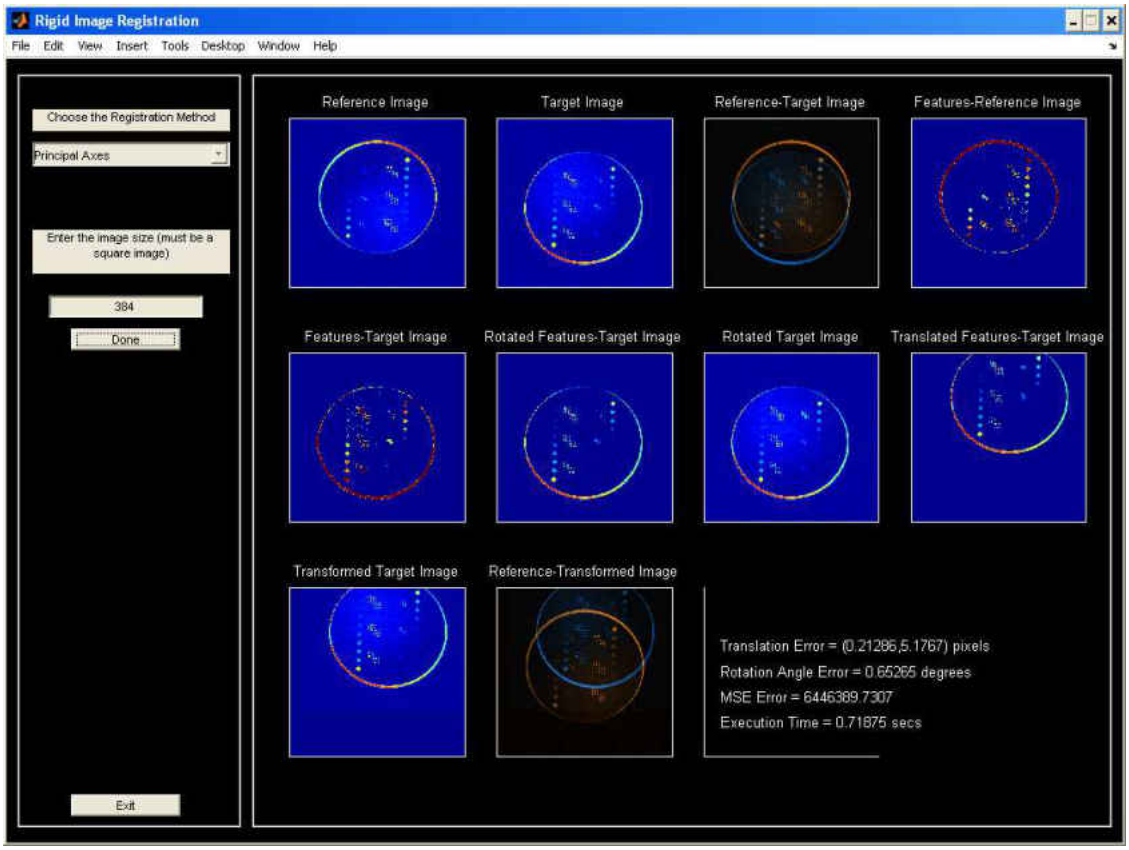
Target90.raw : Translation = (5, 10) Rotation=90



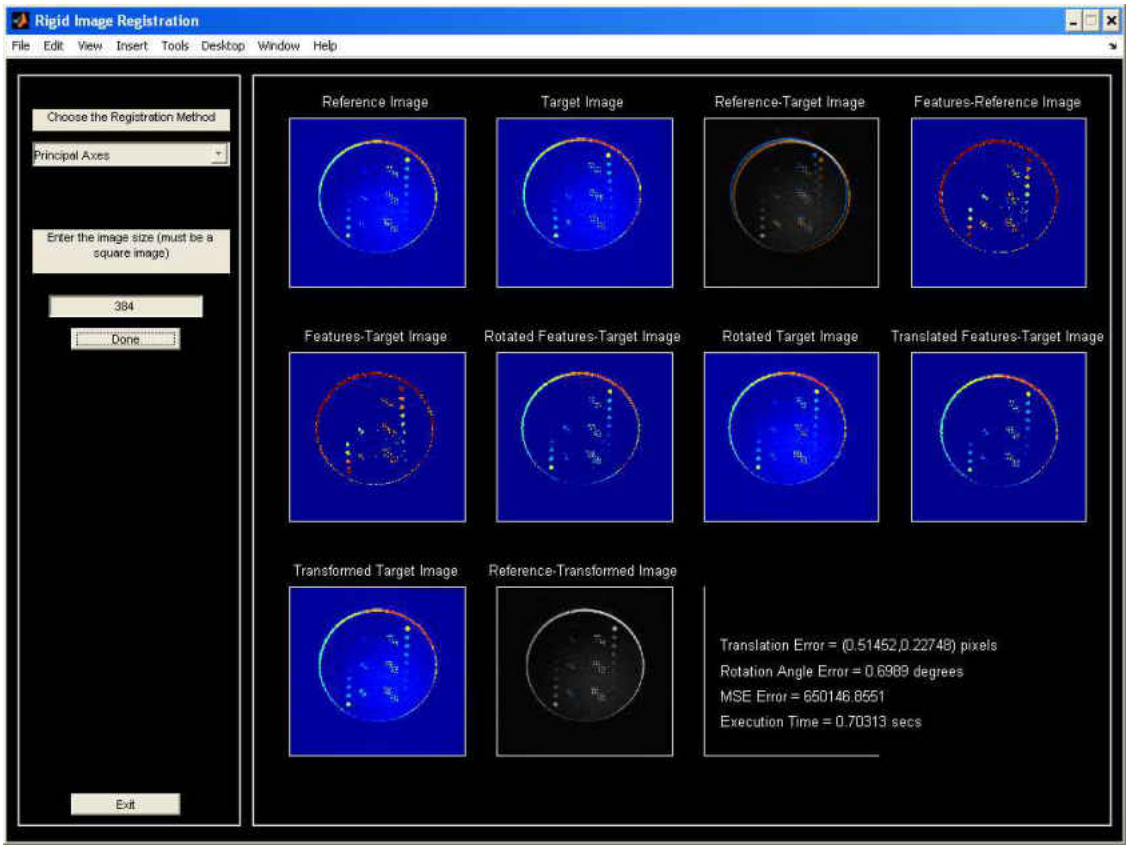
Target150.raw : Translation = (0,0) Rotation=150



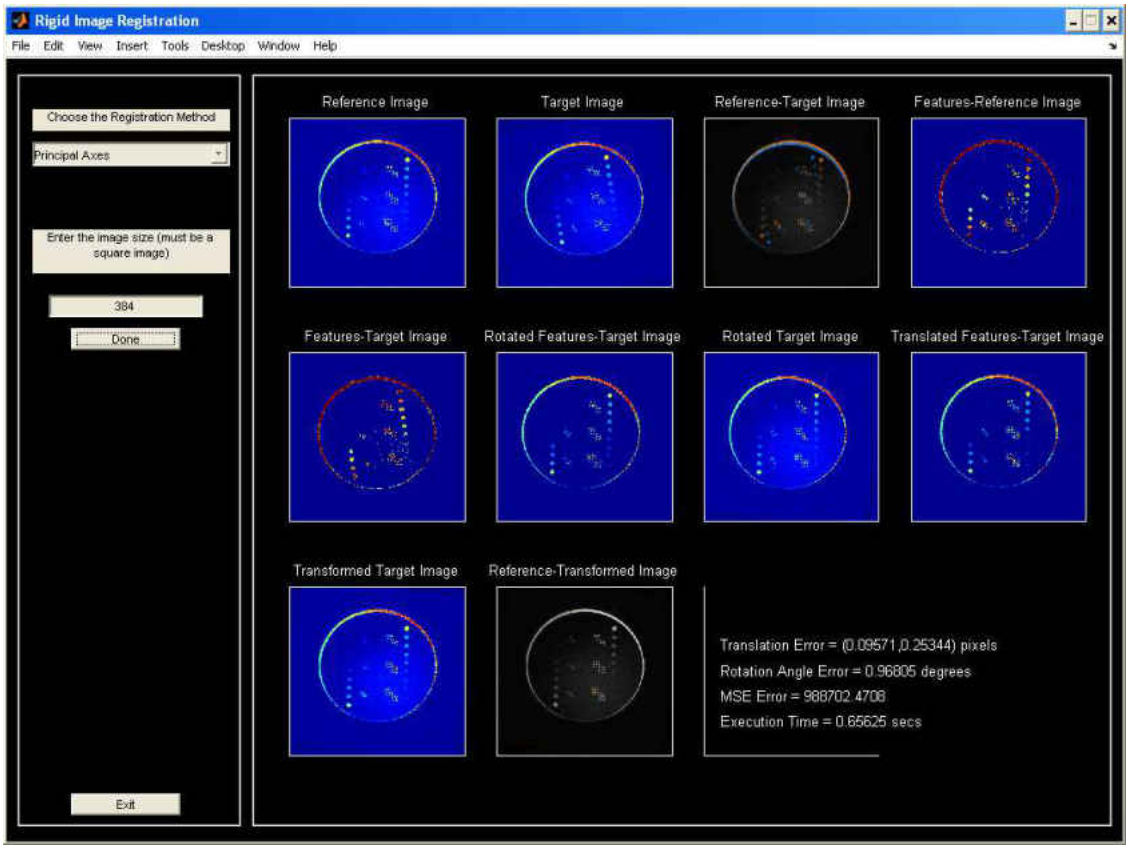
Target180.raw : Translation = (0,0) Rotation=180



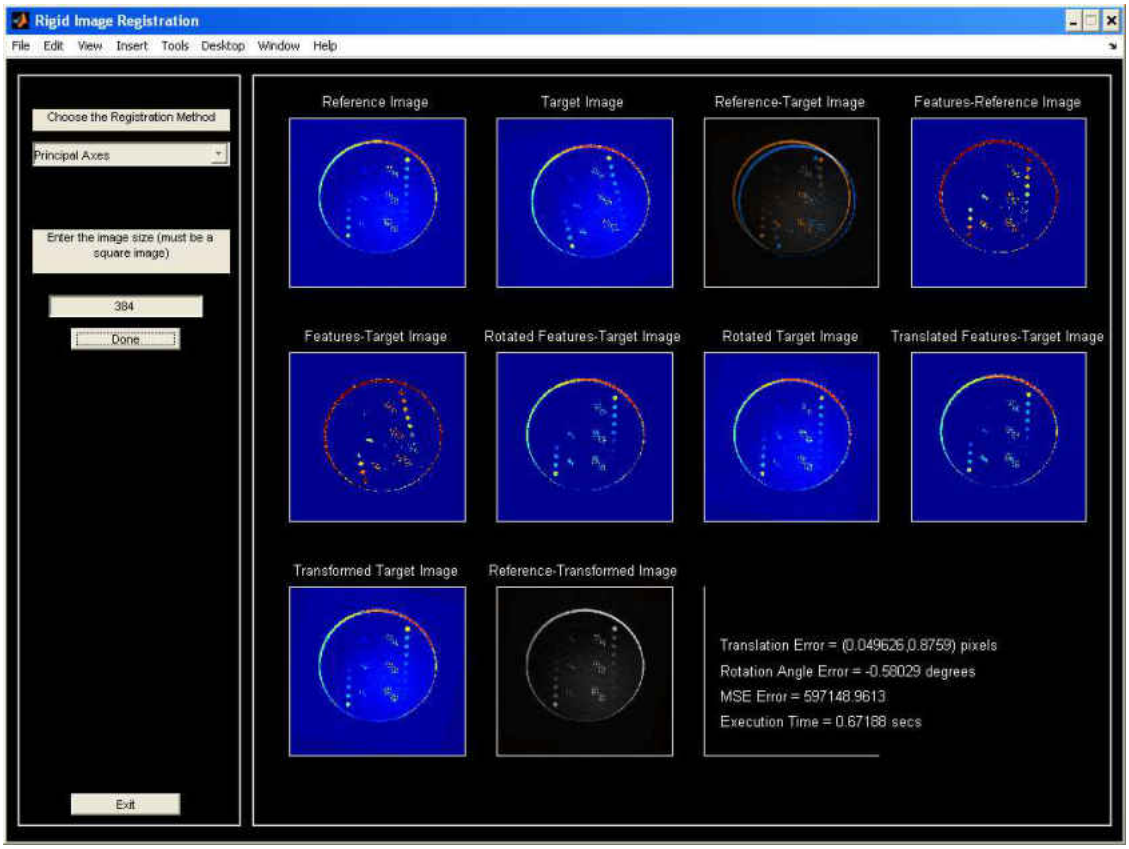
Target_5.raw : Translation = (-5,-5) Rotation=-5



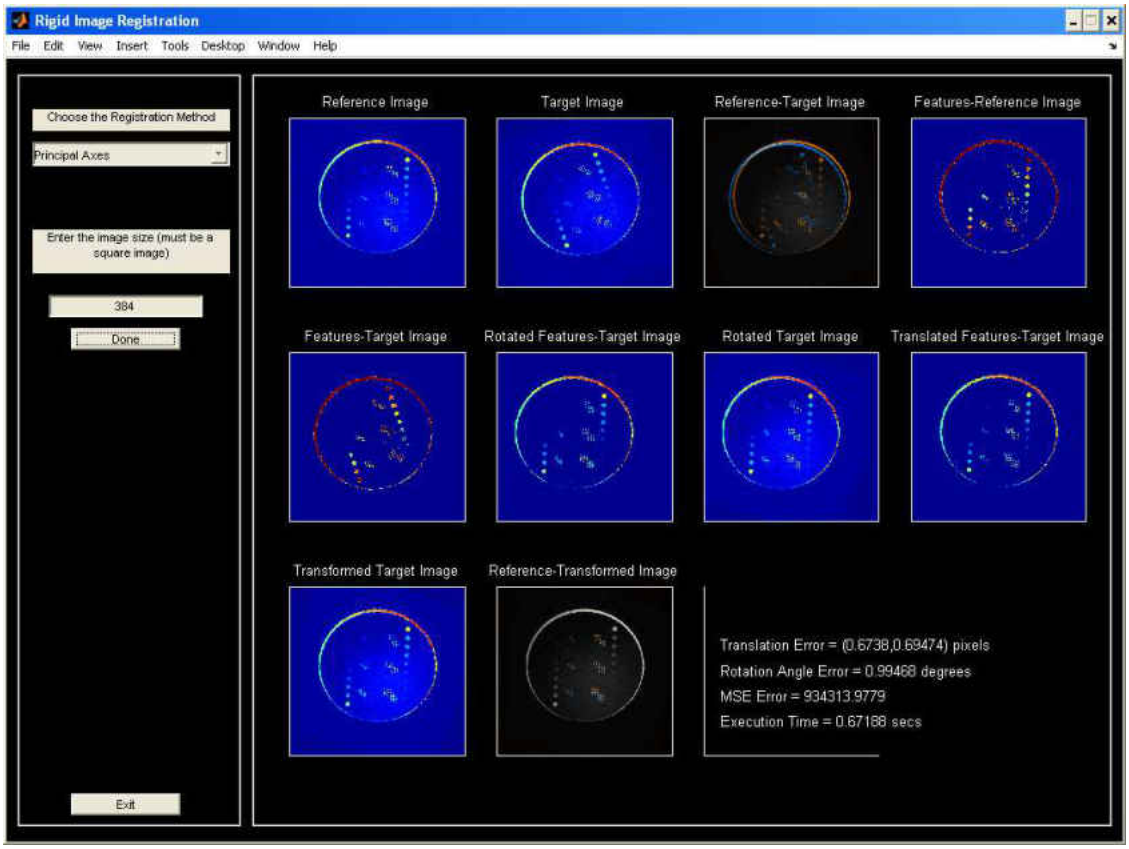
Target_10.raw: Translation = (0, 5) Rotation=-10



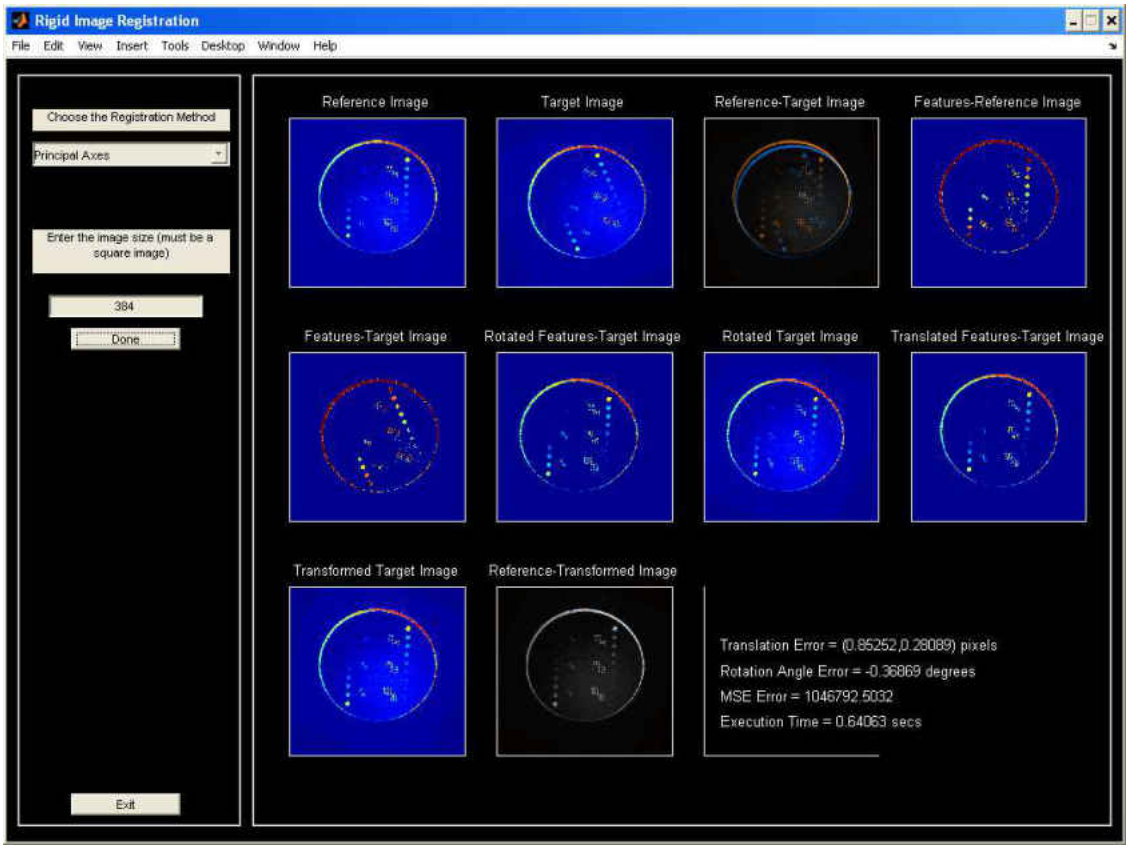
Target_15.raw : Translation = (15, 10) Rotation=-15



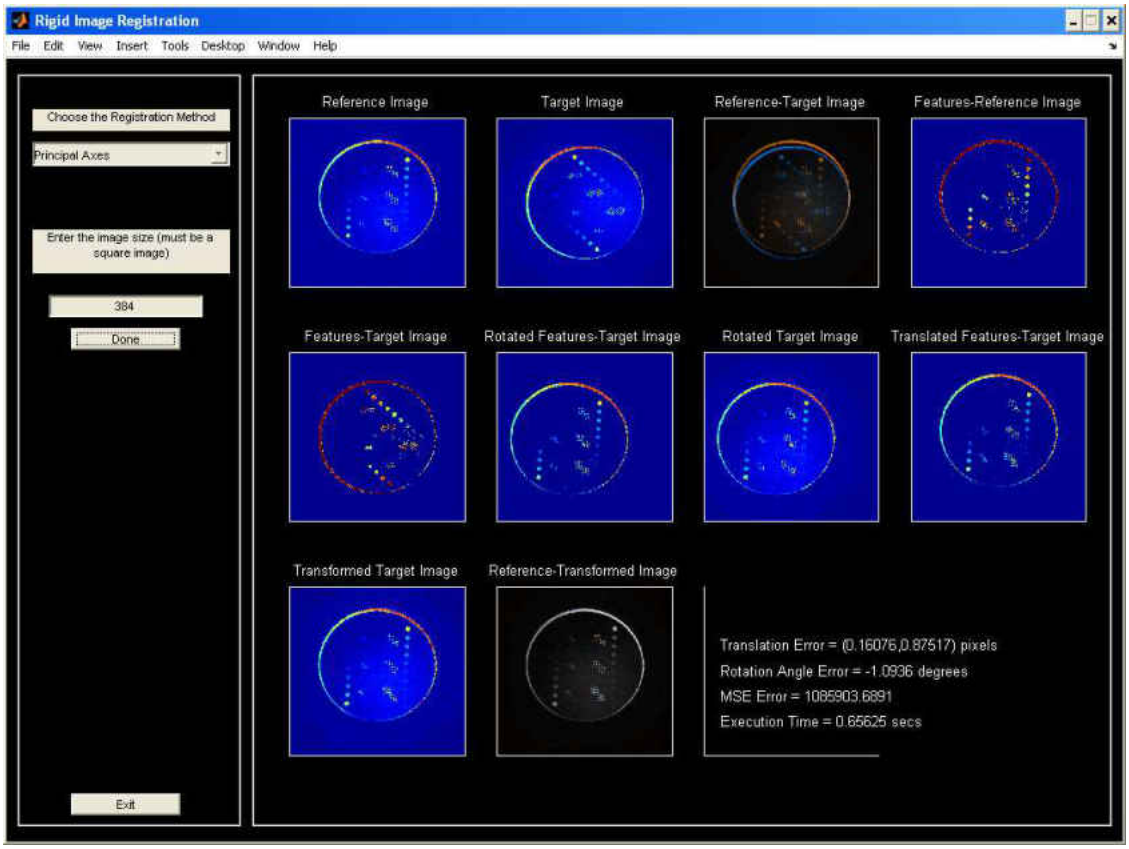
Target_20.raw : Translation = (-5, 5) Rotation=-20



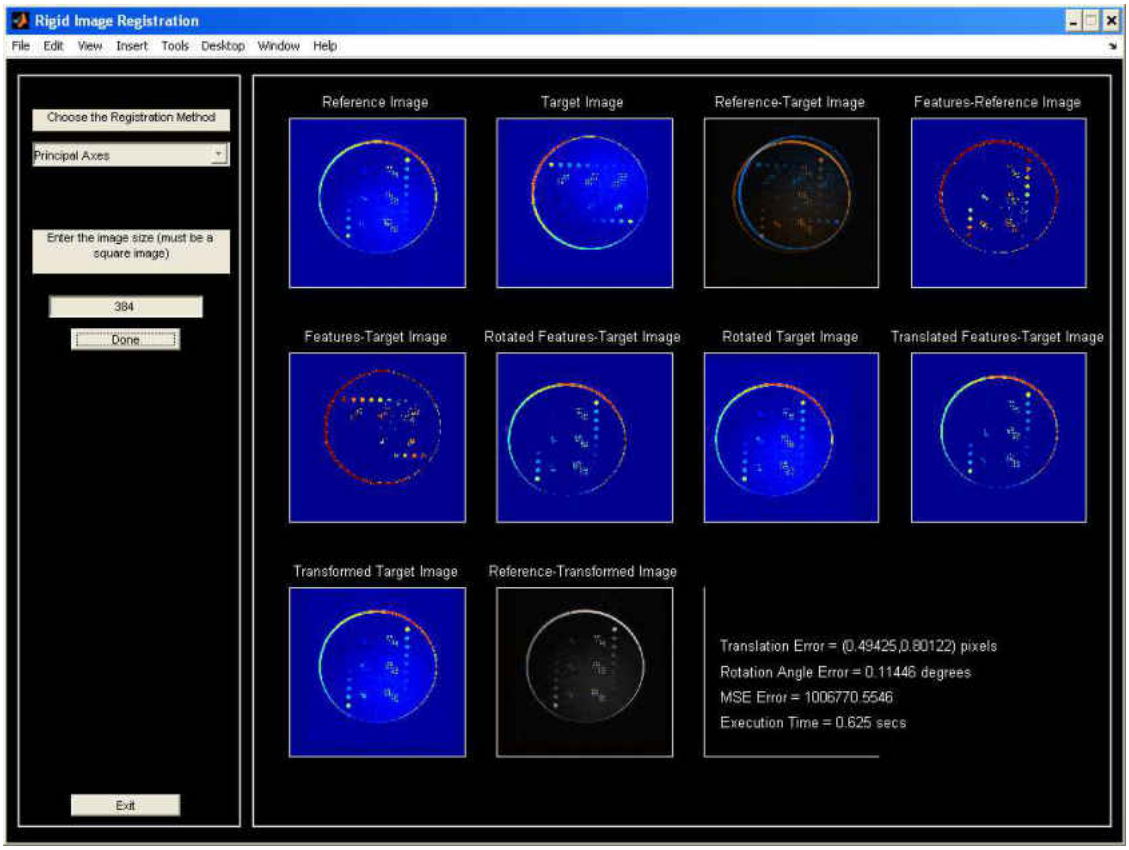
Target_25.raw : Translation = (10, 10) Rotation=-25



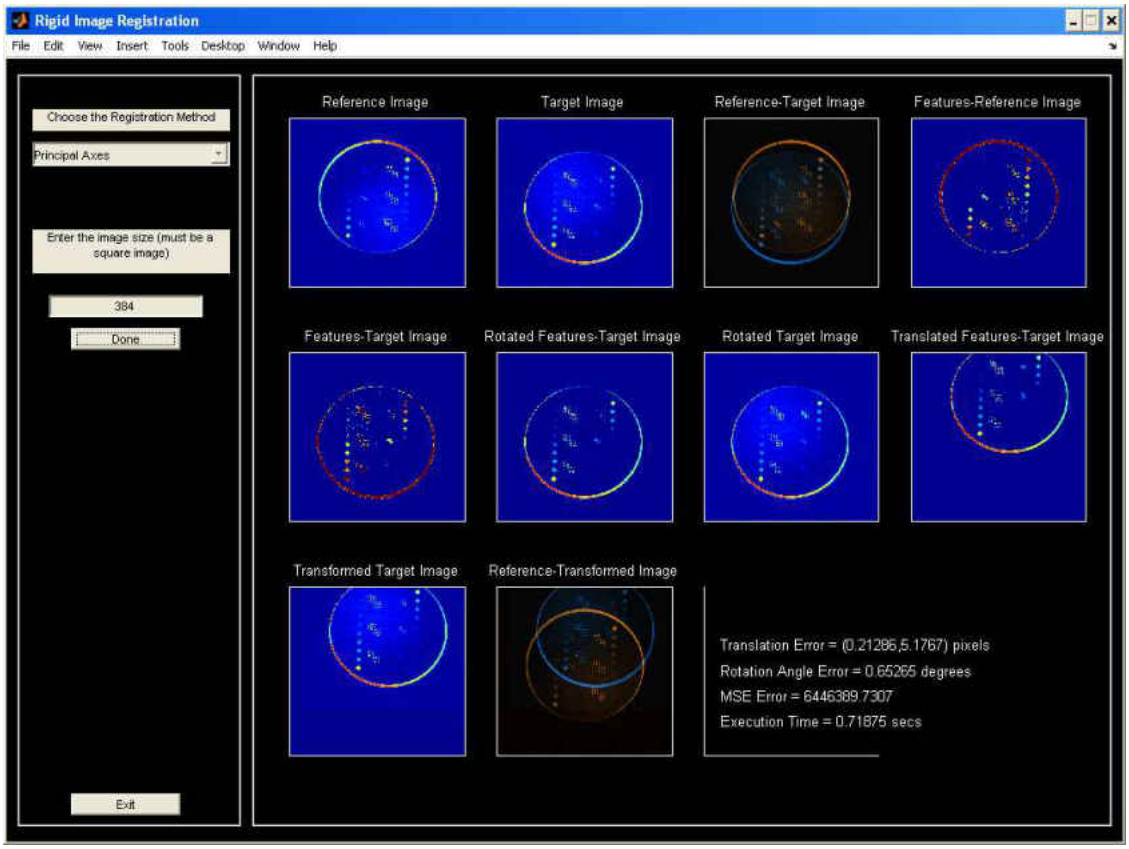
Target_50.raw : Translation = (10, 10) Rotation=-50



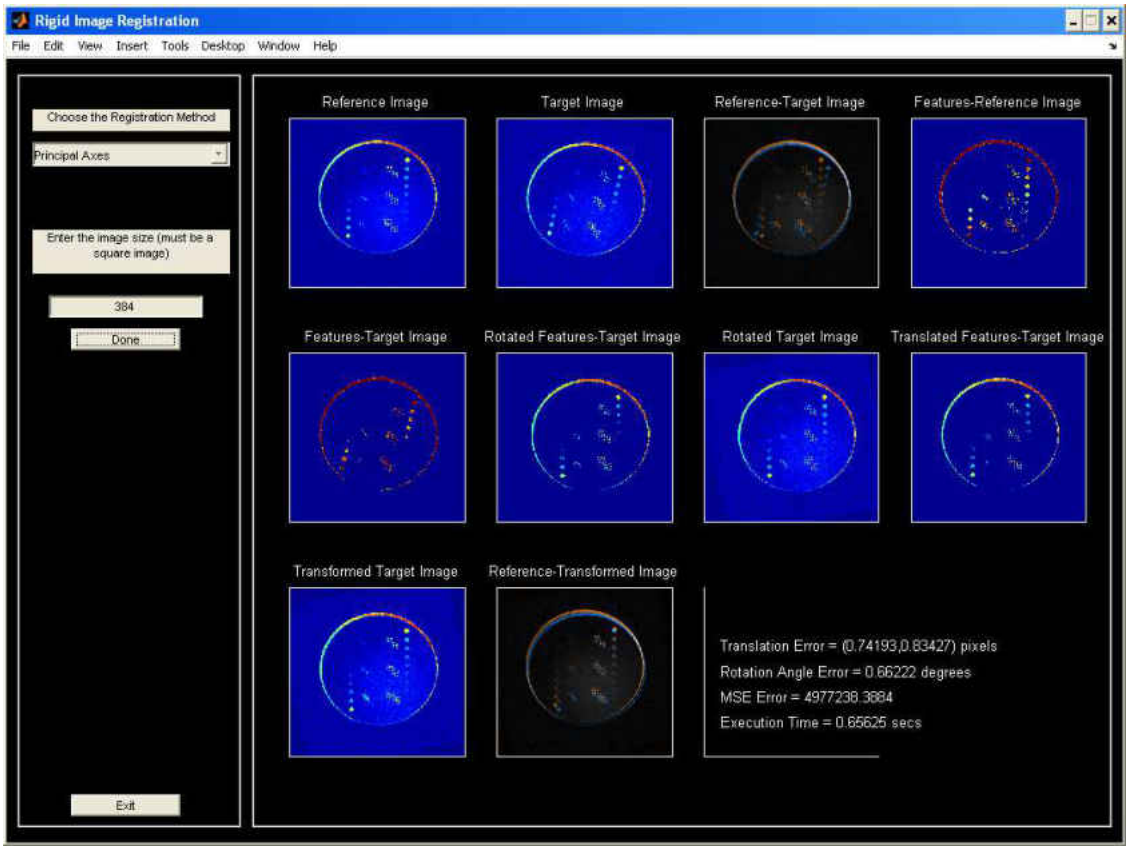
Target_90.raw : Translation = (5, 10) Rotation=-90



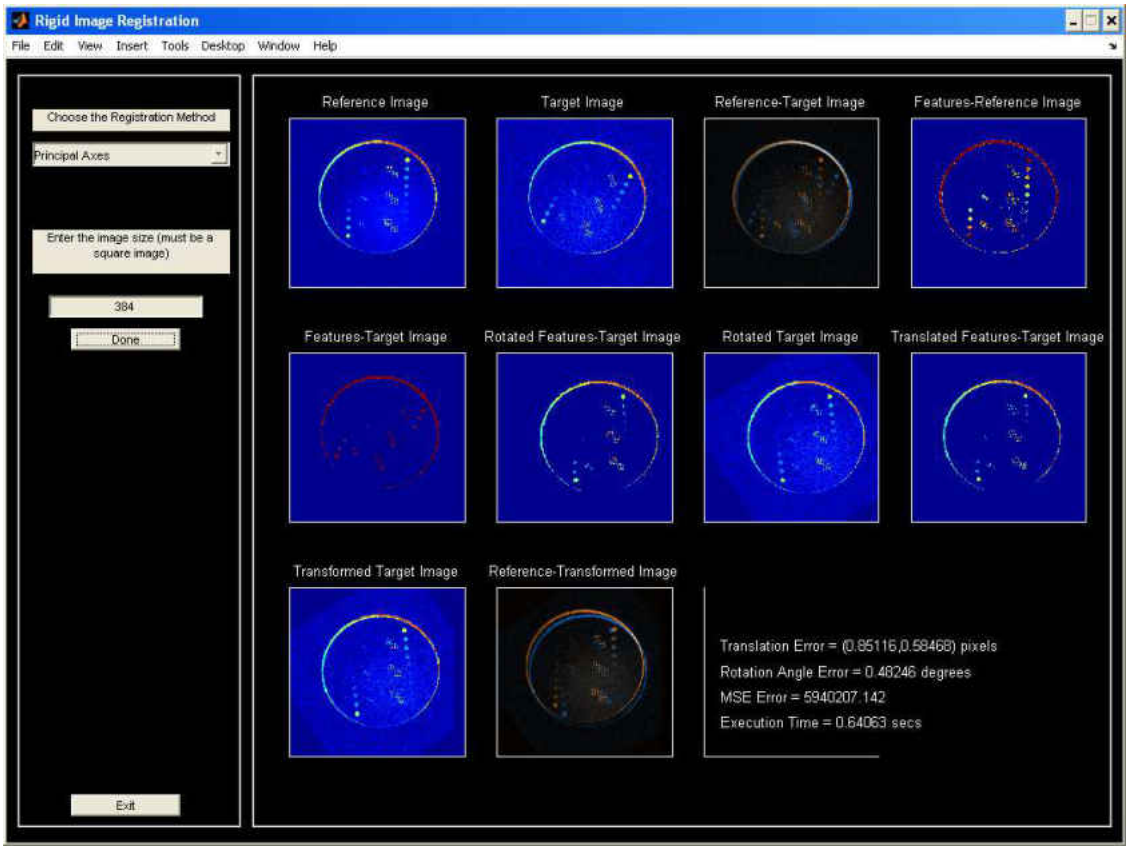
Target_180.raw : Translation = (0, 0) Rotation=-180



Target10n.raw : Translation = (0, 5) Rotation=10 Noise=500 std. dev.



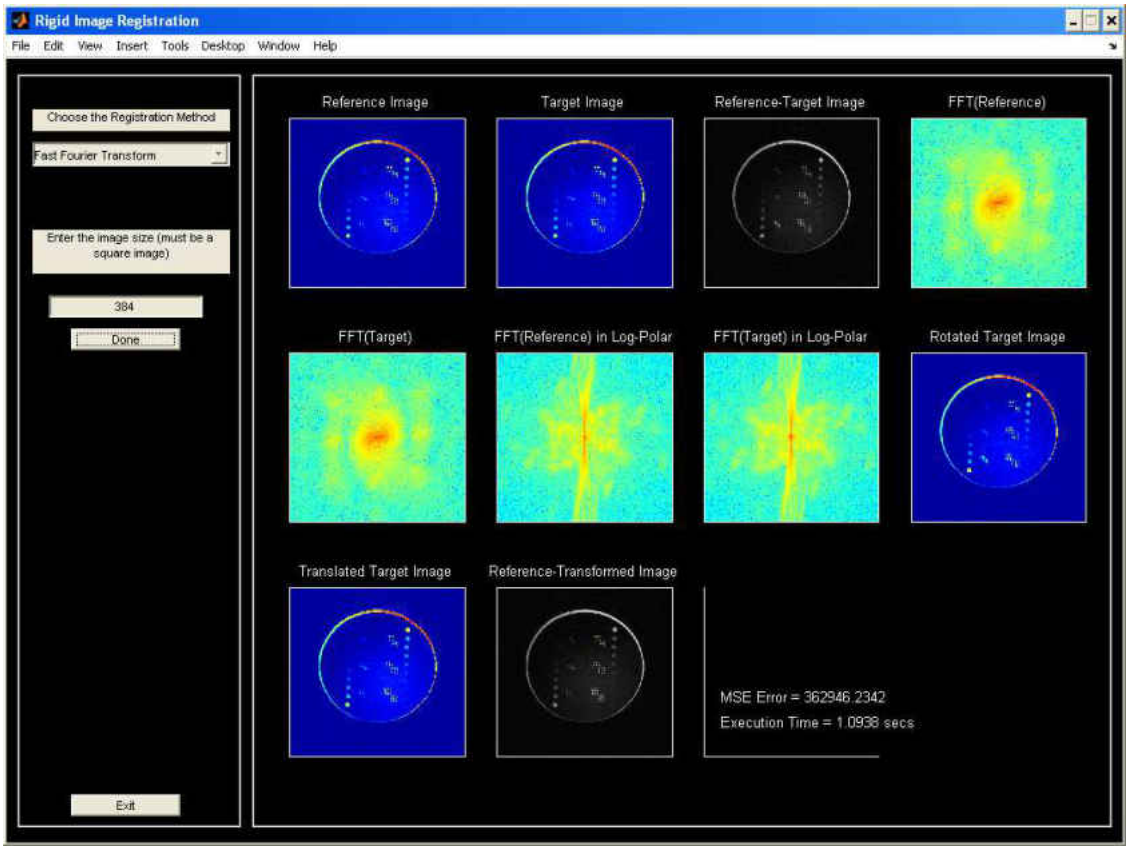
Target25n.raw : Translation = (-20, 20) Rotation=25 Noise=1000 std.dev.



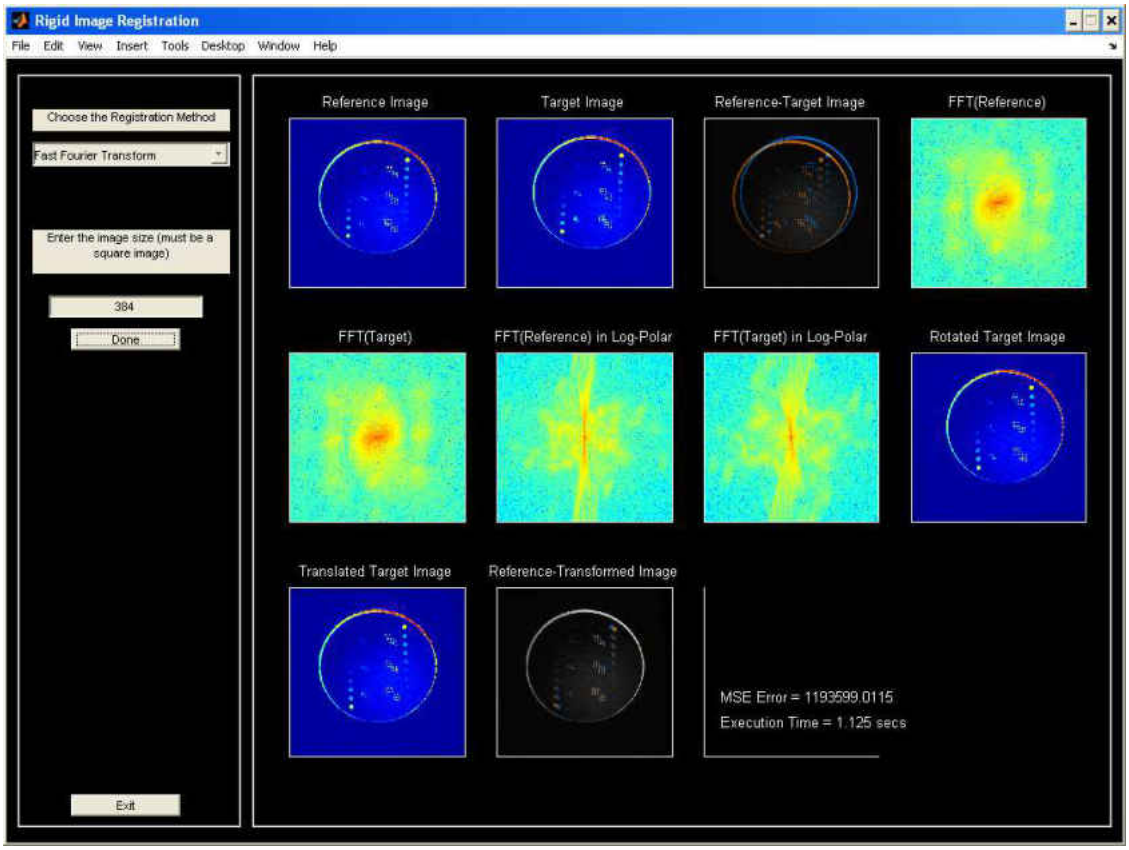
APPENDIX F

REGISTRATION OUTPUT WINDOWS USING FFT ALGORITHM

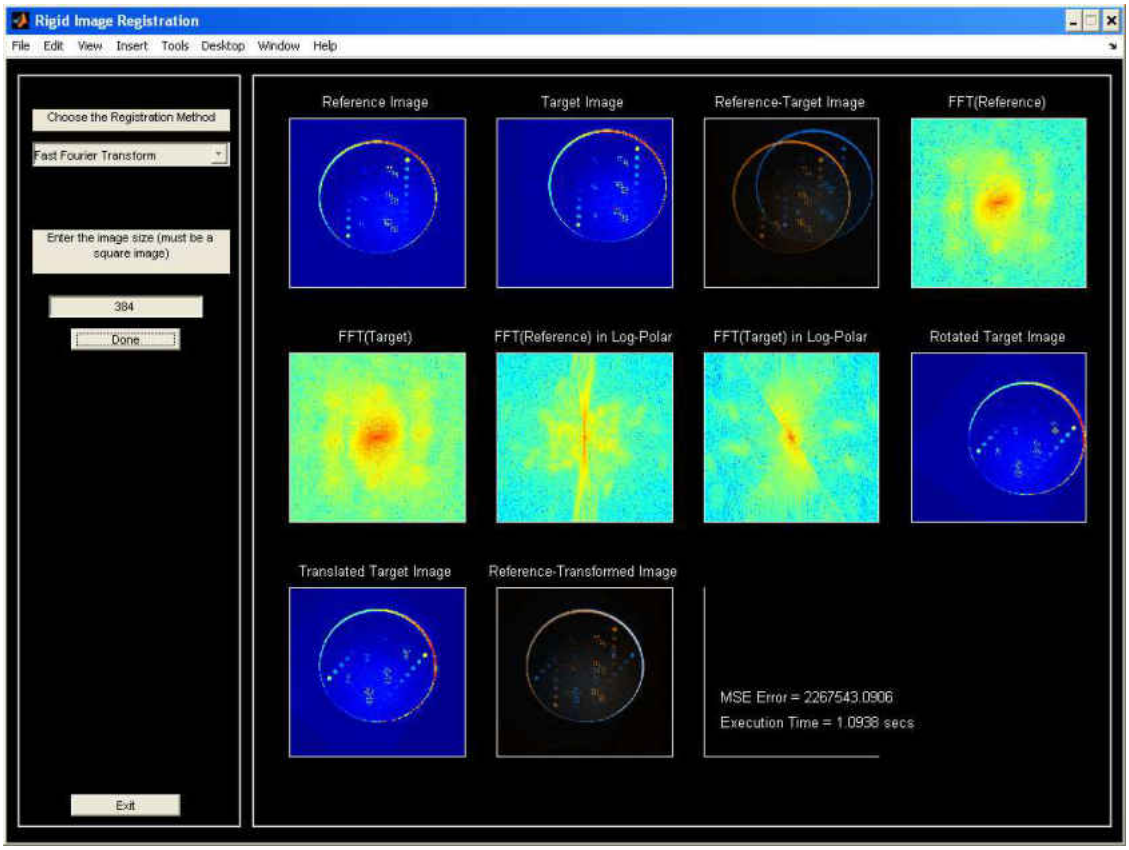
Target.raw : Translation = (0, 0) Rotation=0



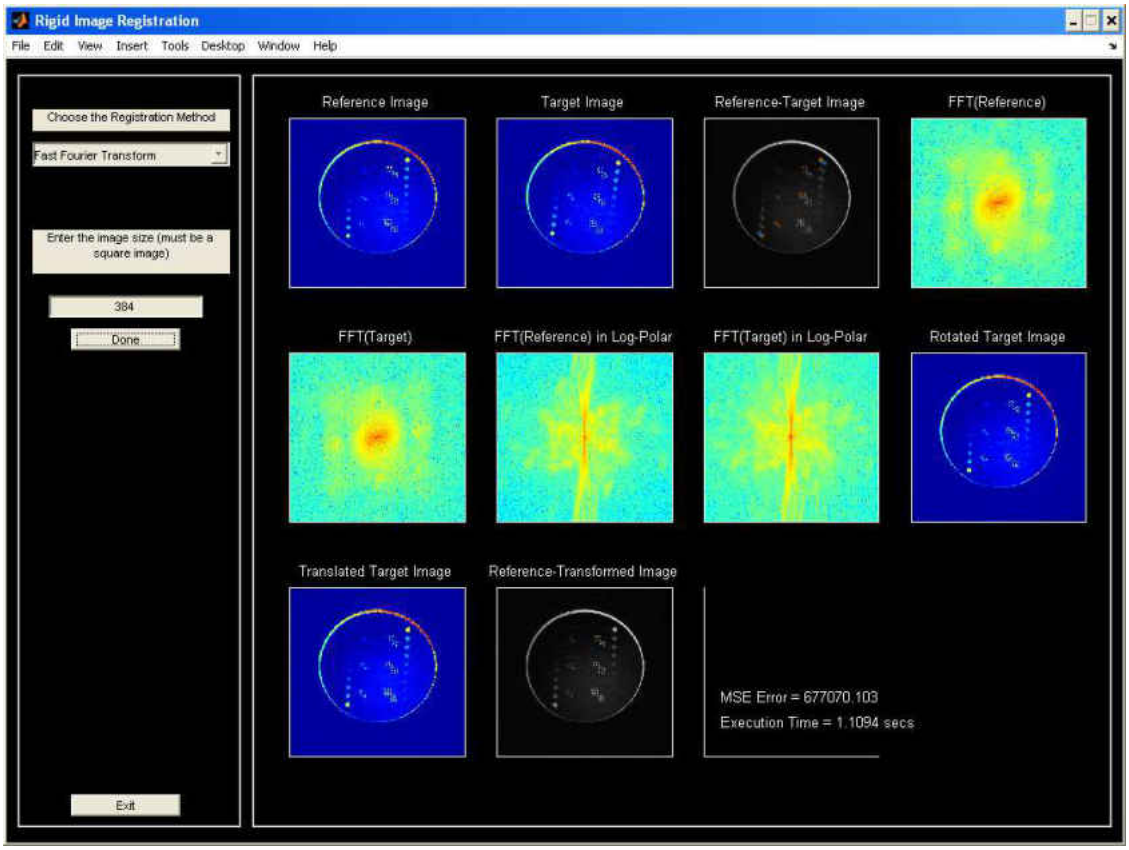
Target0.raw : Translation = (15,-10) Rotation=0



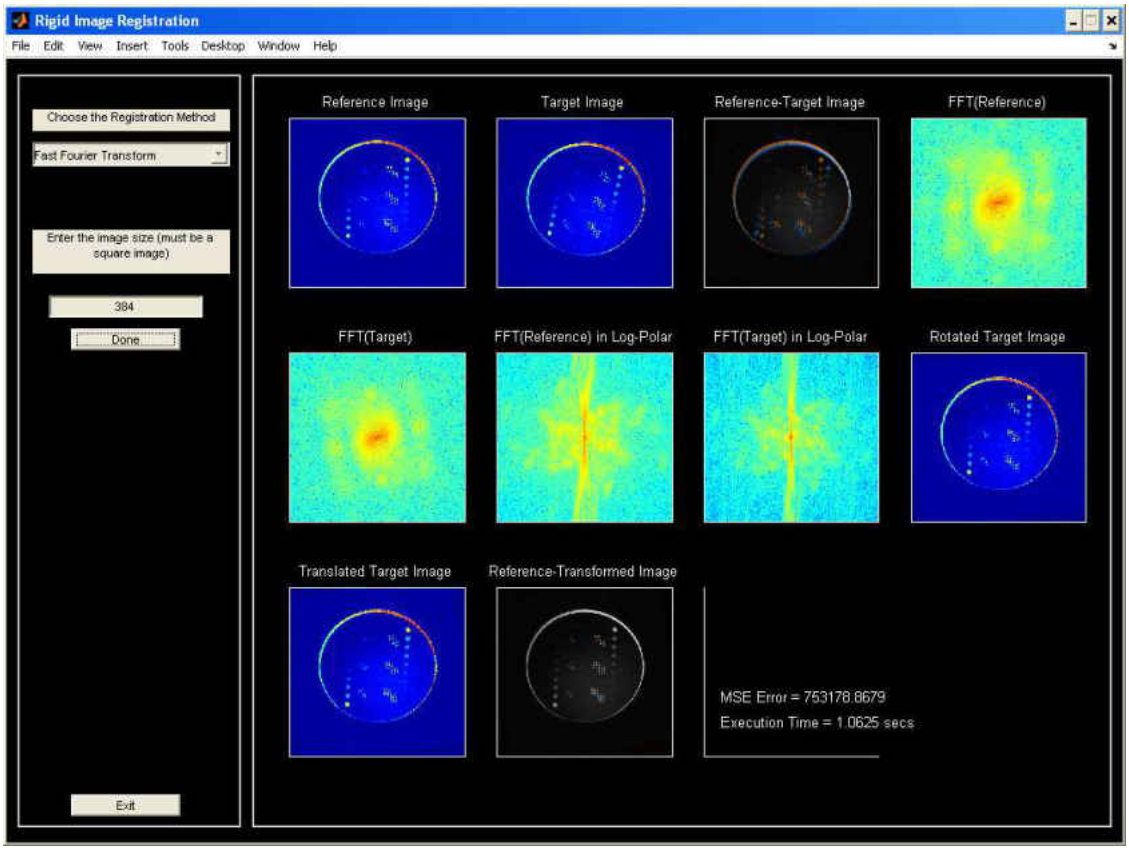
Target00.raw : Translation = (50,-25) Rotation=0



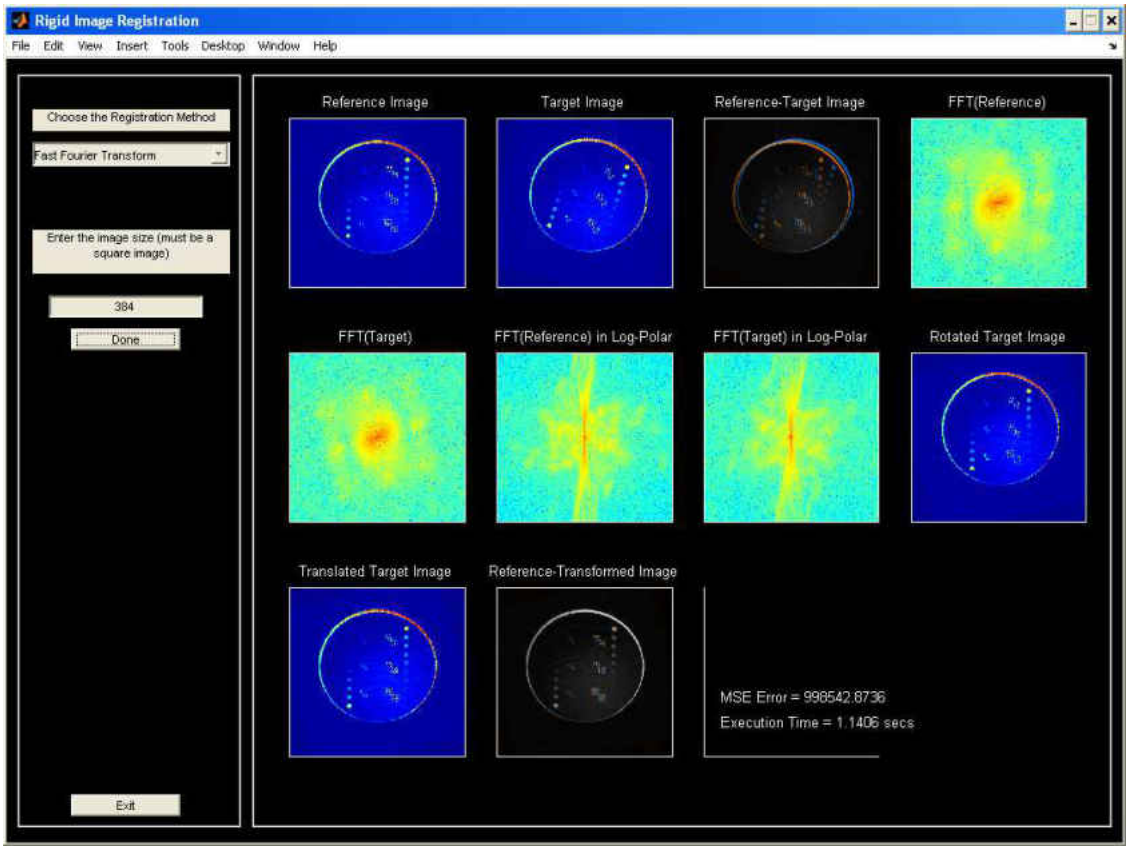
Target5.raw : Translation = (5,5) Rotation=5



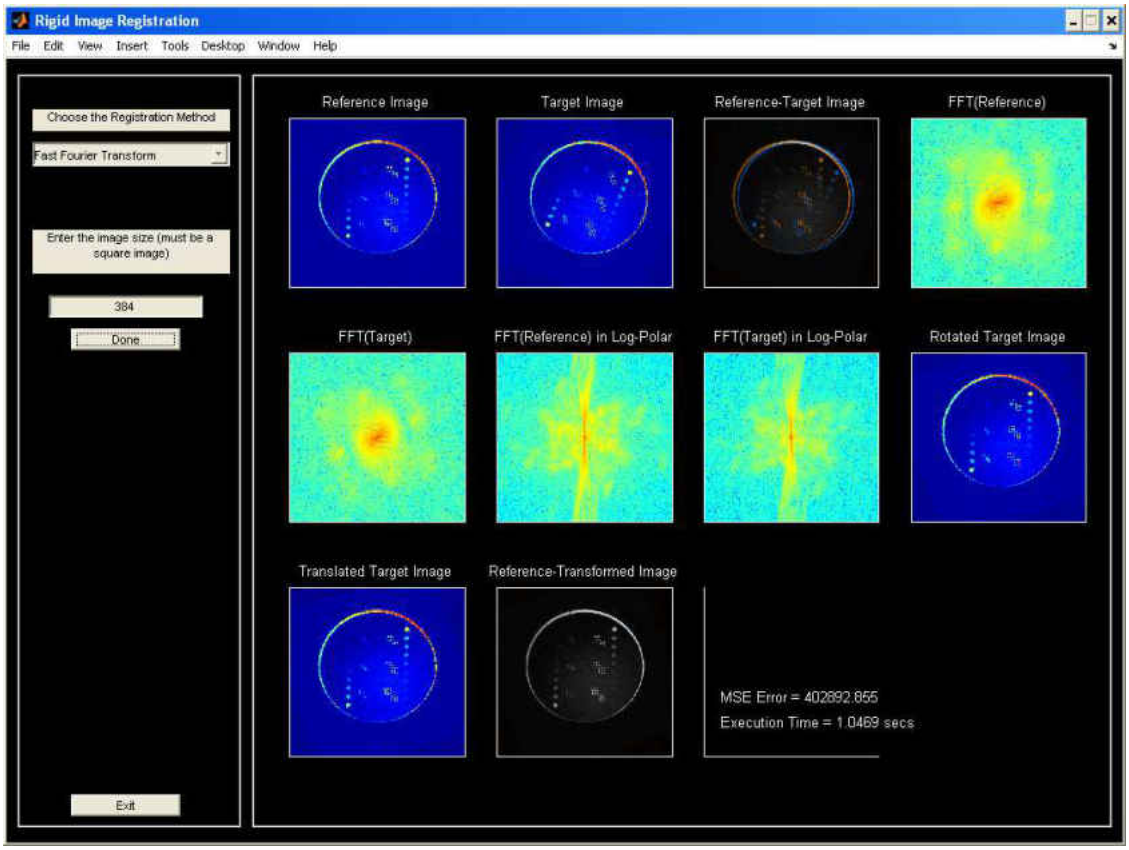
Target10.raw : Translation = (0,5) Rotation=10



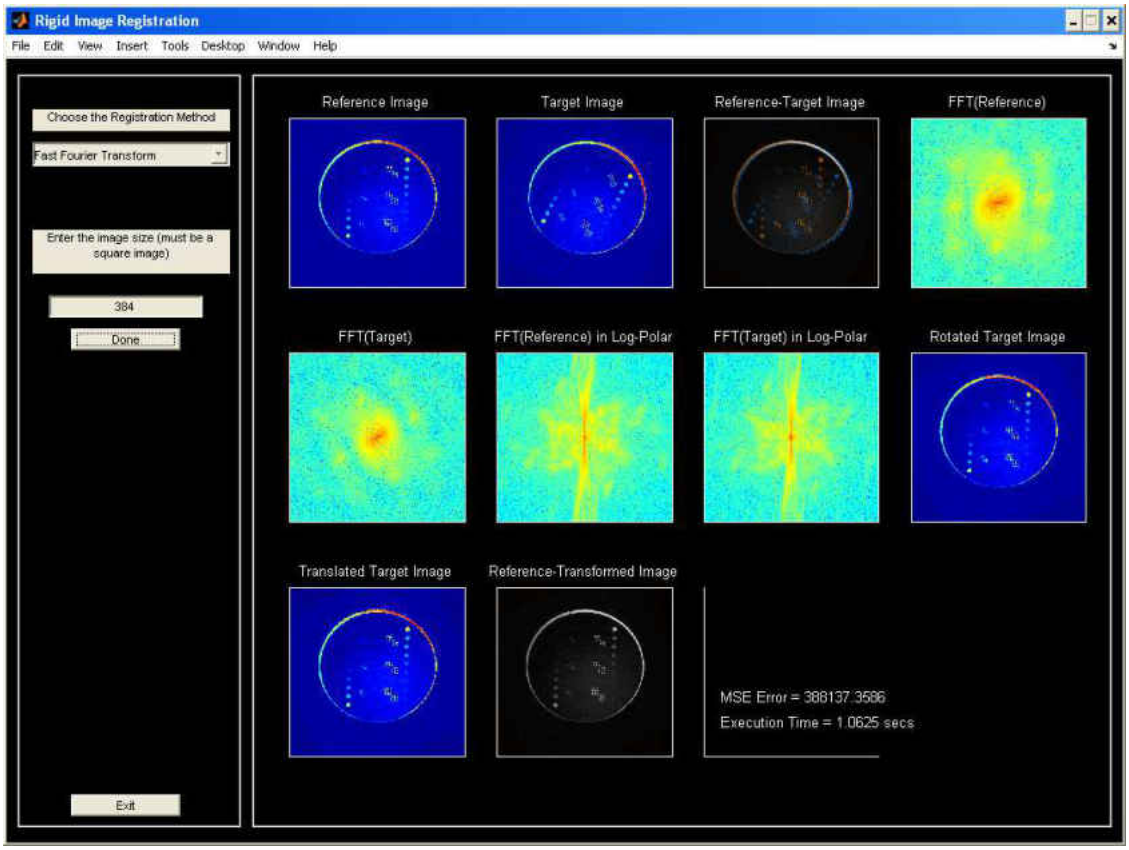
Target15.raw : Translation = (15,10) Rotation=15



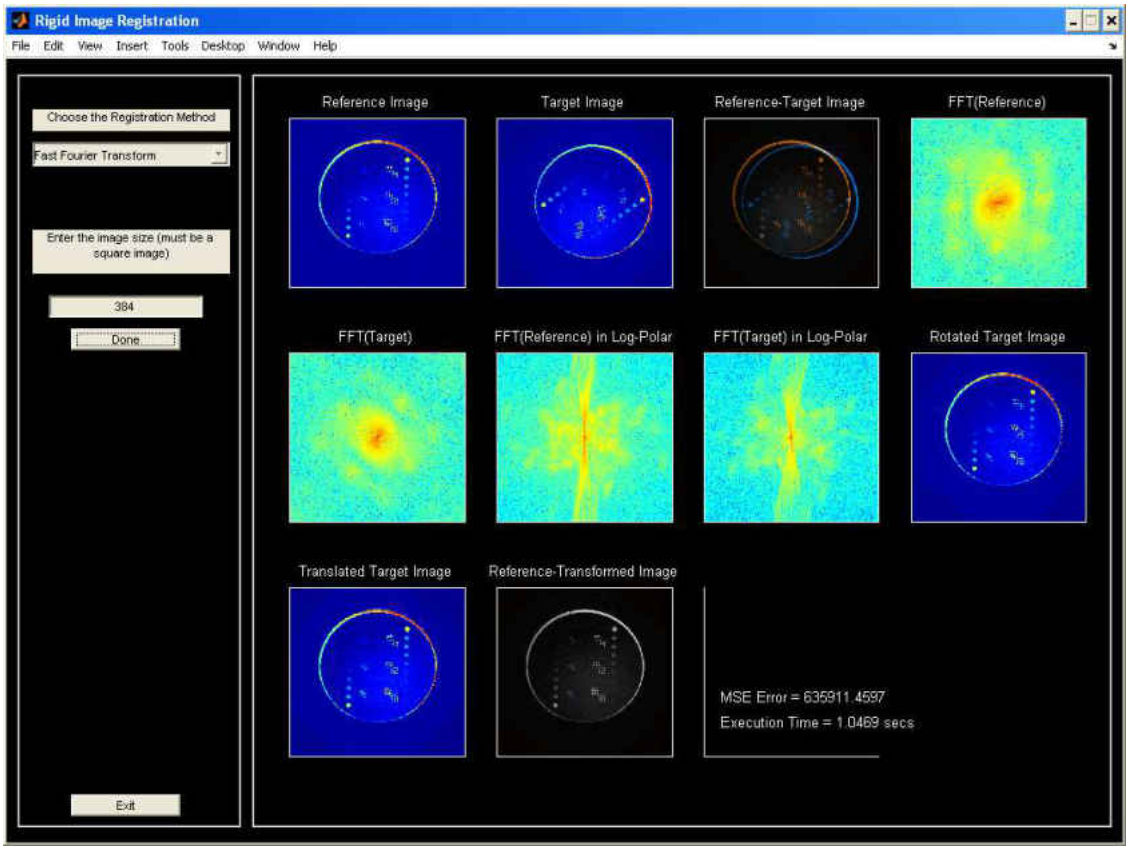
Target20.raw : Translation = (10,20) Rotation=20



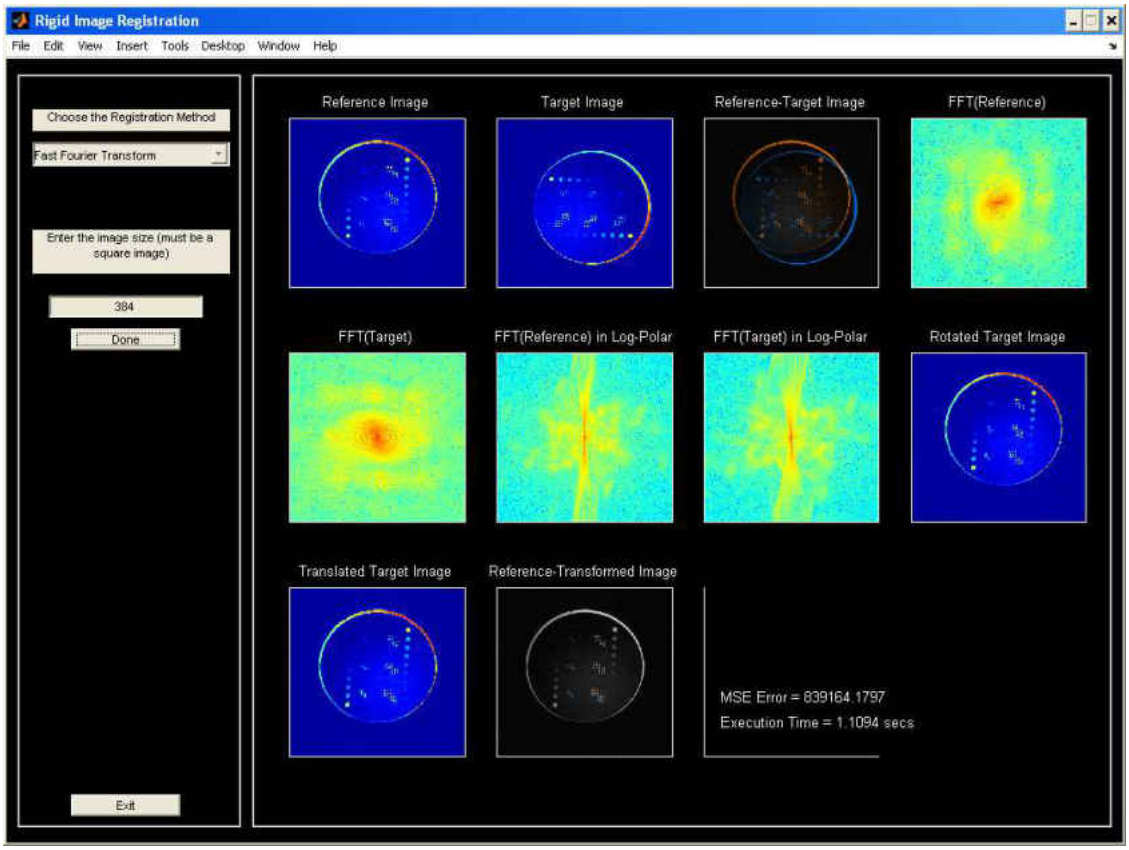
Target25.raw : Translation = (-20,20) Rotation=25



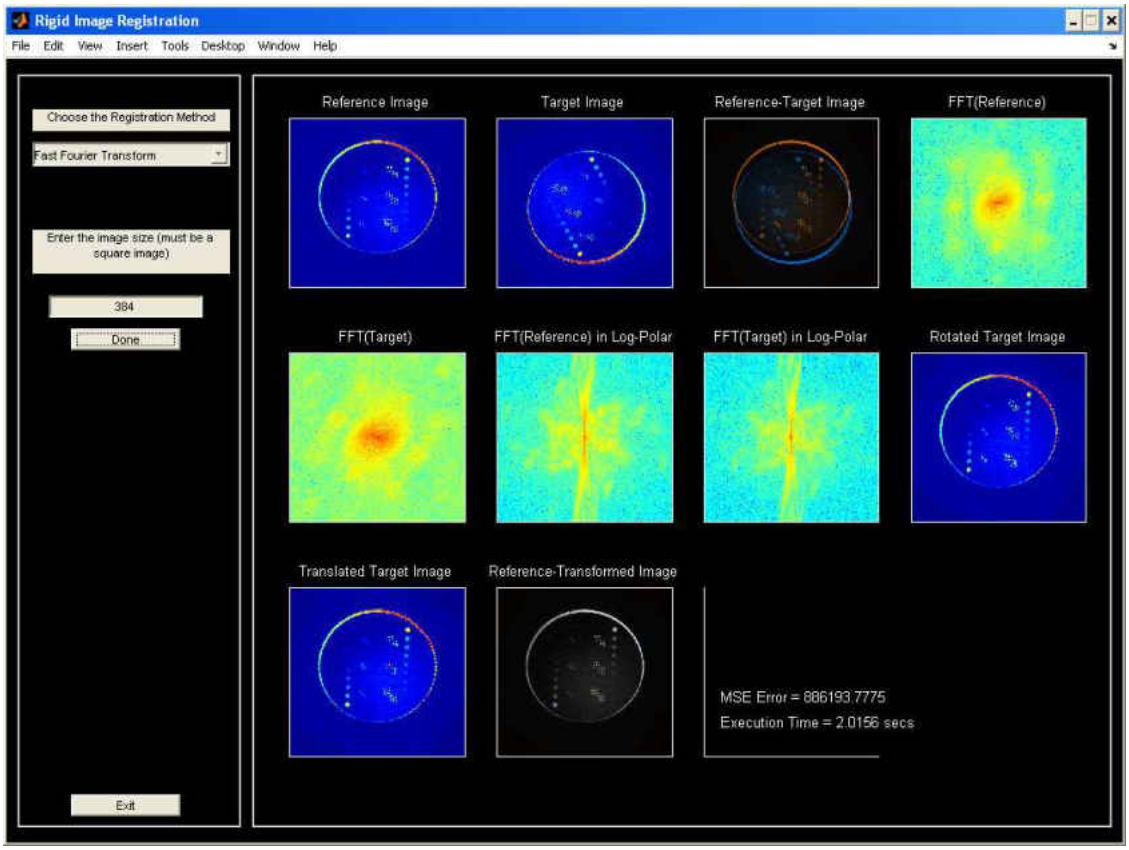
Target50.raw : Translation = (25,-25) Rotation=50



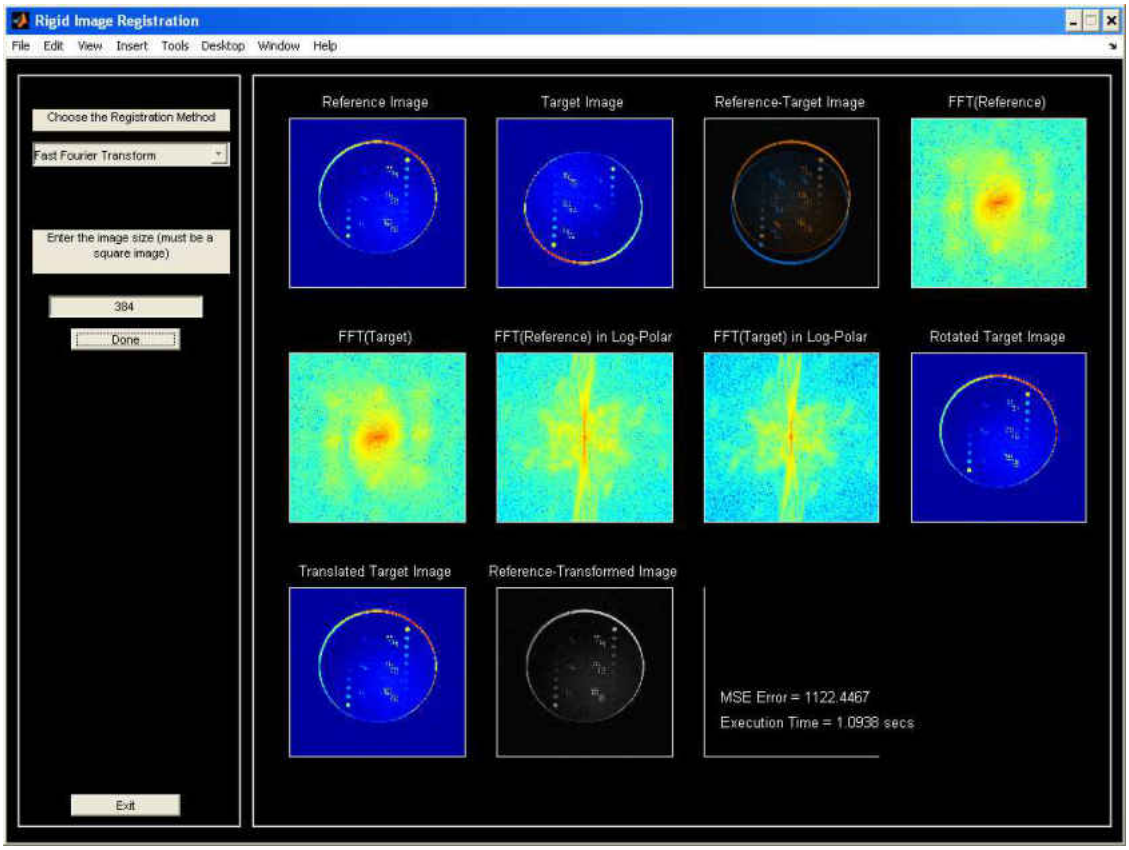
Target90.raw : Translation = (5,10) Rotation=90



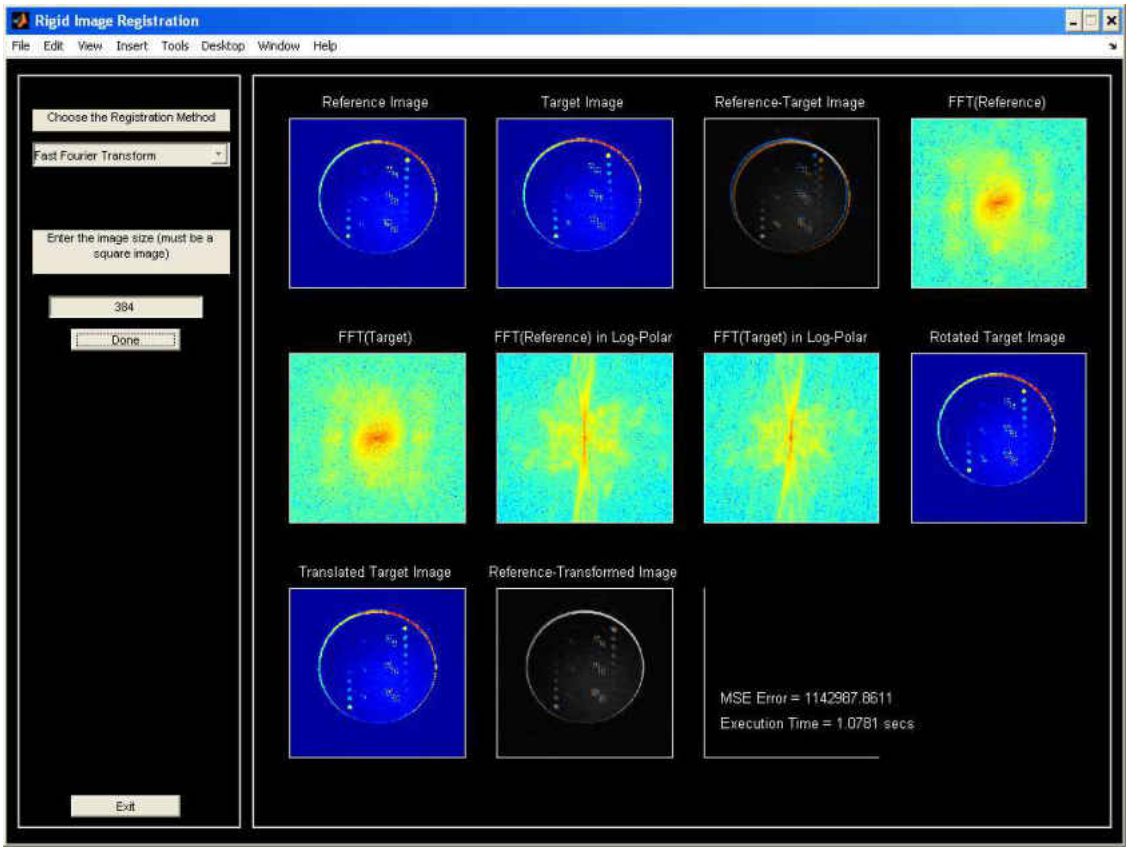
Target150.raw : Translation = (0,0) Rotation=150



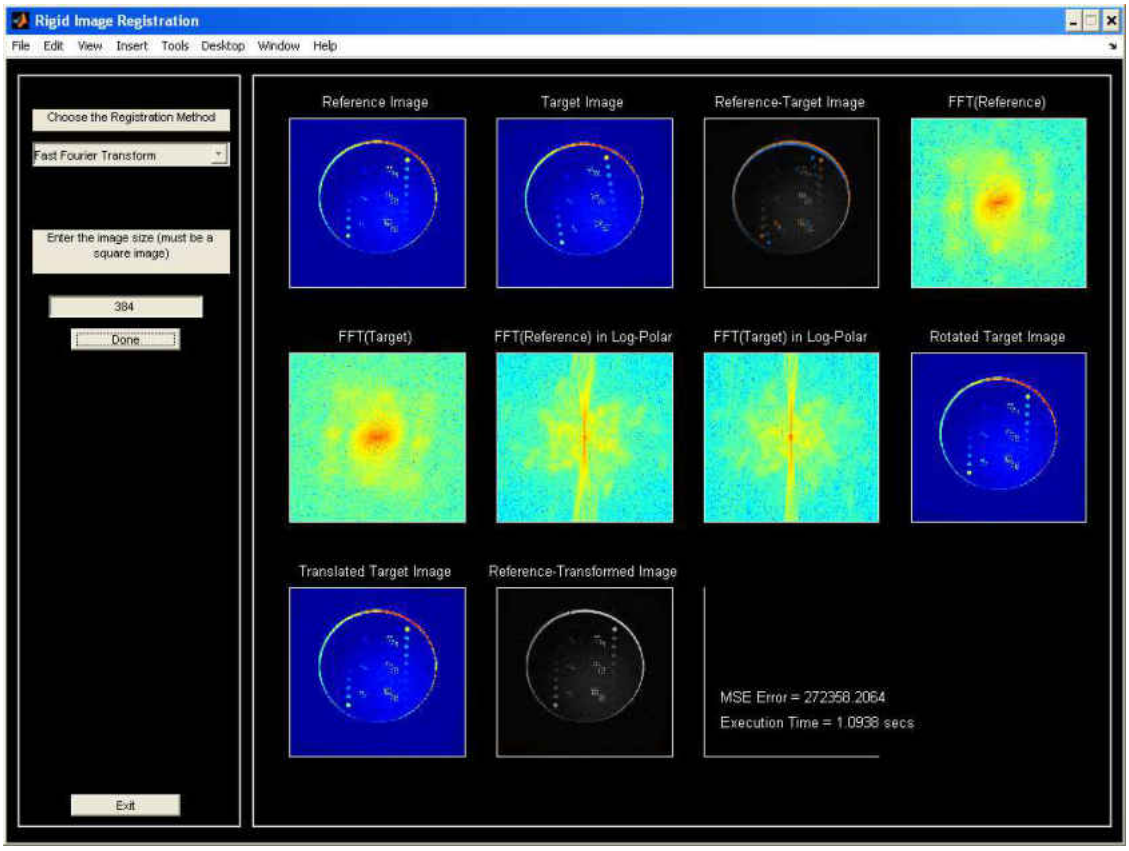
Target180.raw : Translation = (0,0) Rotation=180



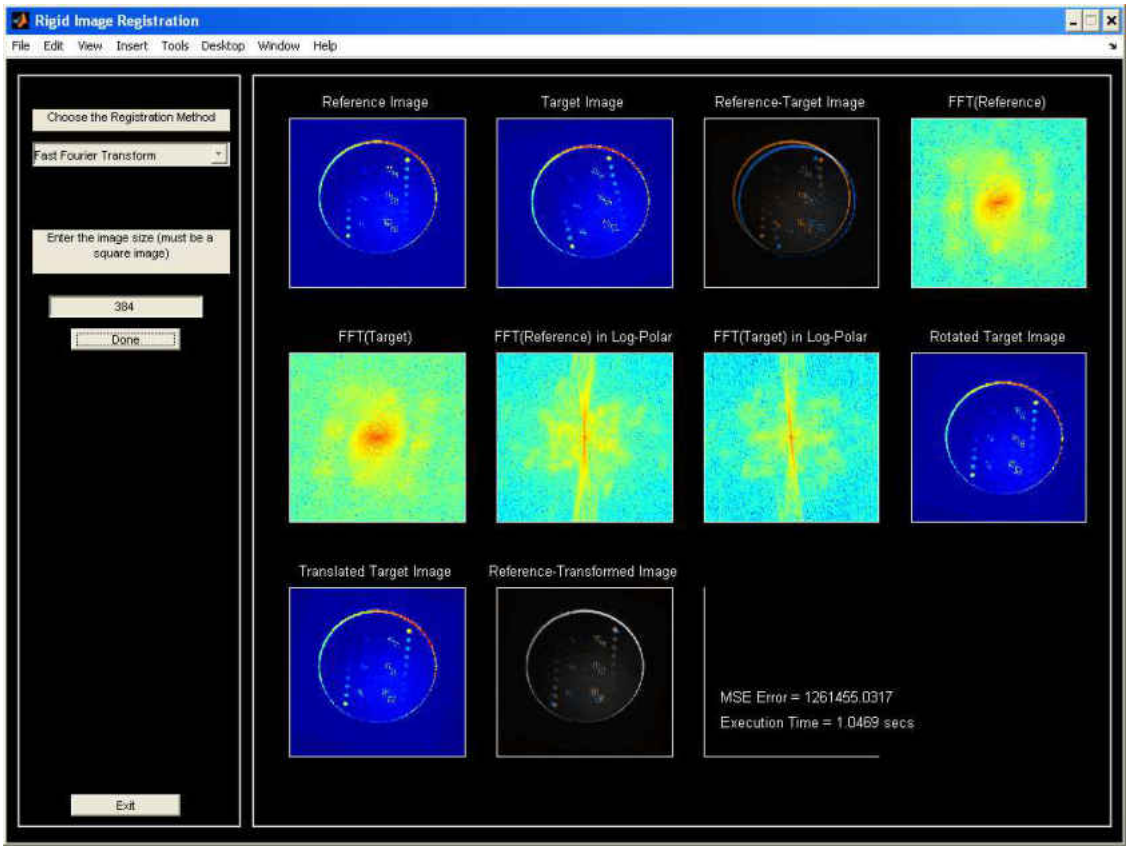
Target_5.raw : Translation = (-5,-5) Rotation=-5



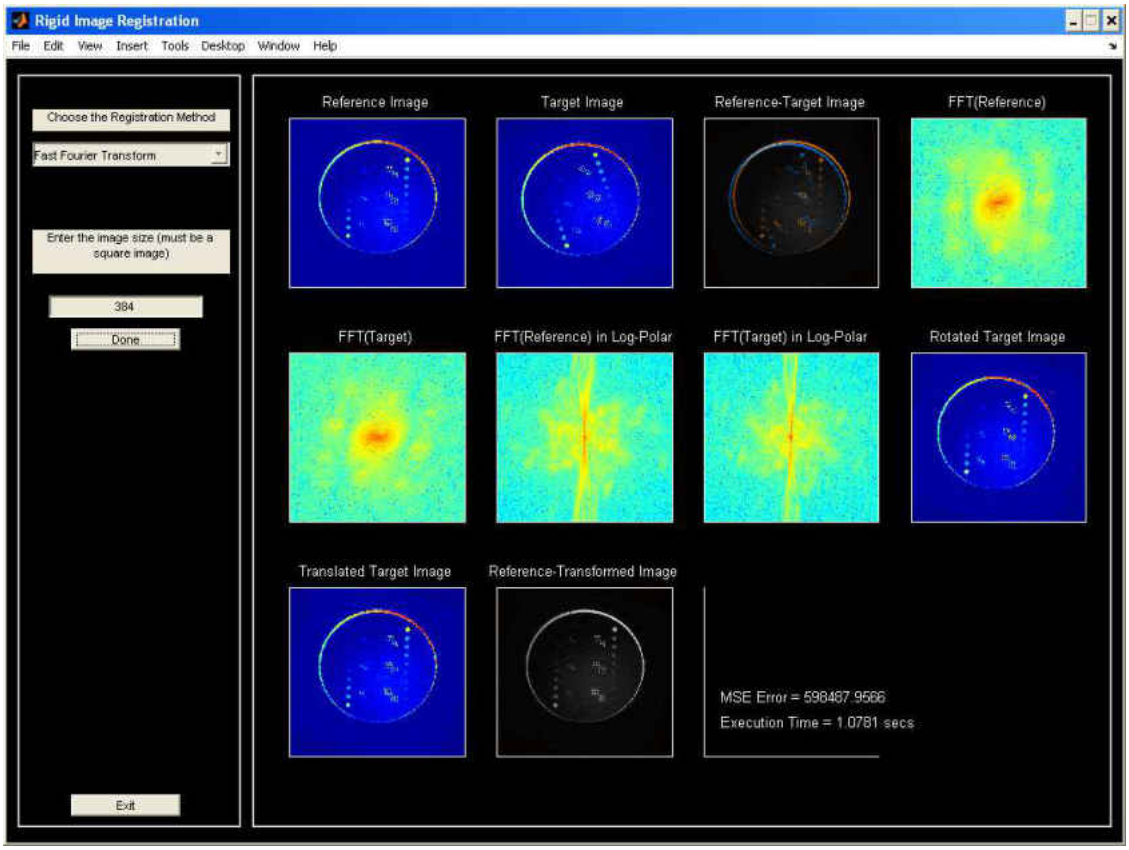
Target_10.raw : Translation = (0,5)
Rotation=10



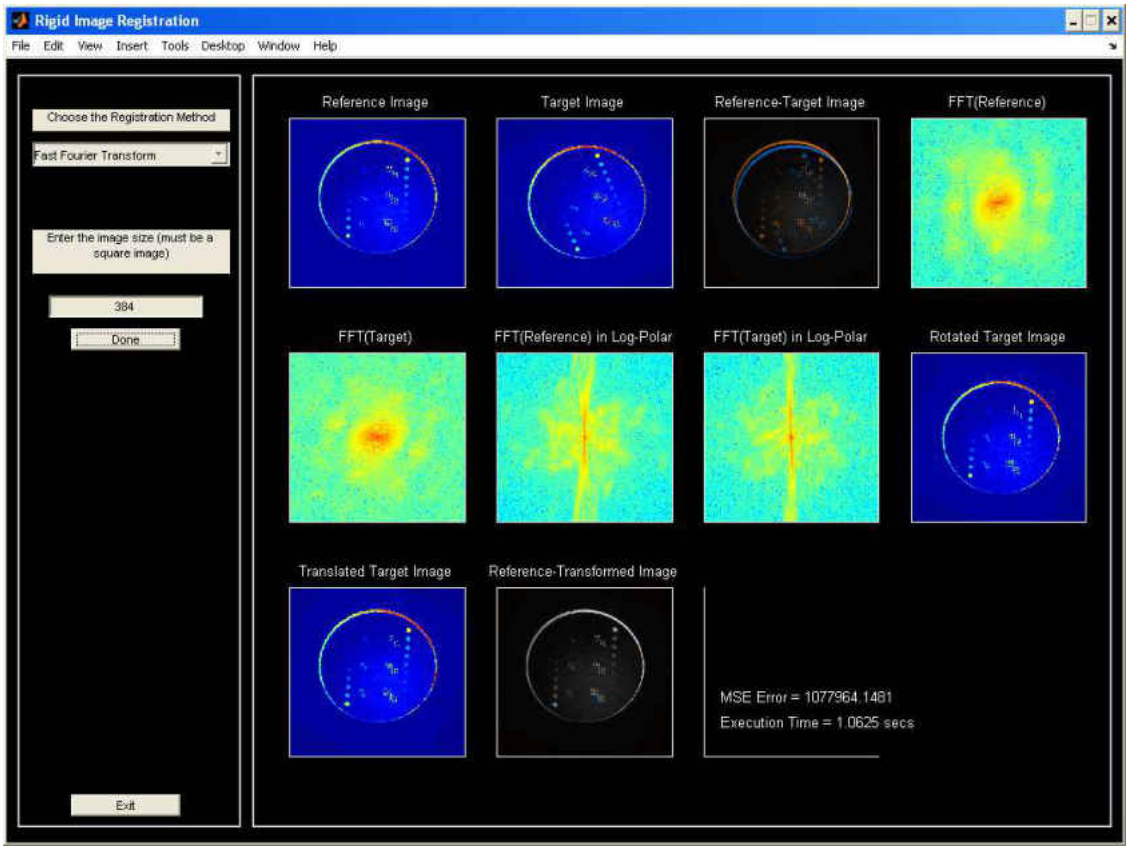
Target_15.raw : Translation = (15,10) Rotation=-15



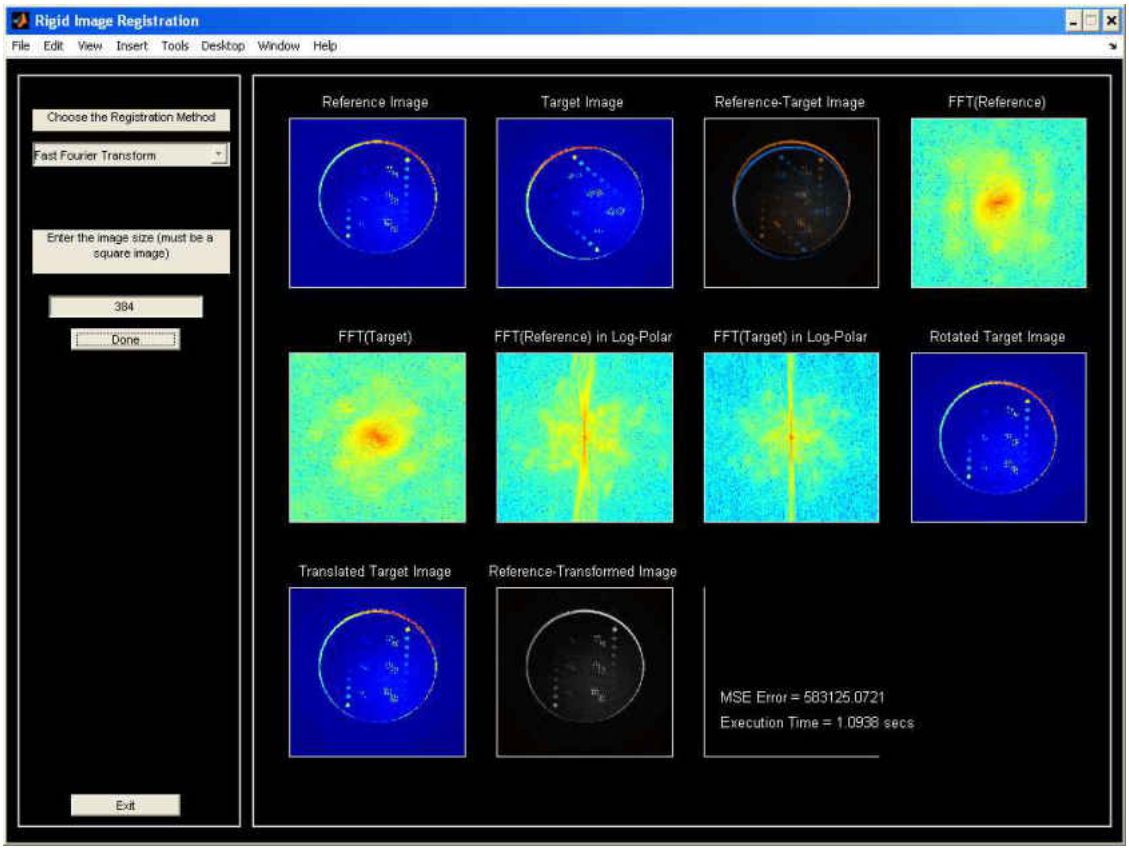
Target_20.raw : Translation = (-5,5) Rotation=-20



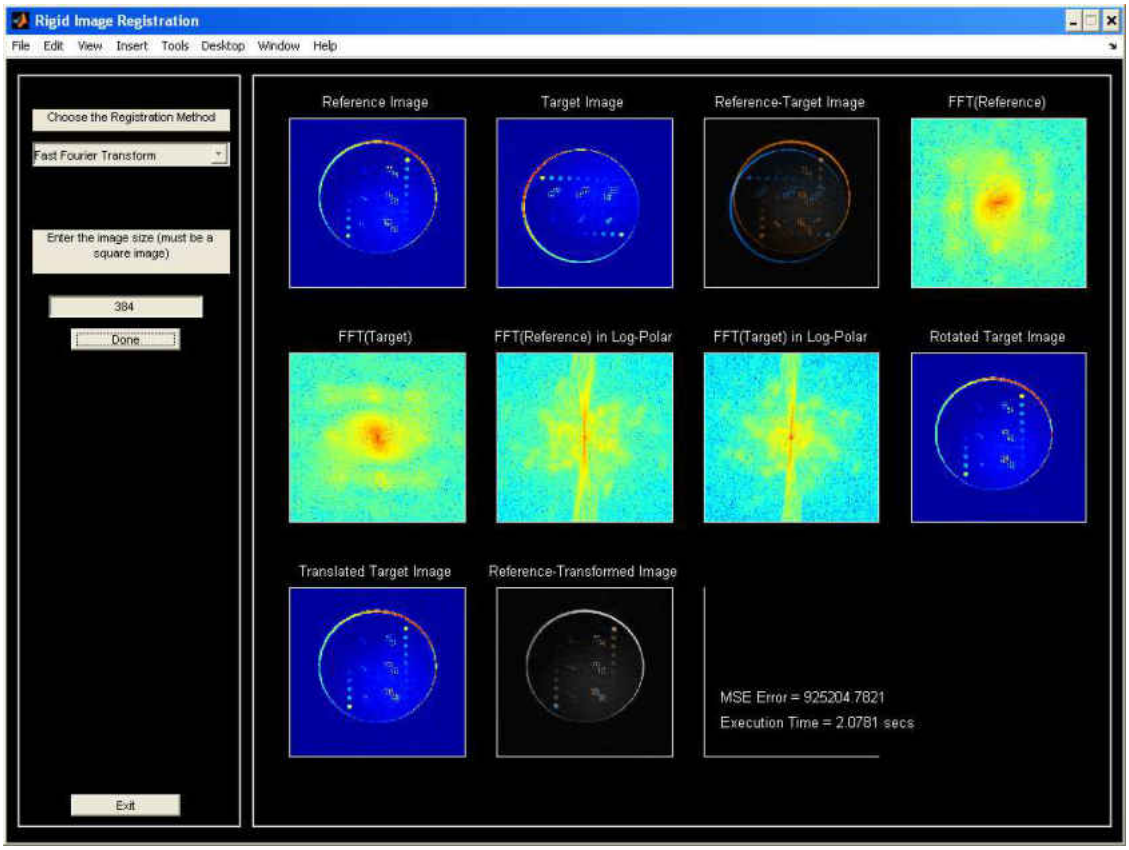
Target_25.raw : Translation = (10,10) Rotation=-25



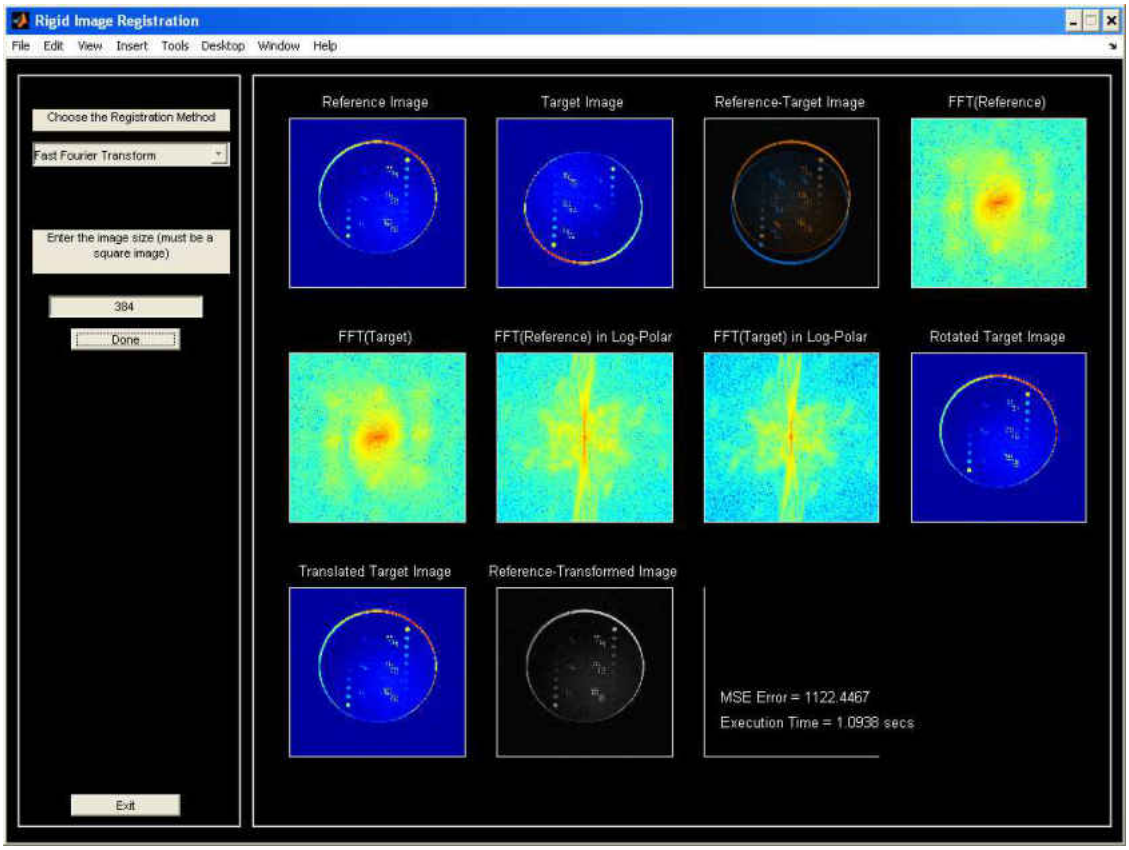
Target_50.raw : Translation = (10,10) Rotation=-50



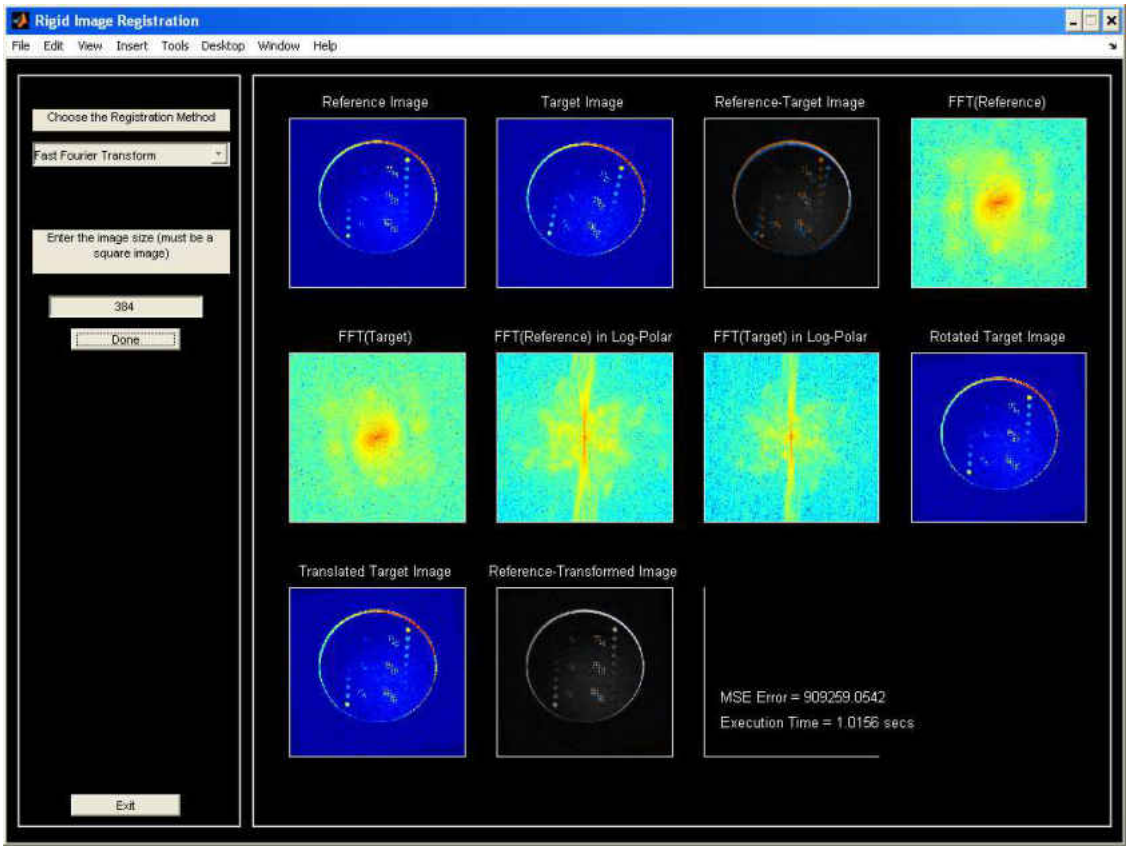
Target_90.raw : Translation = (5,10) Rotation=-90



Target_180.raw : Translation = (0,0) Rotation=-180



Target10n.raw : Translation = (0, 5) Rotation=10 Noise=500 std. dev.



Target25n.raw : Translation = (-20, 20) Rotation=25 Noise=1000 std.dev.