3D Tomography in Transmission Electron Microscopy

Optimizing alignment using Affine Transform and Scale Invariant Feature Transform

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry
Department of Applied Physics
Name: Muhammad Arshad Haroon
MSc thesis:
Date: September 09, 2015

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry
Department of Applied Physics
Muhammad Arshad Haroon: 3D Tomography in Transmission Electron Microscopy: Optimizing alignment using Affine Transform and Scale Invariant Feature Transform
MSc thesis: 62 pages
Supervisors: Arto Koistinen, Ph.D., Stefanos Georgiadis, Ph.D.
September 2015

Key words: Electron Tomography, tomographs, electron microscopy, fiducial markers ,alignment, volumetric reconstruction ,Affine Transform, Scale Invariant Feature Transform

ABSTRACT

An accurate implementation of the Electron Tomography can reveal useful information e.g. from the subcellular structures in biological samples of the tomographs acquired using electron microscopy. Three dimensional (3D) electron tomography helps studying the presence of different layers and nuclei in the cell tissues, their geometry and their natural formation. This thesis work includes a background study of the Electron Tomography in different fields as well as the implementation of its processes.

The hierarchical processes of 3D Electron tomography were studied and implemented to align and reconstruct the microscopic data of two difference samples prior to the segmentation and visualization. Implementation of the different image registration algorithms and the usage of tomography software suggest that a near error free volume reconstruction is only possible if the microscopy images (tomographs) are well aligned before continuing towards the volumetric reconstruction.

Methods such as alignment via fiducials and markerless alignment together with feature points provided very useful results. Alignment via fiducials produced an exact alignment while the process of markerless alignment was tested with the introduction of manually selected features, Affine Transform and Scale Invariant Feature Transform (SIFT) algorithms. Electron tomography software *IMOD* generated simulations for volume reconstruction (tomograms) presented the acceptable results for both single and dual axis tomography. *3DMOD* software routines were used to perform the segmentation and the visualization of the reconstructed tomograms.

During the course of this thesis work it was observed that Affine Transform and SIFT based alignment techniques performs faster than both the manual seeding of the fiducials on individual tomographs and *IMOD* fiducial seeding. However lack of common feature points on all tomographs limits the SIFT algorithm to perform on a certain views only.

ACKNOWLEDGMENTS

I would like to present my deepen gratitude towards my main supervisors Arto Koistinen, Ph.D. and second supervisor Stefanos Georgiadis, Ph.D. for continuous, patient and valuable support during all the practical work and thesis writing.

I would also like to offer my gratitude to my Supervisor Arto Koistinen and Prof. Reijo Leppalainen, Department of Applied Physics, University of Eastern Finland for providing funding and handling all managerial issues for the this research work as well as arranging visits to the labs in Joensuu and Helsinki.

I am also thankful to Helena Vihinen (Senior Scientist) and Ilya Belevich (Postdoctoral Researcher) at Electron Microscopy Unit, University of Helsinki for their constant support and guidance during this research work.

In the end I would like to dedicate my work to my whole family, my friends and Suuskuu because without their support and encouragement I couldn’t have completed my thesis work in Finland. Above all I am thankful to Allah who provided me this convenience and strengthened me to finish my research work.

ABBREVIATIONS

ET = Electron Tomography

TEM = Transmission Electron Microscopy

SIFT = Scale Invariant Feature Transform

DoG = Difference of Gaussian

SIRT = Simultaneous Iterative Reconstruction Techniques

Contents

List of Figures

List of Tables

1 Electron Tomography: An Overview

Study of microscopic data has been proved very useful in the analysis of structural and organizational properties of different materials. Beyond the limitations of human eye visibility there exists an enormous world of microscopic and even smaller level information. This information about biological cell tissues or the particles of some metal reveals some very important facts for analysis. From the preparation of first ever samples using electron microscopy to the development and deployment of latest electron microscopes the study of ultra-thin structures opened the doors of a vast variety of research to scientists, engineers and biologists [1]. Experts in their related fields started benefiting from the images produced by electron microscopy but with the limitation of conventional microscopes and *2D* image information, the need of *3D* visualization of microscopic objects yet remained a big question. Over the past few years Electron Tomography (ET) has been in practice to overcome the *2D* limitations such as rotation and visualization from multiple views and also volumetric data acquisitions using the third dimension [1].

Transmission Electron Microscopy (TEM) has gone through various technological and technical changes such as numerous changes in hardware as well as in algorithms and fast computational software programs [2]. The high resolution ability of the latest microscopes helped producing the micro and nano scale images which has led to a vast variety of advancement in the area of biological and materials research and industry [3]. TEM was first developed and applied in biological research; however its rapid growth can be seen in other fields such as in materials sciences [4]. Many explorations using ET in the fields of life sciences, material sciences, electronic and polymer industry and educational research are on the record.

1.1 3D Tomography

3D Tomography is a versatile technique allowing researchers to study the structures of objects with nanometer resolution. In this technique a beam of electrons is passed through the specimen sample at incremental degrees of rotation around the center axis of the sample. This information is then acquired and collected to reconstruct the three dimensional image of the original sample. Fig 1-1 illustrates the mechanism of ET.

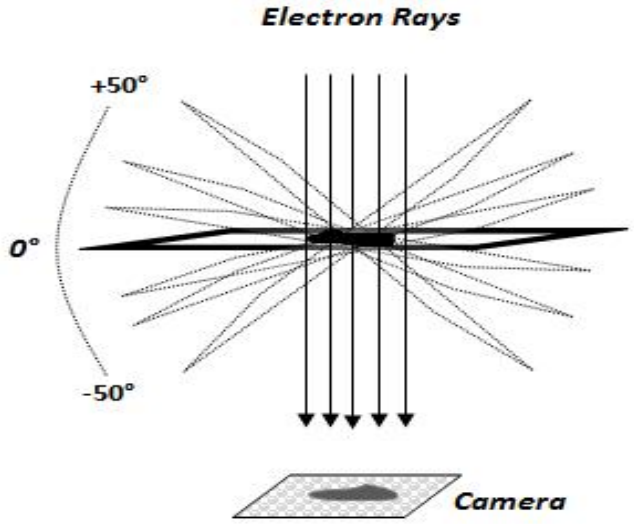

Fig. 1-1 Electron tomography mechanism [11]

Fig 1-2 shows the main principle behind taking the projections at tilt angles and then reconstructing the volume from these projections.

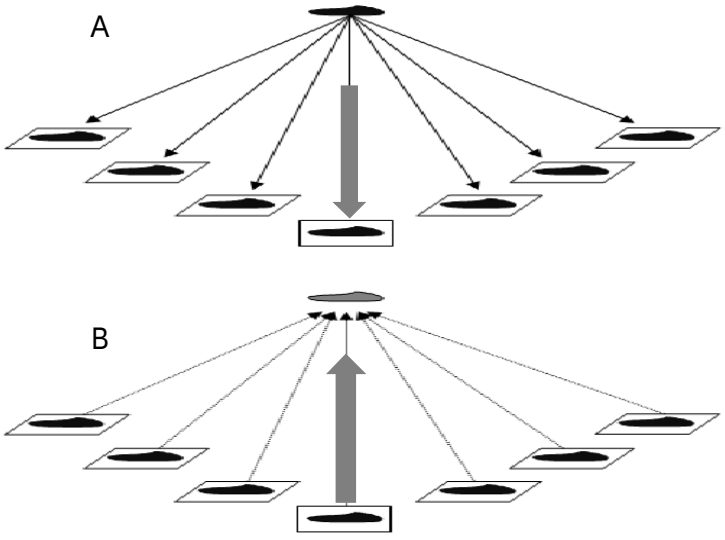

Fig. 1-2 A) Acquiring projections of specimen B) Volume reconstruction from the specimen projections [11]

1.2 Applications of 3D Tomography

With the development of high performance electron microscopes and *3D* visualization the study of tomography expanded its domain from life sciences, industry to material sciences and geophysics. *3D* visualization of nanometer scale objects made it possible for the researchers to find out the buried structures and other useful information i.e. in life and Clinical sciences ET is suitable to study biological structures with nanometer resolution. This makes it extraordinarily versatile, allowing the study of a large range of biological specimens, both in an isolated form and in their cellular context [5]. The continual shrinking of microelectronic devices has resulted in commercial products incorporating complex non-planar features with nano scale dimensions. *3D* imaging of such samples from IC devices is possible and that details such as buried defects and surface roughness can be visualized and *3D* metrology can be performed using ET [6]. In polymers nano-fillers have been assumed to be the most important determining factors of physical properties of the composites. Using *3D* tomography the presence and structure of these nano fillers can be determined with in base polymer's geometry [7]. Figure 1-3 illustrates the presence of silica particles in rubber matrix.

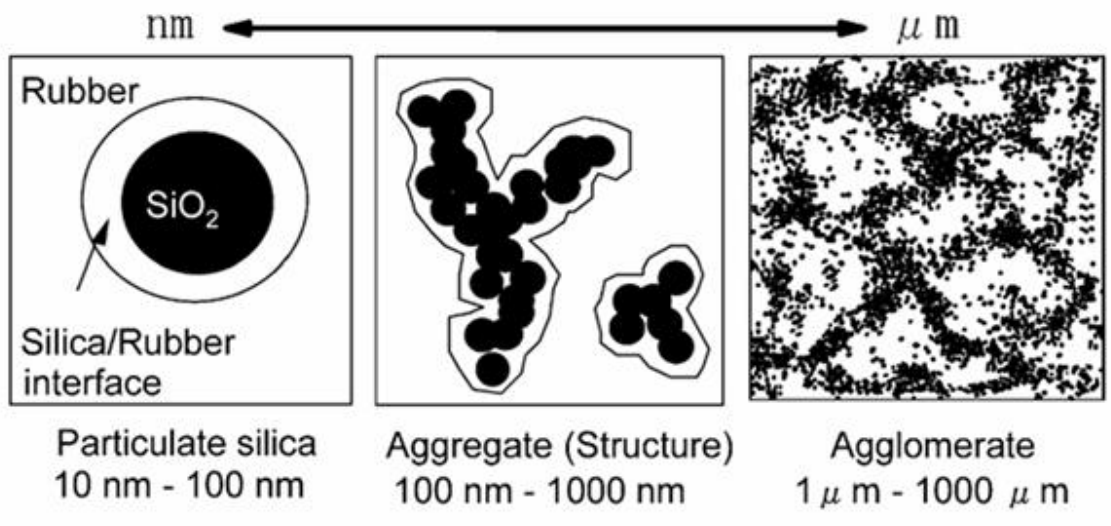

Fig. 1-3 Silica Particles and their morphology in natural rubber [7]

After the 3D reconstruction of the image stack produced by TEM the silica particles were visible in the form of aggregates as shown in the Figure 1-4.

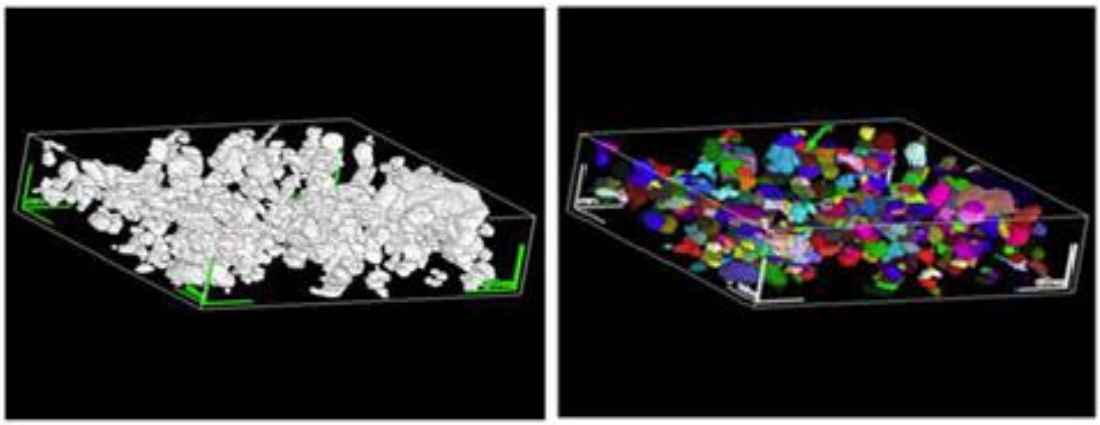

Fig. 1-4 3D Visualization of Silica Aggregate [7]

The shrinkage of microelectronic devices also leads to the study of marking defects in order to avoid production problems. Previous scanning microscopy techniques are limited because of *2D* projections only. On the other hand solid catalysts are of tremendous importance for economy and environment and the drive towards clean and efficient technology calls for precise design and characterization of catalysts. With the development of ET it has become possible to get a *3D* image of both the surface and the interior of the sample as well as structural information in three dimensions on *nano* scale resolution [8]. Figure 1-5 shows the reconstructed tomogram displaying the location of Au (catalyst) nano particles as their strong on-surface and inside metal support interaction.

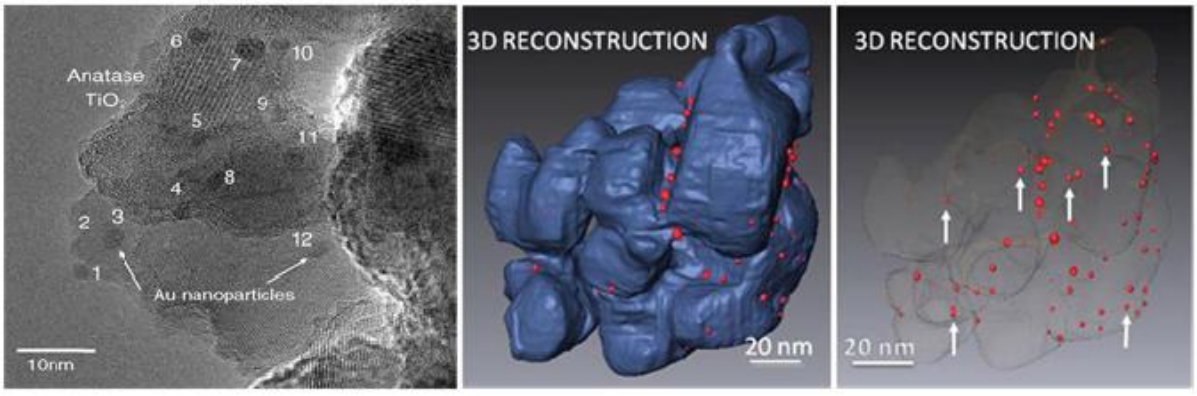

Fig. 1-5 Location of Au particles on TiO2 (left), 3D representation of TiO2 with the presence of Au nanoparticles on surface (center) and inside the metal (right) [8]

2 Principles of Electron Tomography

The principles of ET are similar as for most of the three dimensional imaging algorithms and techniques [9]. Compared to *2D* TEM from few decades ago, ET reveals detailed structural and organizational insight of the nano-scale biological and inorganic objects [10]. However the higher complexity of a *3D* architecture tends to provide less information in ET because of large data presence in a single *2D* image [10].

2.1 Background

In ET the set of *2D* image projections recorded are called tomographs and the reconstruction of these tomographs is called tomogram reconstruction. In this thesis work the terms 'tomograph(s)' and 'image(s)' are used interchangeably. In the process of tomogram generation, volumetric reconstruction must be preceded by the alignment of projections [11]. Alignment algorithms are usually preferred on one another depending upon the nature of the obtained tomographs. Alignment processes are discussed in more detail in the later sections of this chapter. A broader explanation behind the difference in nature is whether gold nano particles were used to help in the alignment during the microscopy process.

The most used *3D* reconstruction methods are algebraic such as in real space or in the Fourier space [11]. Since in chapter 4 of this thesis work it is shown that the tomograms are reconstructed using Fourier methods therefore the discussion of reconstruction methods in this chapter is limited to Fourier space algorithms only.

The definition of the Fourier space theorem explains *2D* image of the tilt series as a correspondent central section in the Fourier transform of the original imaged specimen [10]. i.e. Tilt series acquisition on a tilt angular range is the same as scanning the sample's information in Fourier domain.

2.1.1 Resolution and Parametric Differences

In ET it's important to understand the relationships among specimen thickness, tilt angle range(s) and the attainable resolution. Frank [11] explained the following rule relating all the aforementioned parameters. The relationship between the attainable resolution r, specimen thickness T and the number of projections N is

$$r = \pi T / N \qquad (2.1)$$

Therefore, for a sample of original thickness 200*nm* the attainable resolution of the 101 projections (tomographic data recorded at -50° to +50°) will be ~ 6*nm*.

Equation 2.1 can be used to find the estimate of other involved parameters. i.e. Number of projections or tilt increment on a desired tilt range.

Therefore for an attainable resolution of 5*nm* from original sample of thickness 100*nm*, Equation 2.1 gives 63 projections with an increment of 2°. The above relationship is valid for a limited tilt axis if the specimen is not cylindrical or spherical. For spherical or cylindrical specimen full resolution can be achieved by tilting the sample at -90° to +90°. Frank [11] explained that in practice however this is not achievable due to two main reasons.

1. On high tilts (more than 65°-70°) the specimen holder's design causes mechanical constraints while recording projections
2. At high tilts only few electrons will pass though the specimen thus will provide very less useful contribution of structural data to the projected images

2.1.2 Missing Wedge

The missing information on higher tilt angles results in the missing data on the *z* axis in the shape of a wedge and hence referred by many authors as missing wedge as shown in Figure 2-1.

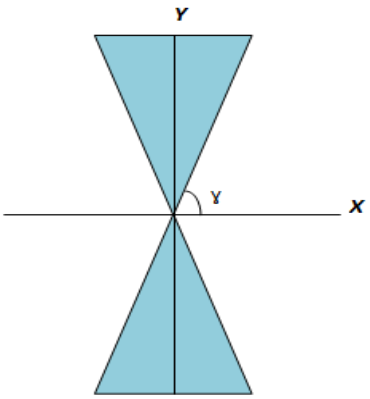

Fig. 2-1 Missing wedge [11]

Frank [11] described an elongation factor γ_{max} in the *z* axis as following.

$$r = SQRT\{\frac{\gamma_{maximum} + Sin\gamma_{maximum} Cos\gamma_{maximum}}{\gamma_{maximum} - Sin\gamma_{maximum} Cos\gamma_{maximum}}\} \quad (2.2)$$

For a tilt range of -50° to + 50° and the maximum tilt angle 50° the elongation factor will be ~ 1.81, which then changes the previous attainable resolution from 6*nm* to 10.8*nm*. These effects because of missing wedge lead to the development of dual axis tomography [11].

2.2 Alignment

Precise and correct alignment for *2D* projections is very important towards the *3D* tomographic reconstruction. Ideally all the projections should be aligned according to the projection angle used in the image acquisition from original sample. Poor image alignment will result in the blurring of features in the volumetric reconstruction [12]. There are many factors involved behind the difficulty in image alignment. One of the main reasons which make it more difficult is the exposure of electron beam which causes geometric changes in many samples [11]. Also the range of the rotational axis of the object holder is limited due to mechanical limitations [11]. Alignment of the projections also includes rotational and translational alignment [13]. In the attempts towards solving the problem of alignment, finding the initial correspondence in the consecutive images is also important before proceeding to the next phases of tomographic reconstruction [14]. Misalignment of the projections may also be the result of the fact that the tilt axis of the tomographic data isn't orthogonal to the beam direction [15], [16].

Considering all the above mentioned factors behind the unaligned tomography data, the accurate and fine alignment with respect to a common point [17], is necessary in the successful generation of tomograms and volumetric reconstruction.

Different algorithms and techniques are being used to deal with the image alignment prior to tomographic reconstruction. The two most common techniques used for the alignment methods are 1) Using fiducials as seeds 1) and Markerless alignment [18].

2.2.1 Alignment with Fiducial Markers

Solutions to the alignment problem either include or exclude fiducials. Most frequently the fiducial alignment is done via gold nanoparticles [14], [15], [19], [20] that are introduced to the specimen during preparation. Typically these gold particles are present on all the

projections and it's easier to locate them because of their round shape geometry. Frank [12] describes the usefulness of gold nanoparticles because of their spherical shape geometry and due to the high contrast to the background.

Ideally a single projection may contain more than 15 to 25 fiducial markers. Marking an average of 20 particles for all the respective projections yields good results in tomographic reconstruction. The projections are aligned by shifting the images by the difference in the locations of the fiducials.

2.2.2 Markerless Alignment

In contrast to the more adapted technique of using gold nanoparticles in alignment the markerless alignment is preferred when there is a danger of gold particles interfering in reconstruction [21] Another reason for not using gold particles is because of the objects which are freely supported [11]. Some of the highlighted algorithms in literature [11], [21] are the use of common line and cross correlation using geometric shifts in projections with respective angles. According to Frank [21] the principle of cross-correlation alignment can be expressed in terms of discrete *2D* cross correlation function as

$$\boldsymbol{h(m,n)} = \frac{\boldsymbol{1}}{\boldsymbol{MN}} \sum_{\boldsymbol{j=0}}^{\boldsymbol{M-1}} \sum_{\boldsymbol{k=0}}^{\boldsymbol{N-1}} \boldsymbol{f(j,k)g(j+m,k+n)} \tag{2.3}$$

Where *f* and *g* are the optical density measurement of the images while *M* and *N* are respectively the width and height of the images. If *f* and *g* are similar i.e. almost similar views then the value of *h* will be higher on those locations. Here the cross-correlation principle means finding all such locations for every two tomographs. Then using these locations i.e. coordinates points, the relative shift are calculated. Once all the shifts are found for the whole image stack, the image set can be aligned using those shifts.

In many cases the cross correlation technique may not be completely successful and hence different further algorithms can be used to improve the alignment results. One such technique proposed and implemented in this thesis work is the manual selection of common feature points. In the features selection process, common points are found on all the images and then the unaligned images are moved with respect to the location of the points in the central image. There are different ways to use this technique but the simplest method is to compare an image, pair wise with the central image and repeating the procedure for the whole image stack. Since the technique of using feature points is completely manual therefore, the emphasis is only on finding the least error in the alignment.

This error value will then be compared with the more automatic technique of using Scale Invariant Feature Transform (SIFT) algorithm.

2.2.3 Scale-Invariant Feature Transform (SIFT)

Scale-invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. This technique can be used for extracting distinctive features from different views of an image. The extracted features from SIFT algorithm are highly distinctive and a single feature vector can be used to match features in a large database of features from other views of the same image [22]. In tomography the different views of an image sample are not scale variant rather they only differ from each other with respect to the tilt angle and shrinkage or stretching along *z*-axis. In the current study the SIFT algorithm is used in the following pattern in order to compute an alignment feature.

Scale-space extrema detection

Lowe [22] recommends Gaussian function as the only possible scale space detector. Hence for an image, the scale space function can be written as $M(x, y, \sigma)$ which can be obtained from the convolution of variable scale Gaussian $G(x, y, \sigma)$ with an image $I(x, y)$. Where x and y are the coordinates of the image and σ is the relative angle.

$$M(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.4}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)} e^{-(x^2+y^2)/2\sigma^2} \tag{2.5}$$

Difference of Gaussian is used to detect stable key point locations in the scale space of an image view [22]. Difference of Gaussian $DoG(x, y, \sigma)$ is computed as

$$DoG(x, y, \sigma) = \{ G(x, y, l\sigma) - G(x, y, \sigma) \} * I(x, y) \tag{2.6}$$

Where l is a constant multiplicative factor differentiating two adjacent scales. Equation 2.6 can be written as

$$DoG(x, y, \sigma) = M(x, y, k\sigma) - M(x, y, \sigma) \tag{2.7}$$

Key point localization and relative angle orientation

Once a key point is located by the comparison of its pixels to its neighbors, next step is to fit this interpolation to the nearby data for location and scale.

These key points are then selected based on their stability and angle of orientation.

Extracting descriptor and alignment

Followed by the assignment of location, scale and orientation to every key point the descriptor can be extracted by computing the gradient magnitude and orientation at each image [22].

2.3 Tomogram Reconstruction

For the Tomogram reconstruction the weighted backprojection algorithm is used. The backprojection algorithm implements a two dimensional reconstruction of the density $\rho\ (x,y)$ from the specimen projections [23]. In polar coordinates the density is represented as $\rho(r, \varphi)$ as shown in Figure 2-2 A. Conventionally for the backprojection algorithm the projections are spaced by angular range of $2\pi/N$. Where N is the number of Projections. However due to the TEM limitations the max angle is limited to 65°-70°. For example in this study for the second sample there are 121 projections over the range -60° to +60°. Hence the projections are spaced at $4\pi/3\ N$ with 1° increment. Figure 2-3B shows the Fourier space coordinates for the density.

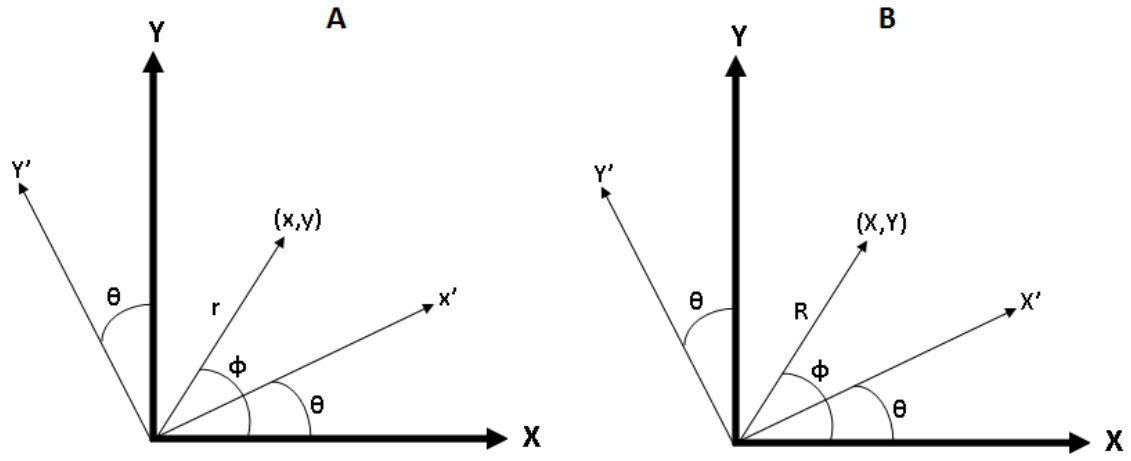

Fig. 2-2 A) Real space coordinates B) Fourier space coordinates [24]

While recording the TEM *2D* projections it is easier to understand that these projections actually intersect on a common line (*z-axis*) along the electron beam. The spacing between the consecutive projections as illustrated above is measured at the incremental tilt angles. Hence the *3D* volumetric reconstruction of these projections can easily be restricted to a number of sequential *2D* reconstructions which are orthogonal to the projection axis.

In the polar coordinate system parallel to the optical axis (*x-axis*) the projected densities can be written as $\rho\theta\ [x'=rcos\ (\Phi-\theta)]$. The following derivation as derived by Gilbert [24] explains the step by step *R* weighted backprojection implementation in *IMOD.*

As seen above the density distribution of the projections is $\rho\ (x, y)$. Then the *2D* Fourier transform of this density is

$$F(k_x, k_y) = \int\int \rho(x,y)\exp[-i2\pi(k_x x, k_y y)]\, d_x d_y \tag{2.8}$$

From this the *1D* density $\rho\theta(x')$ can be deduced as the projected density along the *y* axis on *x* axis [24].

$$\rho_\theta(x') = \sum_{-\infty}^{\infty} \rho(x,y) d_\theta' \tag{2.9}$$

According to the Fourier slice theorem, the Fourier transform of a particular projection $\rho\theta\ (x')$ is a central slice of the Fourier transform of the density distribution at an angle θ along the *x-axis*.

i .e.

$$F[\rho_\theta\{x' = rcos(\Phi-\theta)\}] = F(X,Y)\delta(Y'_\theta) \tag{2.10}$$

Let's assume that

$$\rho'_\theta = \rho_\theta[x' = rcos(\Phi-\theta)] \tag{2.11}$$

Then equation 2.10 becomes.

$$F(\rho'_\theta) = F(X,Y)\delta(Y'_\theta) \tag{2.12}$$

The right hand side on the above equation shows the line section of the *2D* Fourier transform of the original density $\rho\ (x, y)$. Taking the inverse Fourier transform of the equation 2.12.

$$\rho'_\theta = F^{-1}[F(X,Y)\delta(Y'_\theta)] \tag{2.13}$$

By summing up for all the projection angles yields

$$\sum_\theta \rho'_\theta = F^{-1}[\sum_\theta F(X,Y)\delta(Y'_\theta) \tag{2.14}$$

Introducing a sampling function as described by Gilbert [24] to the right hand side of equation 2.14.

$$S(R,\Phi) = \delta(X\cos\Phi + Y\sin\Phi - R) \tag{2.15}$$

$$\sum_{\theta} F(X,Y)\delta(Y'_{\theta}) = F(X,Y)S(R,\Phi) + (N-1)F(0,0) \tag{2.16}$$

Where *S(R, Φ)* is the dirac delta function of unit weight at angles *4π/3N*, where *N* is *1, 2 … N* and at the origin where *R* (the reciprocal radius in Fourier space) is zero. The value of *S(R, Φ)* is zero when *R* reaches beyond the maximum radius in the Fourier space.

In Equation 2.16 the term *(N-1) F(0,0)* appears because of the Fourier transform values of *N* projections contribute at the origin *x*=0, *y*=0 whereas the term *F(X,Y) S(R,Φ)* is the complete Fourier transform of the original sample projected along the line sections at the angular range *4πn/3 N, n=1,2…N* with unit weight of the dirac delta function at the origin. Now taking the Inverse Fourier transform of equation 2.16 and taking convolution with the density *ρ (x, y).*

$$F^{-1}[\sum_{\theta} F(X,Y)\delta(Y'_{\theta})] = \rho(x,y) * F^{-1}[S(R,\Phi)] + F^{-1}(N-1)F^{-1}[F(0,0)] \tag{2.17}$$

The direct back projection method uses the above definition for the reconstruction purposes however there lies one problem. The dirac delta function assigns a unit weight to the sampling function *S(R,Φ)* at all the locations in the Fourier space and the sampling function should expand in the expanding radial direction in the Fourier space therefore it must be increased according to the value of *R* in the Fourier space for a precise reconstruction [24]. Introducing another function *Gθ* which is defined as

$$G_{\theta}(x') = \frac{1}{2\pi^2} * \frac{\partial\, \rho_{\theta}(x')}{\partial x'} * \frac{1}{x'} \tag{2.18}$$

Where *1/x´* is the reciprocal radius in the Fourier space. The Fourier transform of the function *Gθ* is equal to the Fourier transform of the projections times the Fourier space radius value *R*. According to the central slice theorem

$$\sum_{\theta} G_{\theta}[x' = r\cos(\Phi-\theta) = F^{-1}\sum_{\theta} RF(X,Y)\delta(Y'_{\theta}) \tag{2.19}$$

Let's assume that

$$G'_{\theta} = G_{\theta}[x' = r\cos(\Phi-\theta)] \tag{2.20}$$

Equation 2.19 then becomes

$$\sum_{\theta} G'_{\theta} = F^{-1} \sum_{\theta} \mathrm{RF}(X, Y)\delta(Y'_{\theta}) \quad (2.21)$$

Introducing the sampling function *S(R,Φ)* to the right hand side of the above equation and by applying the same argument which was used in the derivation of the equation 2.17 and denoting the reconstructed density as *ρrec* yields

$$\rho_{\mathrm{rec}}(x, y) = \rho(x, y) * F^{-1}[\mathrm{RS}(R, \Phi)] \quad (2.22)$$

Hence the reconstructed density value is convoluted with the inverse transform of the sampling function weighted by the radial value *R* in the Fourier space for an accurate reconstruction [24].

The reconstructed tomograms were later segmented and modelled using *IMOD* software as explained in chapter 4. *IMOD* routines perform relatively better than other imaging software. Figure 2-3 presents the flow chart of the practical steps carried out in this thesis work which is explained in detail in chapter 4.

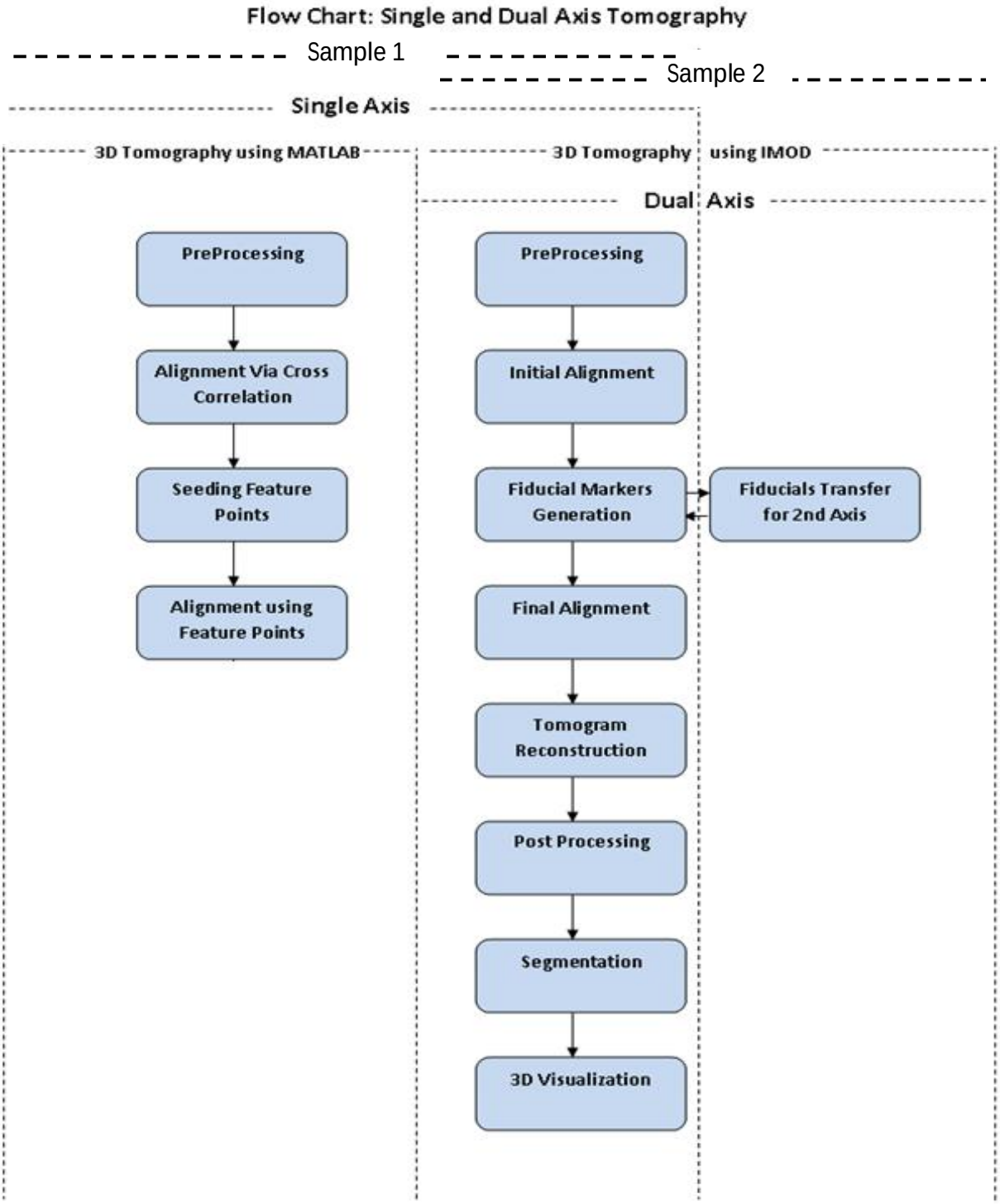

Fig. 2-3 Flow Chart summary of the electron tomography processes for sample 1 and 2

3 Aim of the study

The main purpose of this research study are to carry out research and implementation of electron tomography steps using two different sets of tomographs obtained from biological samples after electron microscopy. Primarily this study compares some of the previously employed alignment techniques with the proposed method of aligning tomographs using Affine transform and SIFT. In addition to the comparison of different alignment techniques, *IMOD* software was used to elaborate the process of tomogram reconstruction, segmentation as well as *3D* modelling and visualization. The aims of the study are

- Background study of the Electron Tomography processes
- Implementation and analysis of alignment techniques
- Volumetric reconstruction of the aligned tomography images

The basic problem in the alignment of microscopy images can be dealt with the proper implementation of image registration techniques. As discussed in the previous chapters, the acquired image stack after electron microscopy contains images which differ from each other in rotation, translation and scaling. Affine transform was used at the primary tool for the preprocessing as well as initial alignment of the image stacks.

SIFT algorithm used in this study is implemented recursively to locate the common feature points on the registered images using Affine transform. Due to the robust nature of SIFT matching descriptors, common points were marked and located in pair wise opposite views of the image stack. SIFT descriptors containing both the feature point attributes and locations of the feature points were used as the only image registration parameters during the course of aligning the image stack. Both Affine and SIFT algorithms were developed in *MATLAB* (see Appendix 3) and image stack data was tested for the alignment results.

IMOD software was also used for fiducial based alignment for both single and dual axis tomography. *IMOD* routines were used to perform the tomogram reconstruction and segregation followed by *3D* visualization.

4 Methods and Implementation

In this chapter the implementation of the ET process is discussed i.e. from image acquisitions and preprocessing to the final segmentation and visualization as described in Chapter 1. The methods and implementation are performed mainly on two different data sets. Figure 4-1 and Figure 4-2 show the original raw data for both sample acquired at different tilt series. The description of both the data sets is as follows.

1. Sample 1 (Tomographs without fiducials markers) – Oral mucosa
   i. 101 images of the size 1336 K x 952 8 bit TIFF images
   ii. Images taken at -50° to +50°
   iii. Tilt angle increment 1°
   iv. Pixel size 8.5 *nm*

2. Sample 2 (Tomographs with gold fiducial markers) - Mitochondria
   i. 121 x 2 images of the size 2K x 2 K, 16 bit TIFF images
   ii. Images taken at -60° to +60° for both axis
   iii. Tilt angle increment 1°
   iv. Fiducial markers size 10 *nm*
   v. Pixel size 2.28 *nm*

The system and software programs specifications are as follows.

i. a) Intel *Core i5* , Two 2.5 GHz processors and b) Intel *Core2 Duo* 2.0 *GHz &* 8 *GB* RAM
ii. *MATLAB* 2012(a) on UEF server (Mathworks Inc., Natick, Massachusetts, USA)
iii. *IMOD* 4.7.7 (University of Colorado, Dept. of MCD Biology, 347 UCB, Boulder, CO 80309, USA)
iv. *VideoMach* developed by Gromada.com

In the first half of this chapter the process of markerless alignment with Feature point selection and SIFT based alignment is discussed while in the lateral part *IMOD* routines are used to perform the tomography steps. Results and observations are discussed in chapter 5.

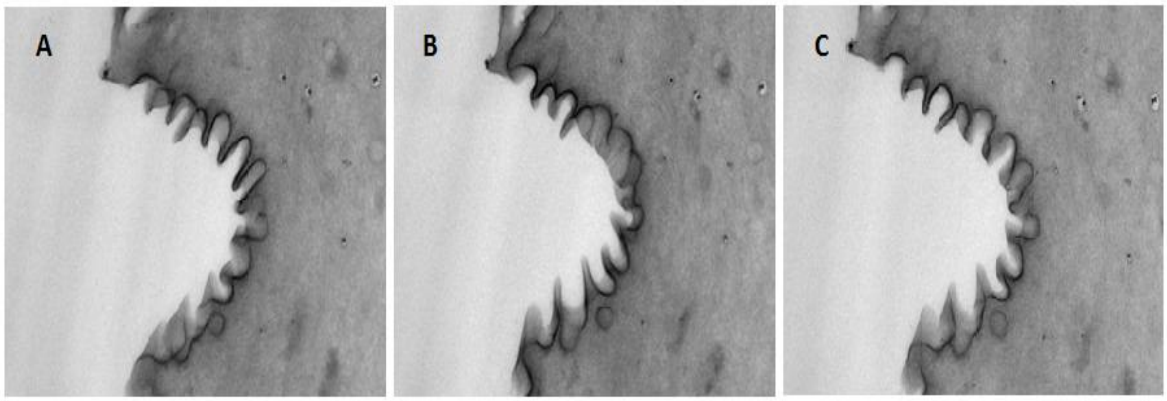

Fig. 4-1 Sample 1 obtained at angles A) -50° B) 0° C) +50°

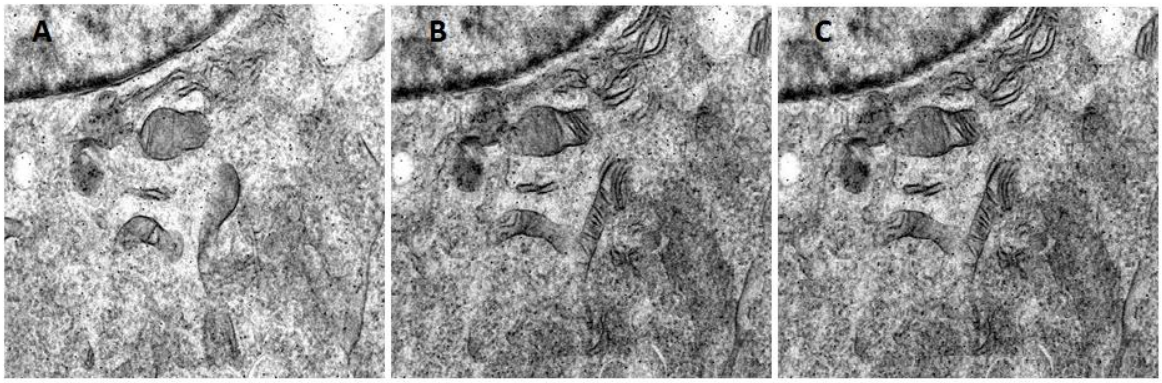

Fig. 4-2 Sample 2 obtained at angles A) -60° B) 0° C) +60°

4.1 Markerless Alignment and Feature Points

4.1.1 Preprocessing

In this section the implementation of Feature point algorithm of aligning tomographs without fiducial markers is explained. Images when taken at tilt angles differ from one another by a few geometric transformations as shown in Figure 4-3. These transformations can be shifting on the horizontal and vertical axis and also rotational axis. Therefore a geometric transformation method 'Affine Transform' employed to find the shifted parameters in the translation, rotation and skewness. The Affine transformation model can be used on the registered images to perform the image registration [25]. In image matching and image registration Affine models and Affine transforms are widely used and preferred over other standard techniques because of their robust estimation of displacements in the image pixels [26].

According to the affine transformation theory for an image pair containing points *I(x,y)* and *I´(x´,y´)* the affine relation between these two points is given by

$$I' = MI \tag{4.1}$$

M is the affine transformation matrix whose elements are in the first Row: *m11 m12 m13*, second Row: *m21 m22 m23* and in third Row: *0 0 1*. For *N* pair of corresponding points in the pair of the images the solution of the above equation can be written as

$$x_j' = m_{11}, x_j + m_{12}, y_j + m_{13} \tag{4.2}$$

$$y_j' = m_{21}, x_j + m_{22}, y_j + m_{23} \tag{4.3}$$

for j=1,...N

After interchanging the sides the above set of equations can be written in the generalized form as

$$\mathrm{b} = Ka \tag{4.4}$$

or

$$Ka = b \tag{4.5}$$

The above equation can be solved using least square fit and for that taking the transpose on both sides yields

$$K^T\mathrm{K}a = K^T\mathrm{b} \tag{4.6}$$

$$a = (K^T\mathrm{K})^{-1}K^T\mathrm{b} \tag{4.7}$$

Here the term $(K^TK)^{-1}K^T$ is the pseudo inverse of scalar *K*. MATLAB provides the solution of problem in Equation 4.7. Affine transformation was performed on the images to reduce the attainable values of offsets in the image alignment process. Images are transformed in such a way that the central image lies usually on the zero tilt angle. The tilted images are then transformed one by one with respect to the central image and the transform values are recorded. Let X_i as the central image then by first moving in the forward direction with the indexed images as X_{i+1}, X_{i+2}…X_L and repeating the same process in the backward direction for images X_{i-1}, X_{i-2}, $X_{M,}$, the comparison of

the pairwise images is completed. Where i is the tilt angle, L and M are the total number of projections in each direction only divided by 2.

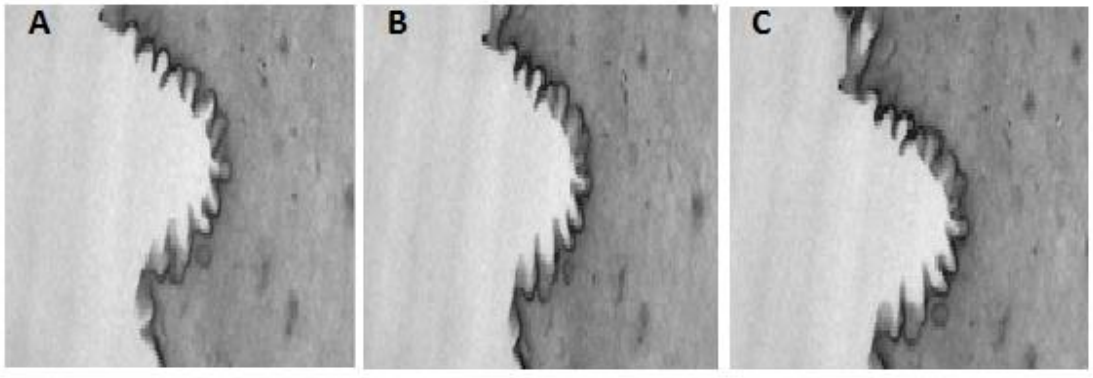

Fig. 4-3 Affine transformation processes. A) Center image B) Tilted image C) Tilted image transformed

It is important to mention here that the transformed images are not the ones processed in the alignment process rather these images will be used only to calculate the transform offset in the tilt images with respect to the central image. Using these offset values along the tilt axis (*x axis*) original image is moved closer to the central image X_i. Figure 4-4 shows the process of affine transformation and moving the original images using offsets.

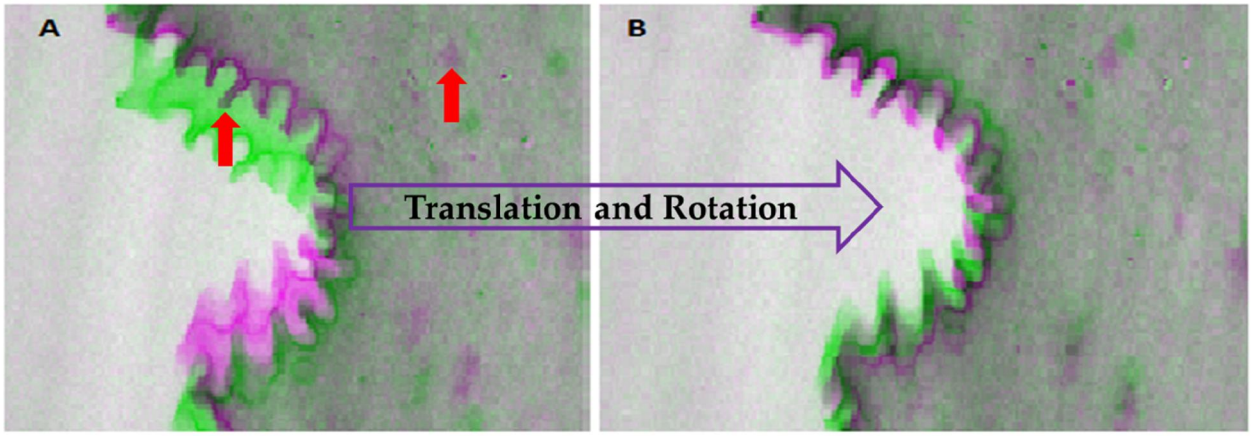

Fig. 4-4 A) Center image and tilted image before Affine transformation B) Images after Affine transformation

After the affine transform and moving the images towards the center with respect to the central image its now easier to find maximum correlation in the alignment process.

4.1.2 Alignment

Image transformation is followed by the implementation of Cross Correlation algorithm in the Fourier domain as presented by Frank [21]. In TEM, the images are taken at tilt angles and therefore when moving forward or backward from the central image, the tilted projections are shrinked along the axis of electron beam. Preprocessing using the affine transform stretches the images with respect to the central image and hence produces images in more registered form then original form. This algorithm implemented in MATLAB (see Appendix 1) then computes the cosine stretching of the image pairs with the ratio of the cosine of the angles between them. Since this method attempts to find the cross correlation between images as pairs i.e. pair wise cross correlation of the tilted image with the central image therefore the cosine stretching formula becomes

$$x = \cos\alpha / \cos\beta \tag{4.8}$$

Where *Cosα* is 0 and *Cos* 0 is 1 so the stretching factor *X* for respective image pair therefore becomes *1/ Cosβ*. The cross correlation is implemented in the Fourier space. Figure 4-5 shows the graph of the Cross Correlation peak for the two compared images in sample 1 where the maximum correlation was found. The pixel values offset in the neighborhood of the peak are computed to find the matching areas in both images.

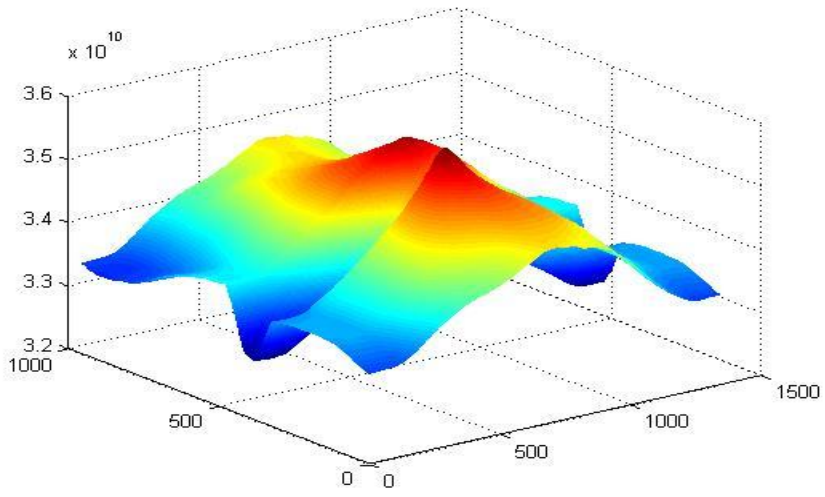

Fig. 4-5 Cross correlation graph and peak in dark red color for sample 1

Once the offset values of the pixel points in both images are found, the moving image is transformed with respect to those offset values. In this way all the images were moved with respect

to the central image. The final transformed images were then stacked together to see the alignment progress. At this point, the markerless alignment for the image data did not quite produced accurate results. While moving from negative to positive axis the distortion in the *y* axis causes the image stack to shutter.

4.1.3 Fine Alignment using Feature Points

Preceded by the markerless alignment and the inspection of the results generated in the cross correlation process, feature points are selected to further improve the alignment of tomograms. For this purpose the tomograms were checked for the occurrence of common features (points). Once the inspection of the points seemed promising, the searched points in every image were selected manually using a MATLAB written routine (see Appendix 2). For 101 images a total of 505 points were selected i.e. 5 points on each image. Figure 4-6 shows the locations of the selected points on all the images.

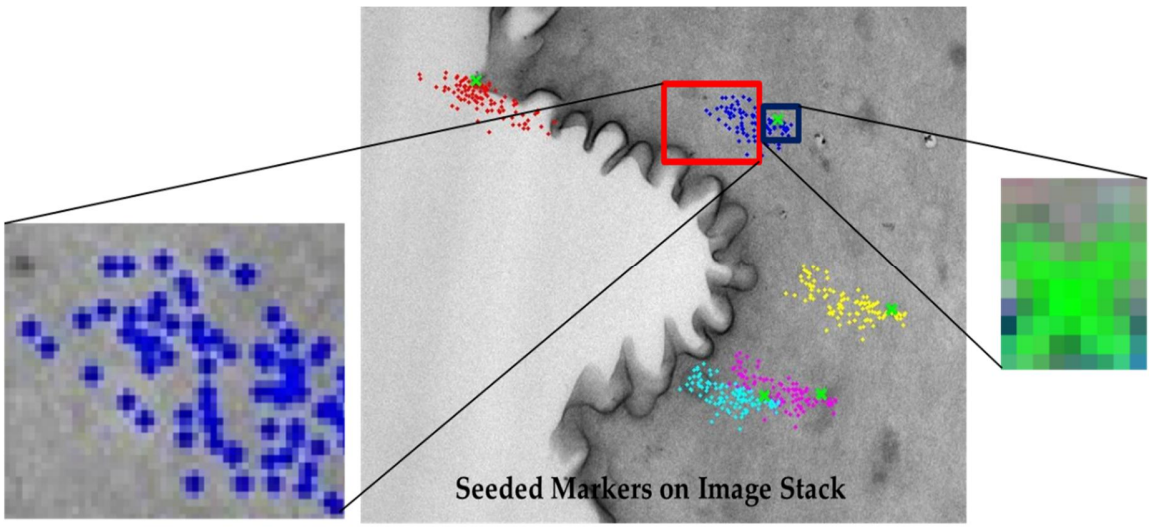

Fig. 4-6 Feature points distribution on the center image of sample 1

The central image was set as the index image and the relevant distances of the index image points were calculated from the location of the same points in the images on either angular direction. The distances were calculated using the Euclidian distance formula as given by

$$D = SQRT\{(x - x_i)^2 + (y - y_i)^2\} \tag{4.9}$$

Where x and y are the coordinates of a point in the central image and x_i and y_i are the points in the tilted images. Once the distances are found, images were compared pair wise i.e. Index image with the tilted images one by one. Finally all the images were moved according to the central image points using the mean of the displacement of all the five points in both images. The results were then checked visually by importing the images in *Videomach* software and playing the video output file. As compare to the resultant video in the markerless alignment process, the new video file looked error free and there was very less distortion found in the movie frames.

4.1.4 SIFT based Alignment

SIFT based alignment algorithm (written in *MATLAB* and *C* language by Lowe [22], (see Appendix 3) runs on individual images and computes the SIFT features for every image. SIFT returns the feature points as seeds based on detection of common points, their matching and the tracking while tomographs are compared pairwise [27]. Figure 4-7 shows the results of individual points marked by the algorithm. Once the points are marked on the individual images, the matches in the images are found pair wise. Every image was compared with the center image and matching point attributes were computed as shown in Figure 4-8 and Figure 4-9. Based on the matching locations the matching points in both images were recorded.

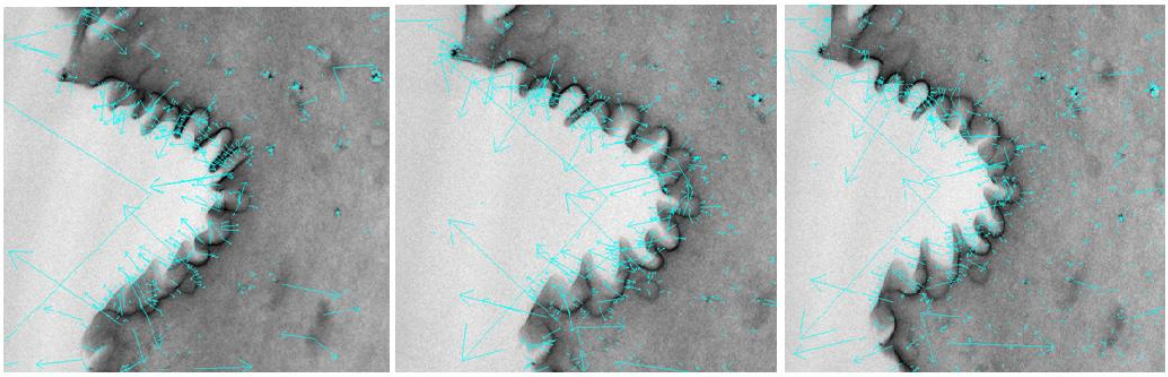

Fig. 4-7 SIFT points located on three different images

This resulted in a very large matrix containing the matches between the center image and all remaining images on both directions of the tilt angles. For finding common features to compute the relative offset distances in the tilt images with respect to the center image, the count for the matches were searched in the center image.

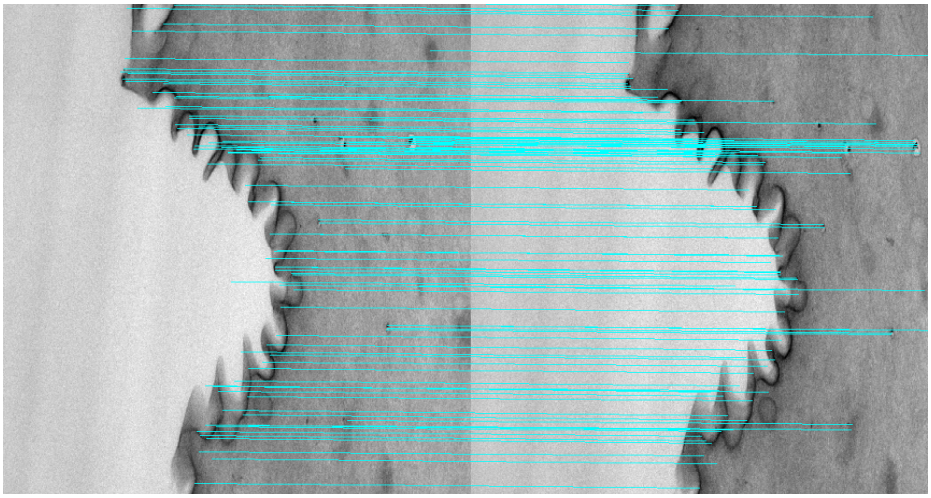

Fig. 4-8 SIFT matching between images with less tilt angle among them

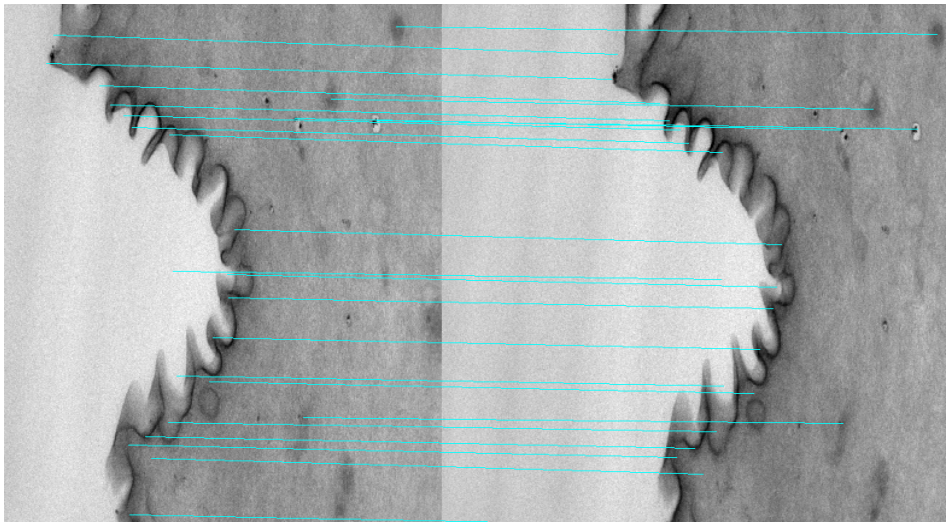

Fig. 4-9 SIFT matching of the images with higher tilt angle difference

Only those points were selected from the center images which occurred most frequently in the matching phase. Figure 4-10 shows the common feature points used for the alignment of the image stack using SIFT. Once the common points are found, the tilt images were aligned using those features with respect to the center image. During the matching process as the algorithm searches for features points in the far distant images from the center image, the number of common points reduces due to the shrinking of the objects in the images. Figure 4-11 shows the graphs of the common points found with respect to the increasing tilt angles.

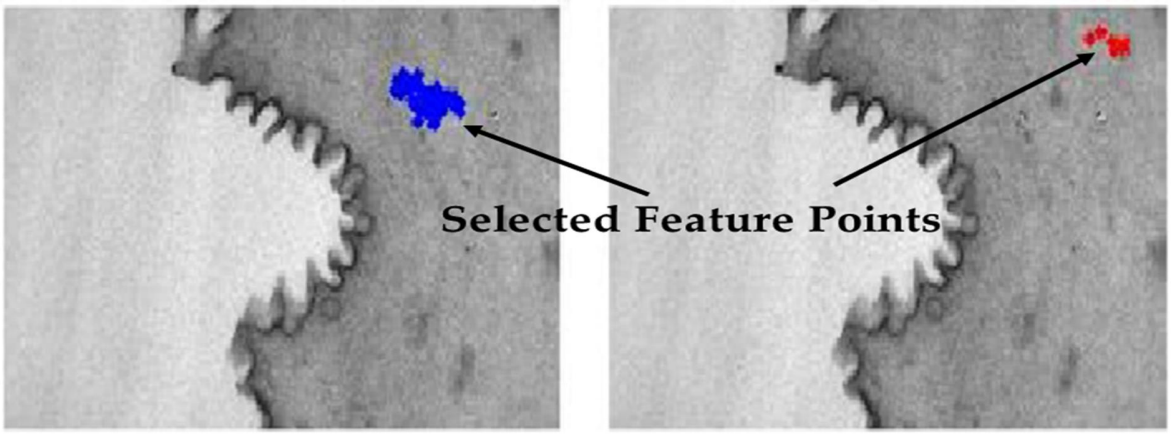

Fig. 4-10 Feature points selected for the SIFT based alignment

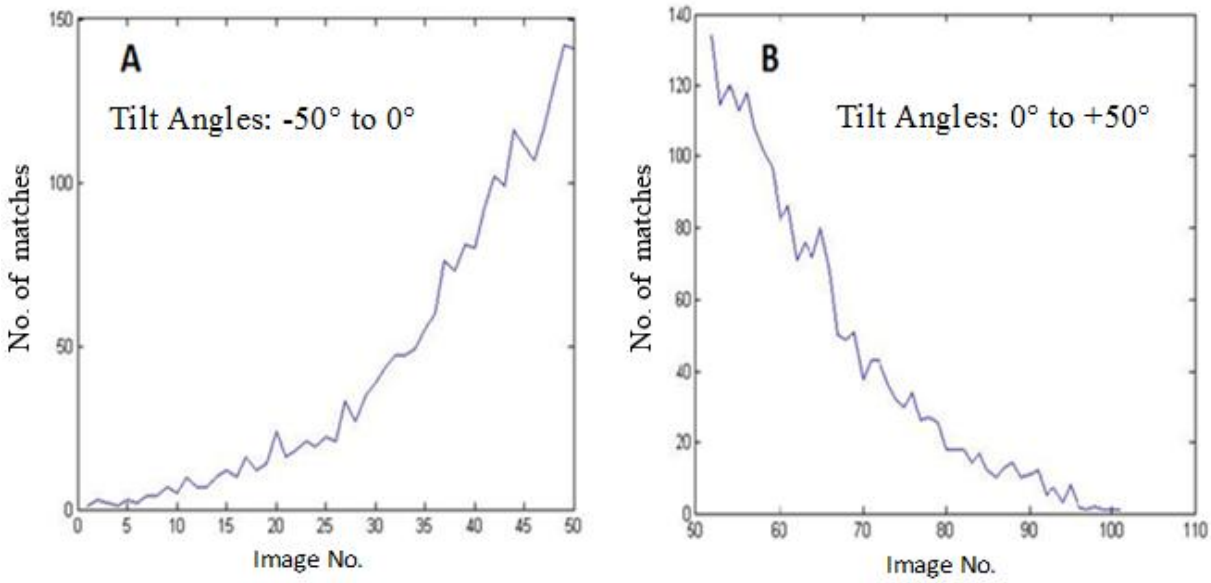

Fig. 4-11 A) No. of matches on negative tilt axis B) No. of matches on positive tilt axis

4.2 Volume Reconstruction with Alignment via Fiducials

4.2.1 Preprocessing

Prior to the alignment via Fiducials in *IMOD*, both samples were loaded in *IMOD* and in the preprocessing phase the *x-rays* were eliminated using IMOD routine *ccderaser*. Originally samples contained some extreme values due to x-rays presence and therefore these values were truncated.

IMOD routine *ccderaser* then searched through the entire stack of the images for extreme intensity values. *ccderaser* removed the extreme intensity values using the algorithm of dividing the image into patches, calculating the mean and standard deviation of the pixel values in the respective patches and then the patches were searched for pixels with extreme values and finally the pixels with values higher than a multiple of standard deviation were replaced with the mean value of the pixels 8 neighborhood.

4.2.2 Alignment

For initial alignment of the preprocessed stack, the *IMOD* routine *Xcorr* was used and images were coarse aligned using translation and rotation followed by cross correlation. Prior to the alignment the images were stretched with respect to the central image and perpendicular to the tilt axis. The stretching factor is the ratio of the cosines of the tilt angles of the respective images. After the stretching the function α calculates the cross correlation in the *IMOD* routine *Xcorr* and the peaks of correlation matrix were examined to find the shifts in the tilted images. This step performs the alignment without taking the notice of fiducial markers present on the sample data. Once the images were aligned the fiducials models were generated for further alignment.

4.2.3 Fiducial Model Generation and Fine Alignment

After the initial alignment via cross correlation, the fiducial markers models were generated in IMOD. Sample 1 images did not contain any gold particles therefore some marks were manually found on the images to be seeded as fiducials. The fiducial models were created on the central image (0° tilt) and roughly 10 to 15 particles were selected as seeds. Sample 2 already contained some gold particles and fiducials were seeded as 25 particles per image. While seeding fiducials opposite surfaces were checked for the exact seeding. Figure 4-12 shows the seeding of fiducials on both sample images. Figure 4-13 shows the model view of the seeds and their presence on the opposite surfaces.

After seeding the fiducials on the images, the tracking process was used in *IMOD* to track the fidcuials on all the surfaces. *IMOD* routine *Bead Fixer* was used in this step. *Bead Fixer* runs iteratively to check for the residual errors due to the gold particle missing or not centered on the individual images.

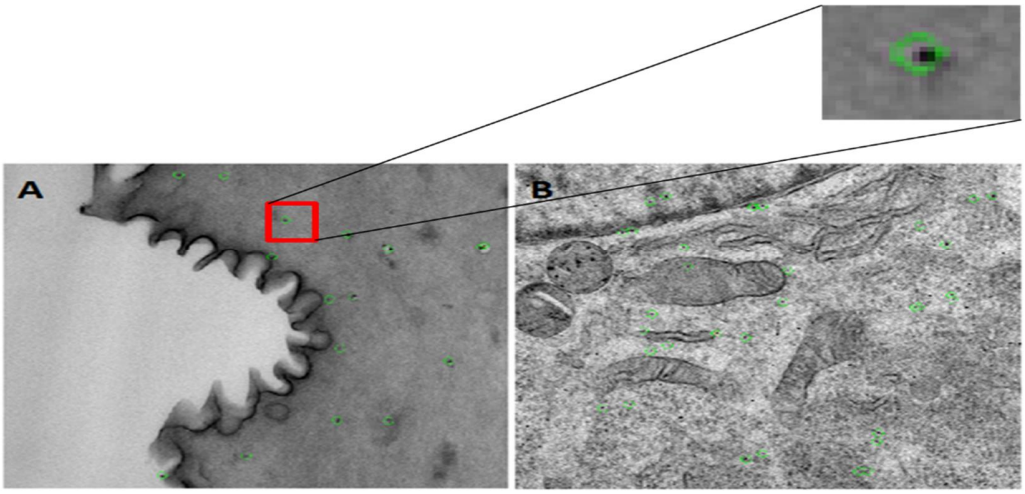

Fig. 4-12 Fiducial markers as seeds in green color

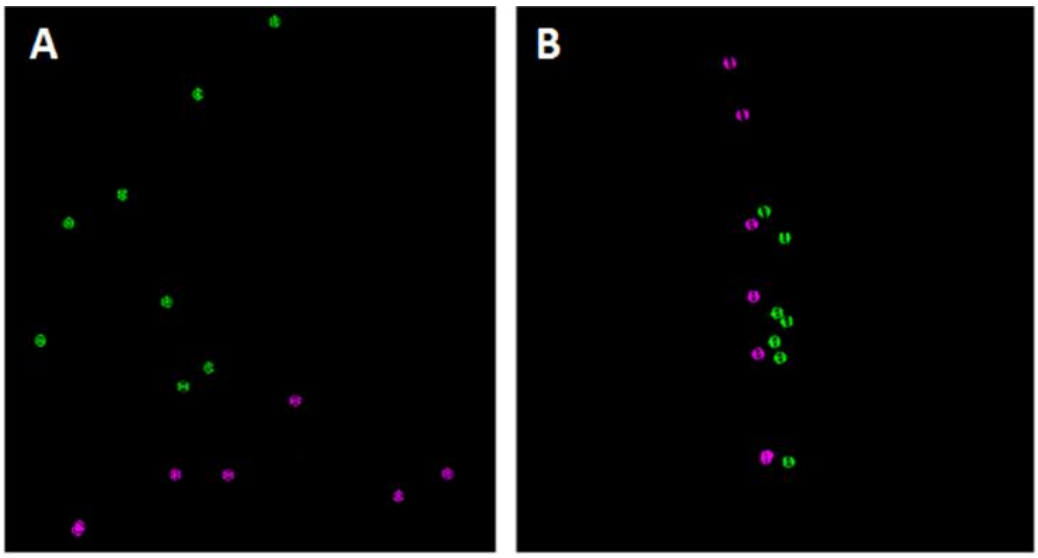

Fig. 4-13 Fiducial seeds on separate views for A) Sample 1 B) Sample 2

After the first full iteration of the seed tracking on both sample images, residual vectors were examined. Figure 4-14 shows the presence of residual vectors on the surfaces. The tail of the residual vector arrow shows the current position of the fiducial while the point head indicates the correct location. However this was not always true and instead of moving the fiducials with respect to the new arrow direction, the center of the seeds were selected as the new location for the seeds. At each iteration, the number of missing points and the mean error were recorded.

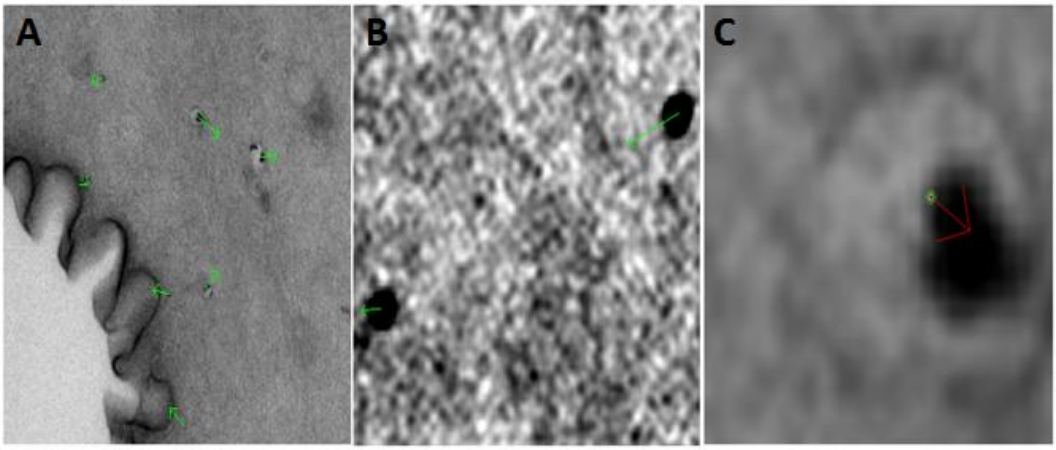

Fig. 4-14 A) Residual error vector on sample 1 image B) Residual error vector on Sample 2 image C) Residual error vector new location marked in red arrow

4.2.4 Transfer of Fiducials to Second Axis

As in the start of this chapter it was mentioned that Sample 2 contains tilt images on both axis, therefore the same *IMOD* routines were used for the second axis too. The process of aligning the second axis images is exactly the same as of first axis. However for seeding the fiducials on the second axis, the fiducials from the first axis were used as seeds. Figure 4-15 shows the transfer of the fiducials from *x-axis* on to the second axis.

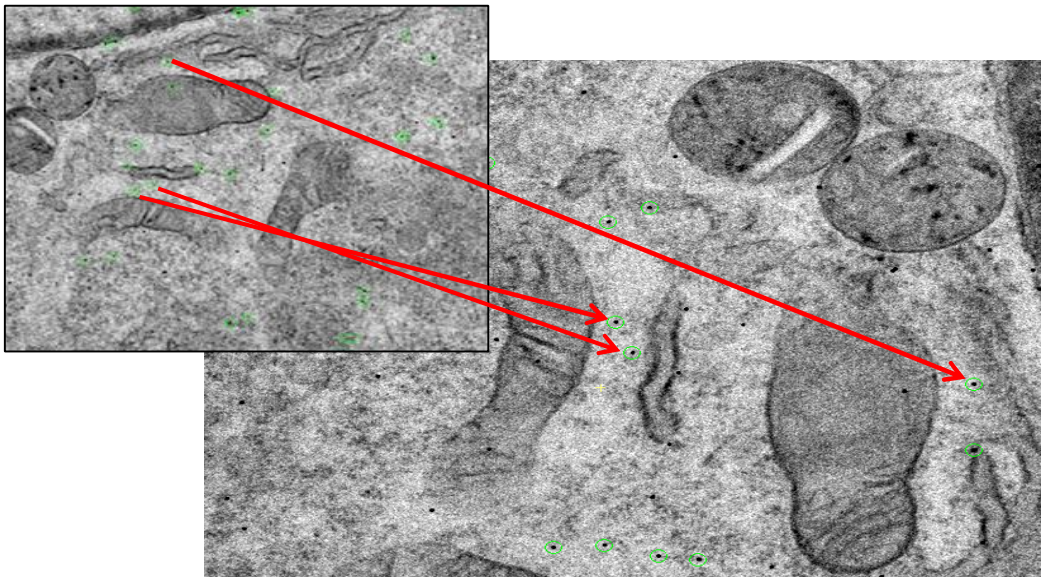

Fig. 4-15 Transfer of fiducials for second axis of sample 2

4.2.5 Tomogram Reconstruction

Tomogram generation in *IMOD* mainly consists of two phases. i) Tomogram positioning and ii) Final tomogram generation. In the first phase the boundary models for the tomograms were acquired. These boundary models represent the top, middle and the bottom surfaces of the image stacks. A boundary model is generated by analyzing the sample's boundaries in the *IMOD* and points are marked where the thickness of the material is visible. A pair of two distant points is connected by a line called a contour. As shown in Figure 4-16A, there are two contours on every surface of the material. In total there were 6 contours for all three surfaces. After the manual positioning of the tomogram's boundaries *IMOD* routine *Tomopitch (Etomo)* was run. This routine sets the values of different parameters such as the thickness and the final values of the tilt axis to make the tomogram look flat. Once these parameters are set by the *Tomopitch*, the *Tiltalign* routine of the *IMOD* is run to compute the final tomogram alignment. Finally the Backprojection algorithm is run and the final tomograms were reconstructed as shown in the Figure 4-16B for Sample 1 and Figure 4-16C for sample 2.

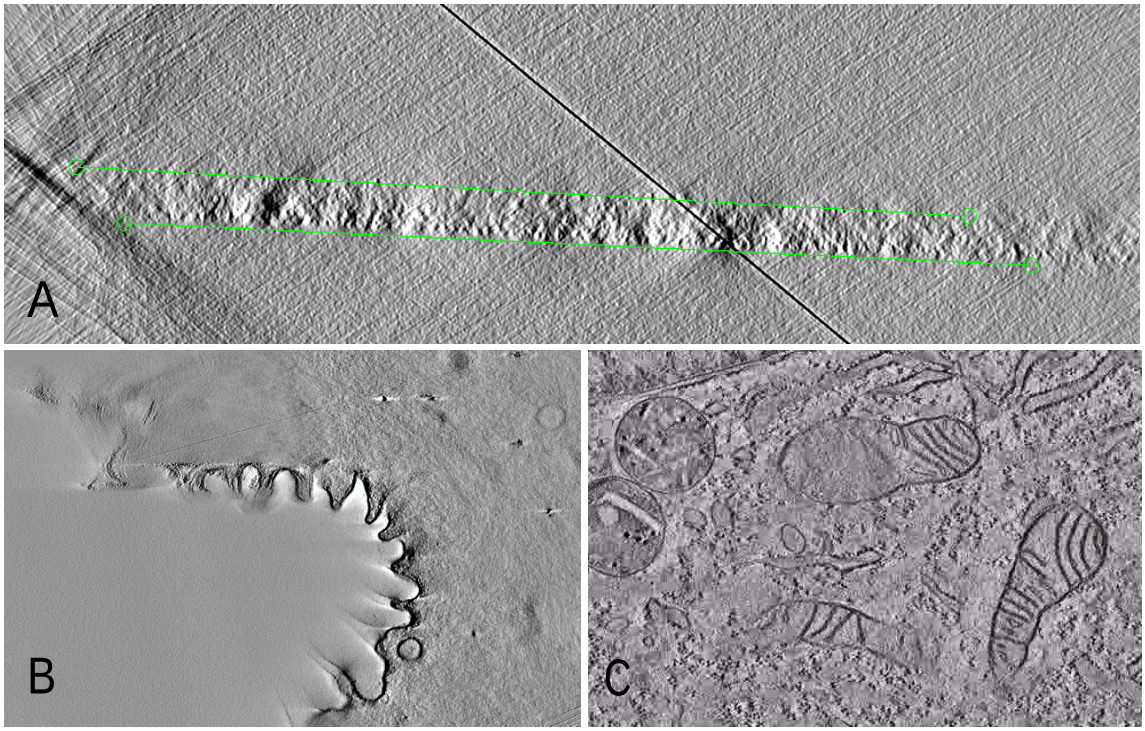

Fig. 4-16 A)Boundary value marking before final tomogram generation B) 2D view of sample 1 final reconstructed tomogram C) 2D view of sample 2 final reconstructed tomogram

4.3 Segmentation and Visualization

IMOD modules *Drawing tool* and *Interpolator* were used for the segmentation of the sub volumes of the generated Tomograms. *Etomo* generated the tomograms in the *.rec* format. The *.rec* models were loaded into *IMOD* and different built-in routines were used to mark various parts of the samples' volumes as shown in the Figure 4-17. These parts were saved as objects. An object may consist of many contours and where necessary some of these contours were filled by interpolation. Figure 4-18 and Figure 4-19 show the segmented parts of the materials for both tomograms.

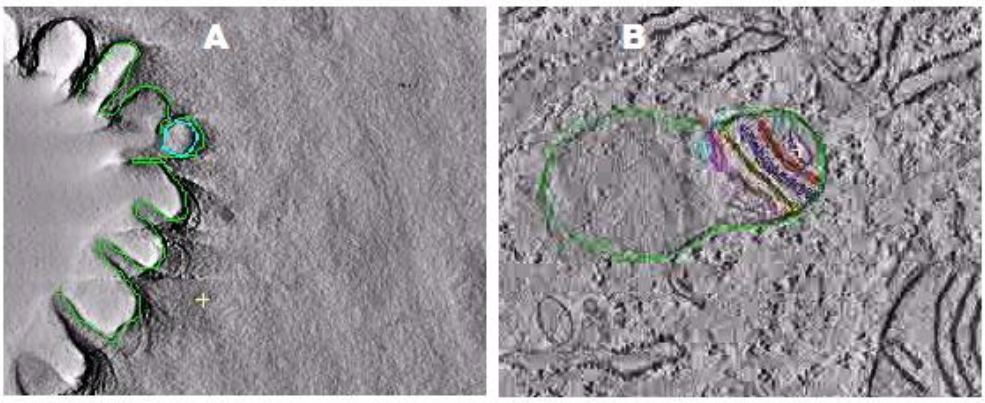

Fig. 4-17 Segmentation of A) Sample 1 tomogram B) Sample 2 tomogram

As shown in Figure 4-18 and Figure 4-19, for both cases the external segmentation (extracting the areas of interest) and the internal segmentation were performed.

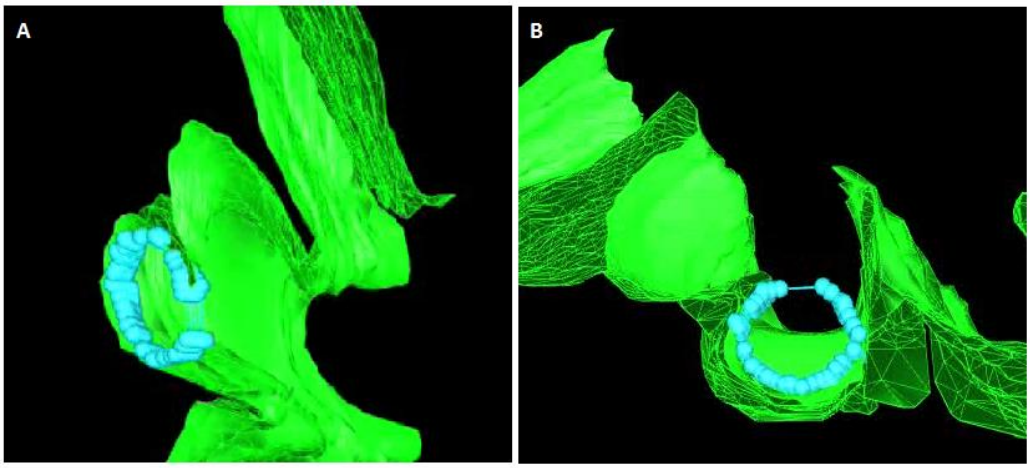

Fig. 4-18 3D segmentation of sample 1 tomogram

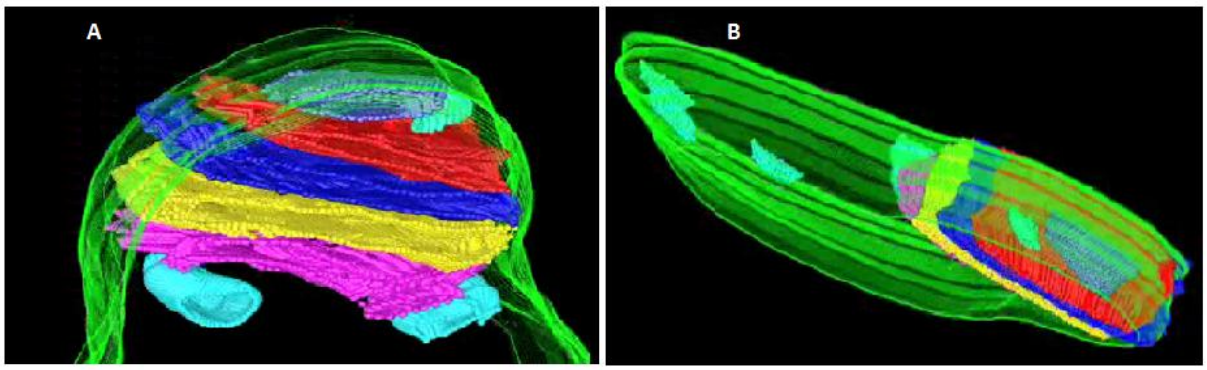

Fig. 4-19 3D view of sample 2 tomogram

Figure 4-18 shows the presence of a hole in the sample along the zigzag main body part. Along the length of the vessel at some places the boundary was thinner than other walls.

Figure 4-19 shows the presence of internal cristae in mitochondria. Different cristae were colored differently to show their structures. After the marking and interpolation of the contours and objects a mesh was drawn around the segmented area. The *VideoMach* software was used to run the simulation in the video format.

5 Results and Discussion

This Chapter presents the results and observations from the implementation of the different alignment and reconstruction techniques used in this thesis work first the results of the markerless alignment and feature selection are presented.

5.1 Markerless and Feature Points Alignment

The improvement in the markerless alignment process by the introduction of feature points resulted in the uniform alignment of the image stack of sample 1. Table 5-1 shows the location of 5 features points in all 101 images of sample 1. It can be seen that the locations of the points vary in every image. Due to the tilt shifts the distances vary in both *x* and *y* directions.

Table 5-1 Feature points locations in images of sample 1

Image No.	Point 1		Point 2		Point 3		Point 4		Point 5	
	X	Y	X	Y	X	Y	X	Y	X	Y
1	514	179	864	157	989	445	858	584	796	603
2	514	182	872	162	982	440	869	590	807	613
3	488	187	854	169	967	446	855	598	790	619
.										
.										
51	409	130	922	195	1115	512	996	653	899	656
.										
.										
99	456	193	839	210	935	486	839	635	756	645
100	510	186	886	202	977	473	881	623	803	632
101	492	202	863	217	950	488	853	640	775	646

In Figure 5-1 the graph shows the difference between the *x* and *y* coordinates of the features points in all 101 images. It's clear that with the incremental tilt change, the respective average differences in the *x* and *y* coordinates increases. The change in both negative and positive axis shows that the

difference values increase in an almost symmetric pattern. This symmetrical behavior is because of the equal tilt changes in both directions from the central (index) image.

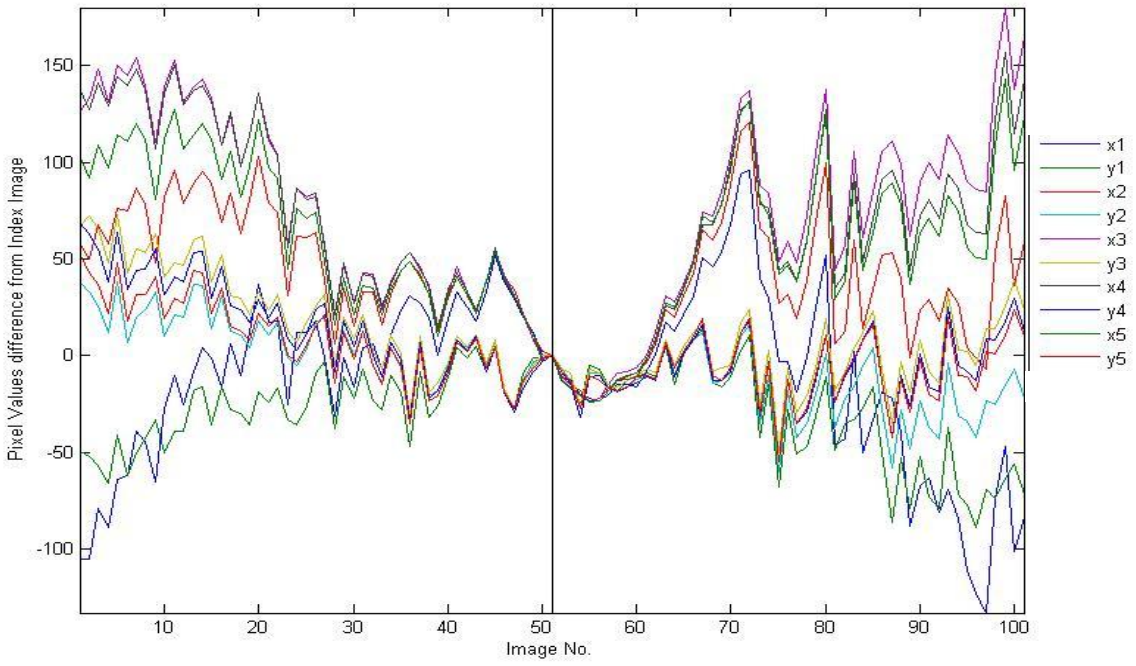

Fig. 5-1 Difference in the offset values of the feature points (x1, y1...x5, y5) with respect to the center image at 0°

Figure 5-2 shows the Euclidian distance graph of the pixel values in all 101 images with respect to the index image. After the fine alignment process based on the feature points locations, the new pixel values of the features points can be seen in Table 5-2.

The respective graph of the differences in the pixel locations in all 101 images can be seen in Figure 5-3 whereas Figure 5-4 shows the new Euclidian distance measures of the updated locations of the feature points after the alignment process.

It is clear from Figure 5-3 and Figure 5-4 that after the successful alignment the change in the *x* and *y* locations and the change in the distances of the 5 feature points in all 101 images is uniform. These results can also be verified with the distortion free video demonstration of this alignment process. The *Root Mean Square (RMS)* error value for this approach is 36.27.

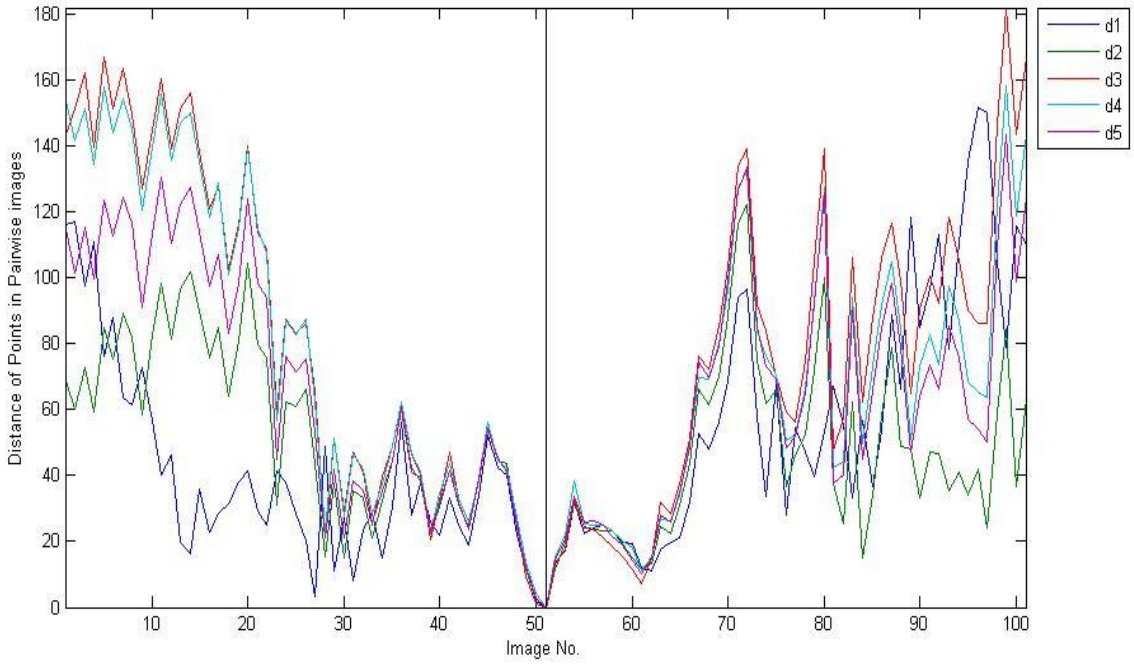

Fig. 5-2 Euclidian distances (d1...d5) of the feature points in the images with respect to the center image at 0°

Table 5-2 Feature points locations in aligned images of sample 1

Image No.	Point 1		Point 2		Point 3		Point 4		Point 5	
	X	Y	X	Y	X	Y	X	Y	X	Y
1	578	215	928	193	1053	481	922	620	860	639
2	573	214	931	194	1041	472	928	622	866	645
3	565	212	931	194	1044	471	932	623	867	644
.										
.										
51	409	130	922	195	1115	512	996	653	899	656
.										
.										
99	559	188	942	205	1038	481	942	630	859	640
100	567	192	943	208	1034	479	938	629	860	638
101	574	193	945	208	1032	479	935	631	857	637

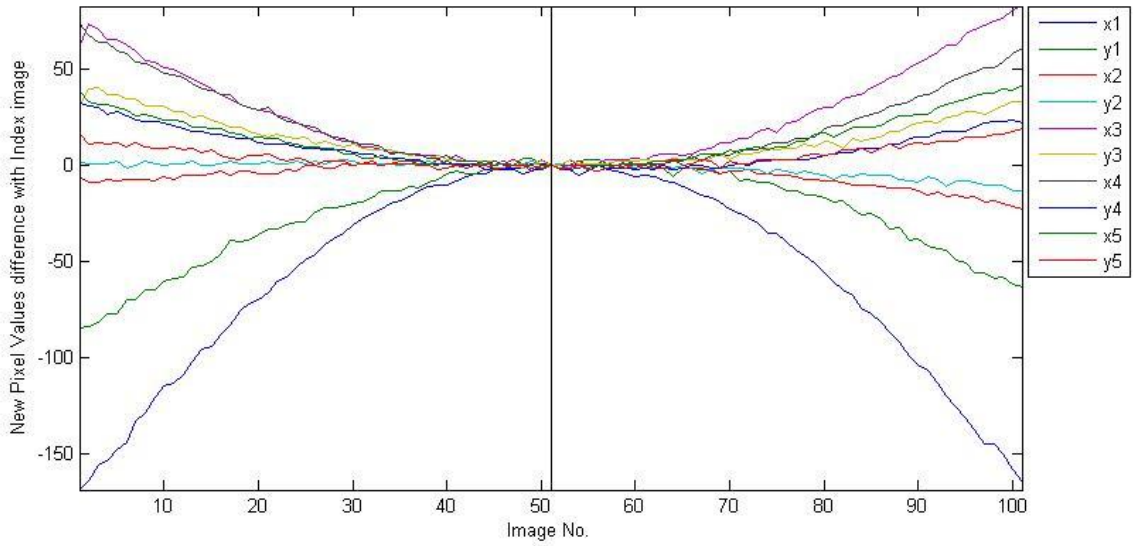

Fig. 5-3 Differences of the feature points (x1, y1...x5, y5) locations in the images with respect to the center image after alignment.

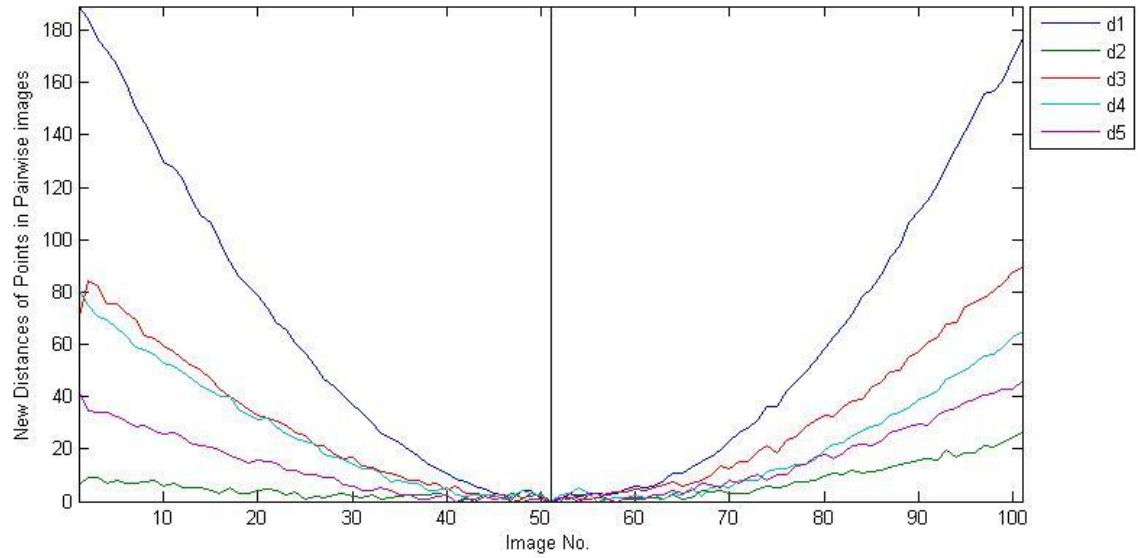

Fig. 5-4 Euclidian distances (d1..d5) of the feature points from the center image after alignment

5.2 SIFT based Alignment

Using SIFT algorithm the alignment methods were implemented based on the offset locations and their distances in the tilt images of sample 1. For total 101 images, the number of common feature points found was 2. SIFT features for different images were found on different various locations;

however few feature points were located in most of the images. To locate the two most common feature points the efficiency of the SIFT algorithm is shown in the Table 5-3.

Table 5-3 SIFT algorithm efficiency to locate the common feature points in sample 1 images

SIFT Methods	Number of images	Images with Common Features	Efficiency
Feature Point 1	101	84	83.16 %
Feature Point 2	101	80	79.20

Figure 5-5 shows the occurrence of the common points in all images, plotted on the center image. Red marked points were found in 84 images while blue marked points were found in 80 images. Data points marked with other colors were found in only few images and therefore were ignored while proceeding towards alignment of the image stack.

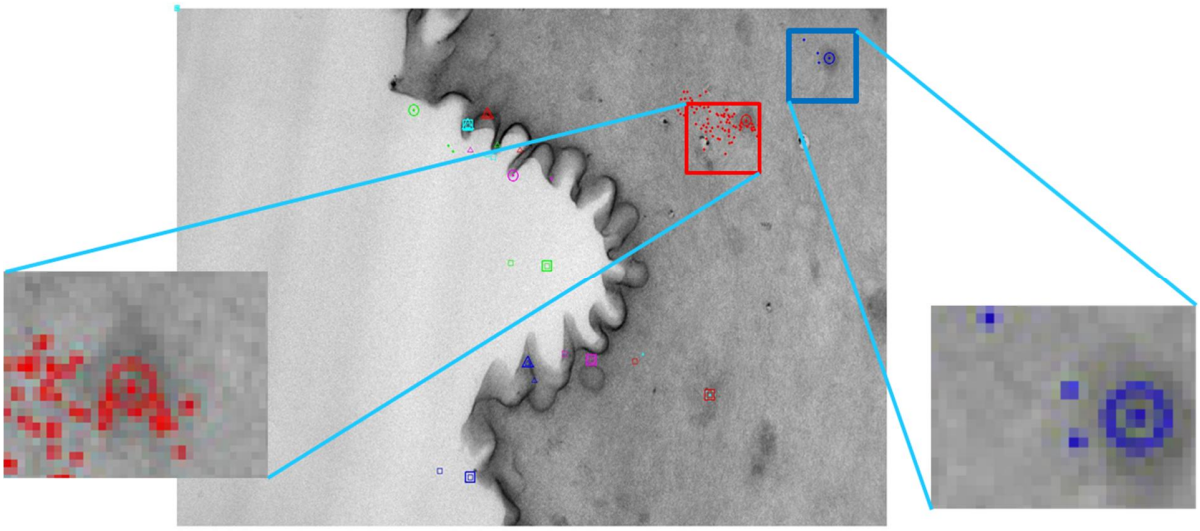

Fig. 5-5 Locations of the feature points in the image stack

The progress of the SIFT algorithm however still did not help in the attainable satisfactory alignment of the image stack. The offset distances in the *x* directions were easy to be moved while moving in the *y* directions produced some errors in the alignment process. Figure 5-6 shows the

difference of offsets for the two common points. Figure 5-7 shows the Euclidian distances between center image and the tilt stack before and after the alignment.

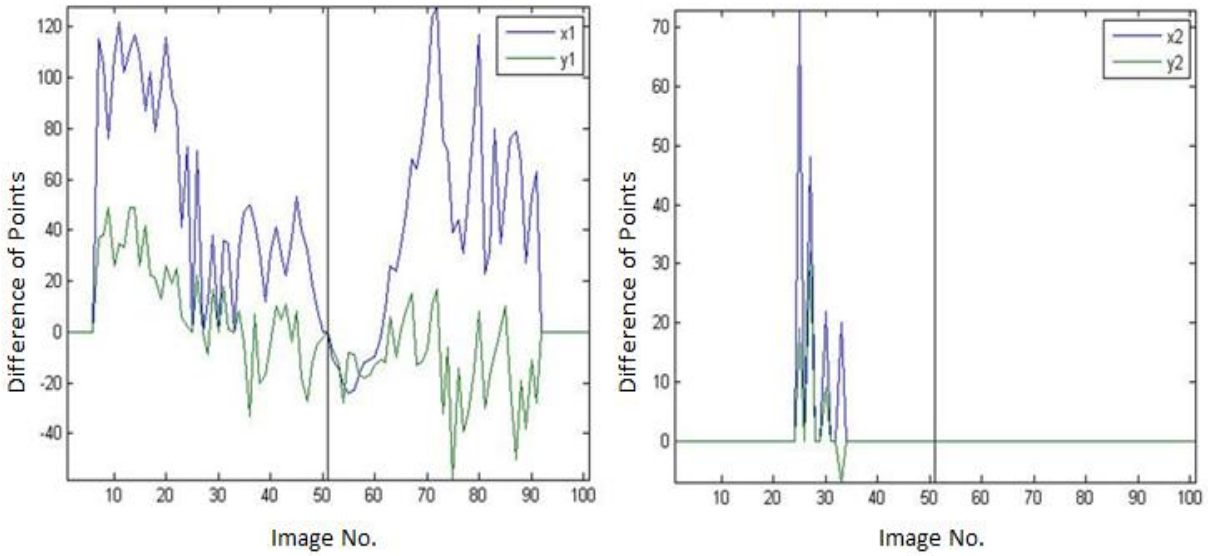

Fig. 5-6 Differences of the first (left) and second (right) feature points (x1, y1 & x2, y2) locations in the images with respect to the center image before SIFT alignment

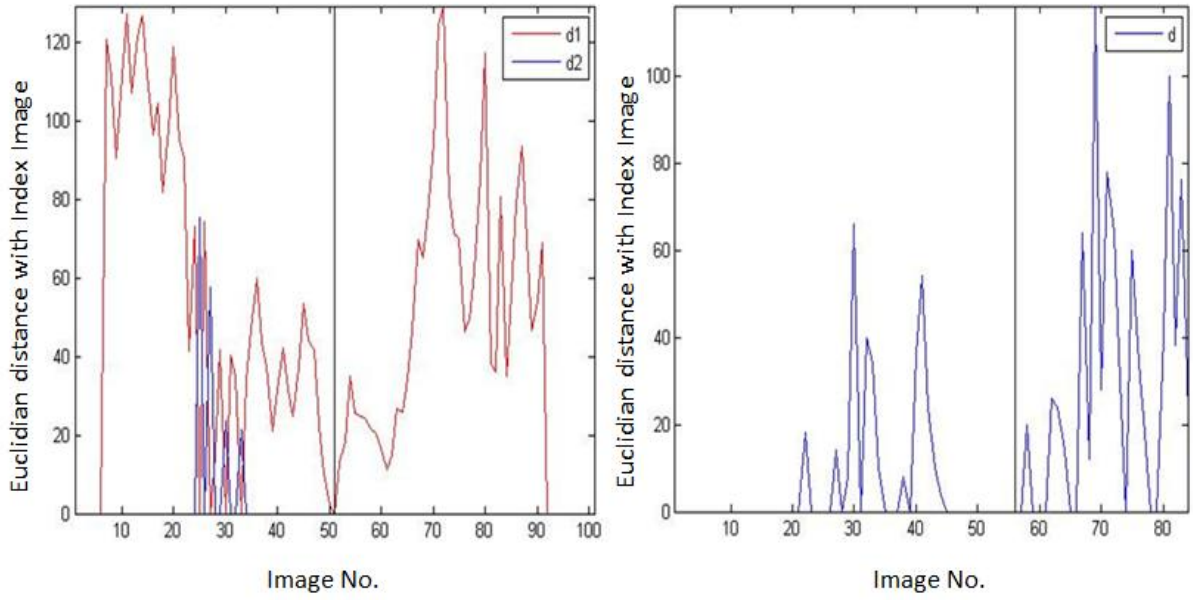

Fig. 5-7 Euclidian distances (d1, d2) of the feature points from the center image before SIFT alignment (left) and Euclidian distance (d) on Y axis after SIFT alignment (right)

It is clear that after the SIFT alignment the distances in x direction were completely moved in all the images whereas some images were still having distances in y axis. The root mean square errors *RMS* of the SIFT algorithm were calculated as 44.93 for the first feature point and 54.06 for the second feature point which are much higher than the target *RMS* value of 36.27 from the manual alignment.

Figure 5-7 shows that after the alignment the offset distances in the *x* axis were reduced to almost zero for most of the images but the *y* axis offset distances caused the distortion in the alignment.

5.3 Fiducial Markers Alignment

For the second technique using features extraction, the residual errors in the alignment process for the images of both samples were reduced in an iterative process. In each iteration of *TiltAlign* routine of *IMOD* the number of missing points was reduced and the corresponding residual error values were recorded. The number of iterations and the residual mean error values were almost same for both the samples as shown in Table 5-4.

Table 5-4 Convergence of Residual Error

Iteration No.	Sample 1 Residual Error (mean)	Sample 2 Residual Error (mean)
1	0.662	0.698
2	0.621	0.652
3	0.585	0.605
4	0.559	0.568
5	0.547	0.555
6	0.547	0.551
7	0.542	0.538

5.4 3D Visualization

Finally the successful segmentation and visualization results for the both the samples can be seen in Figure 5-8 and 5-9. Figure 5-8 shows the *3D* view of the First sample tomogram volume whereas Figure 5-9 shows the *3D* view of the second sample tomogram volume.

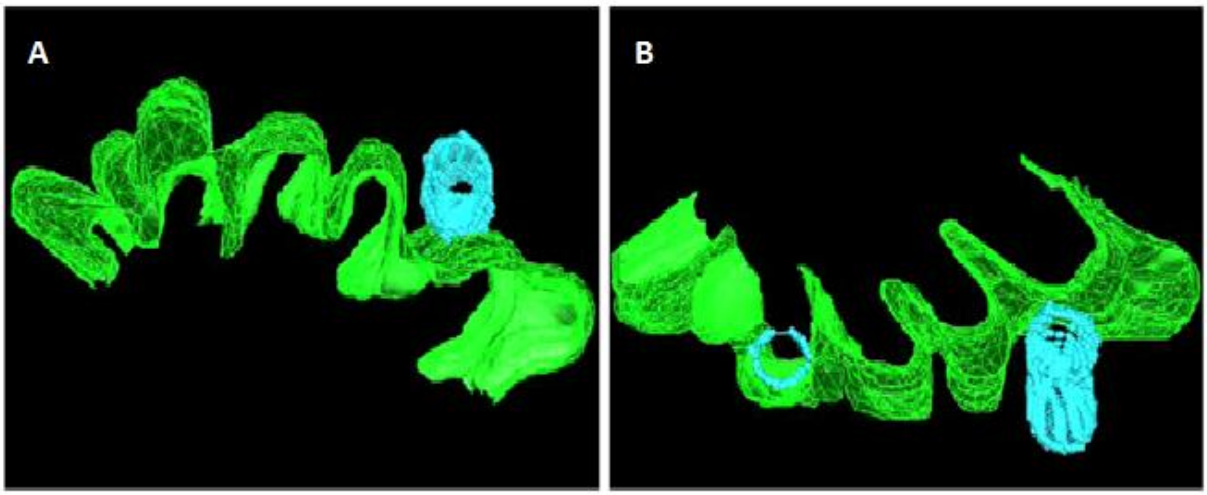

Fig. 5-8 3D view of the sample 1 reconstructed volume

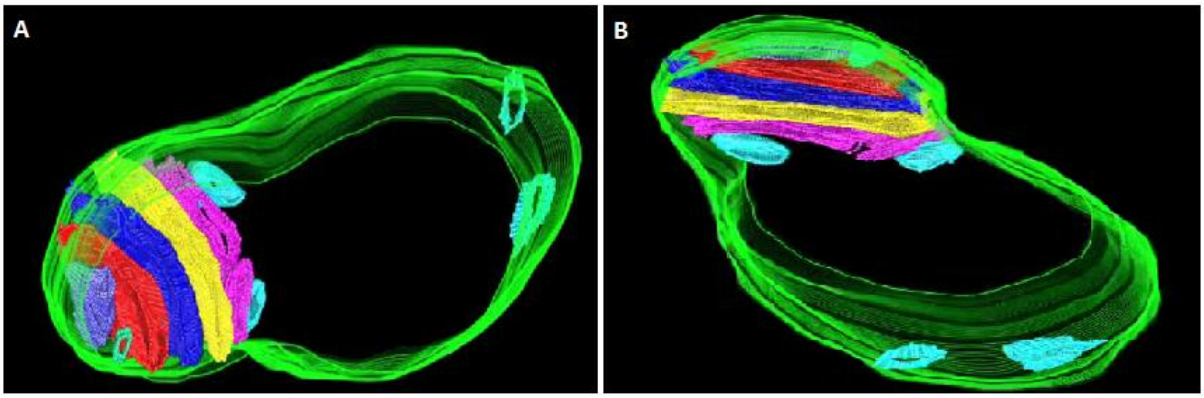

Fig. 5-9 3D view of the sample 2 reconstructed volume

5.5 Conclusions

The techniques implemented in the previous chapters work acceptably fine for the tested data. Using the feature based alignment, different algorithms can be written to perform the volumes reconstruction of the tomograms. *IMOD* and *MATLAB* routines will be best suitable to perform the segmentation and visualization processes. Different sub cellular objects can be viewed separately to examine the sub volume properties of the tomograms.

For the second technique based on fiducial markers alignment, *IMOD* software routine *3DMOD* can be used to extract the quantitative information of the sample from the final reconstructed and

segmented tomograms. For SIFT algorithm, the *RMS* value can be reduced more if the proper feature points are found for all the images. As seen from the manual alignment method of feature points that the error value was less than that of SIFT algorithm. For future work the SIFT algorithm will be optimized and the *RMS* value will be reduced. The residual error in the alignment can be further reduced if the tomographs are 16 bit images.

As a conclusion, this study suggests that *IMOD* routines when used carefully both for single axis and dual axis tomography will produce near perfect results. Following summary presents the time costs and the suitability of the implemented algorithm.

IMOD Alignment: (More Manual)

- Average time for single axis tomography = 20-25 Minutes
- Average time for dual axis tomography = 35-40 Minutes
- Alignment results suitable for volumetric reconstruction

Feature Point Based Alignment: (More Manual)

- Average time for single axis tomography 45-55 Minutes
- Alignment results suitable for volumetric reconstruction

SIFT based Alignment: (More Automatic)

- Average time for single axis tomography 40-45 Minutes
- Alignment results acceptable for few tomographs
- Alignment results not suitable (currently) for volumetric reconstruction
- More common feature points can produce better alignment but more time consumption

IMOD routines are useful when error free alignment is targeted, however seeding fiducials requires manual work as compare to the proposed technique using SIFT features.

Although the affine transform used for registering images produced near error free results in initial alignment, there is still some room to adjust the mismatches of pixels while matching displaced pixels in opposite views and the improvement can be achieved after smoothing all the displacement vectors [26]. Since Affine transformation is usually carried out on the edges on the image segments and if the edges are blurry or too noisy, it will not yield proper image registration [25]. SIFT features proved to be accurate and the whole process is automatic however manual changes might be needed when SIFT doesn't perform on some of the tilt images. The problem arises when the SIFT descriptor returns quite many feature points in opposite views. This leads to the problem of

optimizing the feature points. Minimum features points can help in achieving better alignment [27].

As for the Future work in the alignment process and volume reconstruction, the algorithm of local binary pattern and Simultaneous Iterative Reconstruction Techniques (SIRT) will be implemented and tested.

References

1. Bárcena, M., & Koster, A. (2009). Electron tomography in life science, *20*(8), 920--930.
2. Midgley, P., & Dunin-Borkowski, R. (2009). Electron tomography and holography in materials science. *Nature Materials*, *8*(4), 271--280.
3. Midgley, P., & Weyland, M. (2003). 3D electron microscopy in the physical sciences: the development of Z-contrast and EFTEM tomography. *Ultramicroscopy*, *96*(3), 413--431.
4. Zečević, J., de Jong, K., & de Jongh, P. (2013). Progress in electron tomography to assess the 3D nanostructure of catalysts. *Current Opinion In Solid State And Materials Science*, *17*(3), 115--125.
5. Ma, S. (n.d.). TEM 3D-Tomography of High-Pressure Frozen Cells Reveals Detailed Viral Components in the Maturation of Semliki Forest Virus.
6. Kübel, C., Lee, T., Su, D., Luo, J., Lo, H., & Russell, J. (2006). Application of electron tomography for semiconductor device analysis. *Microscopy And Microanalysis*, *12*(S02), 1552--1553.
7. Kohjiya, S., Katoh, A., Shimanuki, J., Hasegawa, T., & Ikeda, Y. (2005). Three-dimensional nano-structure of in situ silica in natural rubber as revealed by 3D-TEM/electron tomography. *Polymer*, *46*(12), 4440--4446.
8. Hernández -Garrido, J., Yoshida, K., Gai, P., Boyes, E., Christensen, C., & Midgley, P. (2011). The location of gold nanoparticles on titania: A study by high resolution aberration-corrected electron microscopy and 3D electron tomography. *Catalysis Today*, *160*(1), 165--169.
9. Wang, X., Lockwood, R., Malac, M., Furukawa, H., Li, P., & Meldrum, A. (2012). Reconstruction and visualization of nanoparticle composites by transmission electron tomography. *Ultramicroscopy*, *113*, 96--105.
10. Frank, J. (2006). Cryotomography: Low-dose Automated Tomography of Frozen-hydrated Specimens .*Electron tomography* (2nd ed.). New York: Springer. 113—117
11. Frank, J. (2006). Fiducial Marker and Hybrid Alignment Methods for Single and Double-axis Tomography .*Electron tomography* (2nd ed.). New York: Springer. 163—167

12. Winkler, H., & Taylor, K. (2006). Accurate marker-free alignment with simultaneous geometry determination and reconstruction of tilt series in electron tomography. *Ultramicroscopy*, *106*(3), 240--254.
13. Owen, C., & Landis, W. (1996). Alignment of electron tomographic series by correlation without the use of gold particles. *Ultramicroscopy*, *63*(1), 27--38.
14. Brandt, S., & Heikkonen, J. (2000). Automatic alignment of electron tomography images using markers, 277--287.
15. Levine, Z., Volkovitsky, A., & Hung, H. (2007). Alignment of fiducial marks in a tomographic tilt series with an unknown rotation axis. *Computer Physics Communications*, *176*(11), 694--700.
16. Díez, D., Seybert, A., & Frangakis, A. (2006). Tilt-series and electron microscope alignment for the correction of the non-perpendicularity of beam and tilt-axis. *Journal Of Structural Biology*, *154*(2), 195--205.
17. Liu, Y., Penczek, P., McEwen, B., & Frank, J. (1995). A marker-free alignment method for electron tomography. *Ultramicroscopy*, *58*(3), 393--402.
18. Masich, S., \"Ostberg, T., Norl\'en, L., Shupliakov, O., & Daneholt, B. (2006). A procedure to deposit fiducial markers on vitreous cryo-sections for cellular tomography. *Journal Of Structural Biology*, *156*(3), 461--468.
19. Perkins, G., Renken, C., van der Klei, I., Ellisman, M., Neupert, W., & Frey, T. (2001). Electron tomography of mitochondria after the arrest of protein import associated with Tom19 depletion. *European Journal Of Cell Biology*, *80*(2), 139--150.
20. Hayashida, M., Iijima, T., Tsukahara, M., & Ogawa, S. (2013). High-precision alignment of electron tomography tilt series using markers formed in helium-ion microscope. *Micron*, *50*, 29--34.
21. Frank, J. (2006). Markerless Alignment in Electron Tomography .*Electron tomography* (2nd ed.). New York: Springer. 187—212
22. Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. International Journal Of Computer Vision, 60(2), 91--110.
23. Sandberg, K., Mastronarde, D., & Beylkin, G. (2003). A fast reconstruction algorithm for electron microscope tomography. *Journal Of Structural Biology*, *144*(1), 61--72.

24. Gilbert, P. (1972). The reconstruction of a three-dimensional structure from projections and its application to electron microscopy. II. Direct methods. *Proceedings Of The Royal Society Of London. Series B. Biological Sciences*, *182*(1066), 89--102.
25. Lin, H., Du, P., Zhao, W., Zhang, L., & Sun, H. (2010). Image registration based on corner detection and affine transformation, *5*, 2184--2188.
26. Fuh, C., & Maragos, P. (1991). Motion displacement estimation using an affine model for image matching. *Optical Engineering*, *30*(7), 881--887.
27. Han, R., Zhang, F., Wan, X., Fernández, J., Sun, F., & Liu, Z. (2014). A marker-free automatic alignment method based on scale-invariant features. Journal Of Structural Biology, 186(1), 167-180.

Appendix 1: Matlab program routines: Initial alignment

crop.m

```
% Crop the Sample 1 images to reduce the area while preserving the region
% of interest

clear; close all
load OFFS
pathf = 'Sample1/';
files = dir(fullfile(pathf, 'TomoImage*'));
chopyx_topleft = max(offsets,[],2);
chopyx_bottomright = min(offsets,[],2);

% Croping images and saving

for i = 1:length(files)

    I = imread( fullfile(pathf, files(i).name) );
    [x, y] = size(I);
    xmin = chopyx_topleft(2);
    ymin = chopyx_topleft(1);
    width = (y + chopyx_bottomright(2)) - xmin;
    height = (x + chopyx_bottomright(1) - 65) - ymin;
    rect = [xmin ymin width height];
    I = imcrop(I, rect);
    imwrite( I, fullfile('CropTomo',sprintf('CropImage%d.tif', i)) )
    fprintf('Save image %d\n',i)
end
```

preAlignPair.m

```
% Electron Tomography Allignment
% Implementation of Image allignment method based on the cross-corelation
% and affine transformation of the cropped images in the Fourier domain .

% Part of the code used as it is available on matworks.com in examples
% Works for all images from most negative tilted angle to most positive
% tilted angle sequentially

% Computes allignment in pairwise images

clear; close all

pathf = 'CropTomo/';
files = dir(fullfile(pathf, 'CropImage*'));

iter = 1;
offsets = zeros(2,length(files));
angle = 0;

A0 = imread( fullfile(pathf, files(62).name) ); % image 62 , reference image
```

```
for k = 1:length(files)

    if k ~= round(length(files)/2)

        if k < round(length(files)/2)
            % rotation angle
            angle = angle + 1;
            im2 = k + round(length(files)/2)
           B = imread( fullfile(pathf, files(im2).name) ); % image k , compared image
            if k == 1
                im1 = round(length(files)/2)
                A = imread( fullfile(pathf, files(im1).name) ); % image 62
                ang2 = 0;
            else
                A = D; % previous image
                angle2 = angle - 1;
                ang2 = 1;
            end

        else
            % rotation angle
            angle = angle - 1;
            im2 = k + angle + angle
            B = imread( fullfile(pathf, files(im2).name) ); % image k
            if k == round(length(files)/2)+1;
                im1 = round(length(files)/2)
                A = imread( fullfile(pathf, files(im1).name) ); % image 62
                ang2 = 0;
            else
                A = D; % previous image
                angle2 = angle + 1;
                ang2 = 1;
            end

        end

        tx = 0;             % x translation
        ty = 0;             % y translation
        r2 = cosd(angle);
        T2 = [1/r2 0;
            0 1;
            tx ty];

        if ang2
            r1 = cosd(angle2);
            T1 = [1/r1 0;
                0 1;
                tx ty];
            % Affine transformations
            stretch_lc = maketform('affine',T1);
            resamp = makeresampler({'nearest','cubic'},'fill');
            x1_pos = fix( (T1(1)*(size(A, 2))-(size(A, 2)))/2 + 1);
            x2_pos = fix( (T1(1)*size(A, 2)) - (T1(1)*(size(A, 2))-(size(A, 2)))/2 );
            y_pos = size(A, 1);

            [A, xdata, ydata] = imtransform(A,stretch_lc,resamp,...
                'XData', [x1_pos x2_pos],...
```

```
        'YData', [1 y_pos],...
        'FillValues', 255);
end

stretch_lc = maketform('affine',T2);
resamp = makeresampler({'nearest','cubic'},'fill');
x1_pos = fix( (T2(1)*(size(B, 2))-(size(B, 2)))/2 + 1);
x2_pos = fix( (T2(1)*size(B, 2)) - (T2(1)*(size(B, 2))-(size(B, 2)))/2 );
y_pos = size(B, 1);

[C, xdata, ydata] = imtransform(B,stretch_lc,resamp,...
    'XData', [x1_pos x2_pos],...
    'YData', [1 y_pos],...
    'FillValues', 255);

% Cross-correlation starts
% Fourier transform
A1 = A;
A = fftshift(fft2(A));

[x, y] = size(A);
G = mat2gray(fspecial('gaussian',[x y],75)); % Gaussian filter
A = A .* G;

% Fourier transform
C = fftshift(fft2(C));
C = C .* G;                                  % Gaussian filter

% Compute cross-correlation
C = A .* conj(C);

% Apply inverse Fourier and inverse shift
C = abs(ifftshift(ifft2(C)));
close all
mesh(C)
pause

% Compute offset from center of the Cross-correlation matrix
maxC = max(C(:));
[ypeak, xpeak] = find(C == maxC);               % Find Max

% Offsets
[yc, xc] = size(C);
ych = fix(yc/2); xch = fix(xc/2);
yoff = ypeak - ych;
xoff = xpeak - xch;
if size(xoff,1) > 1
    fprintf('Many peaks\n')
    return
end
offsets(:,iter) = [yoff; xoff]; % record offsets
iter = iter + 1;

D = ones(yc, xc) * 0.5;
D = uint8(D);
if (xoff ~= 0 && abs(xoff) < xch)
    if yoff == 0
```

```
                yoff = 1;
            end

            if (yoff < 0 && xoff < 0)
                fprintf('yoff: -neg, xoff: -neg\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(1:yc-abs(yoff)+1, 1:xc-abs(xoff)+1) =...
                    B(abs(yoff):yc, abs(xoff):xc);

            elseif (yoff > 0 && xoff < 0)
                fprintf('yoff: +pos, xoff: -neg\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(abs(yoff):yc-1, 1:xc-abs(xoff)+1) =...
                    B(1:yc-abs(yoff), abs(xoff):xc);

            elseif (yoff < 0 && xoff > 0)
                fprintf('yoff: -neg, xoff: +pos\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(1:yc-abs(yoff)+1, abs(xoff):xc-1) =...
                    B(abs(yoff):yc, 1:xc-abs(xoff));

            elseif (yoff > 0 && xoff > 0)
                fprintf('yoff: +pos, xoff: +pos\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(abs(yoff):yc-1, abs(xoff):xc-1) =...
                    B(1:yc-abs(yoff), 1:xc-abs(xoff));

            end
            fprintf('ANGLE from center = %d\n',angle)
            fprintf('------------------------\n')
            fprintf('------------------------\n')

        else
            fprintf('WARNING xoff is zero\n')

        end
       imwrite( D, fullfile('preAlignedTomo',sprintf('preAlignPair%d.tif', k)) )
        close
        figure, imshowpair(A0, D, 'falsecolor')
        figure, imshowpair(A1, D, 'falsecolor')
    else
        offsets(:, iter) = [0; 0];
        iter = iter + 1;
        angle = 0;
    end
    pause
end
```

preAlignCenter.m

```
% Electron Tomography Allignment
% Implementation of Image allignment method based on the cross-corelation
% and affine transformation of the cropped images in the Fourier domain .

% Part of the code used as it is available on matworks.com in examples
% Works for all images from most negative tilted angle to most positive
```

```
% tilted angle sequentially

% Computes allignment from center image

clear; close all

pathf = 'CropTomo/';
files = dir(fullfile(pathf, 'CropImage*'));

iter = 1;
offsets = zeros(2,length(files));
angle = 62;

A = imread( fullfile(pathf, files(62).name) ); % image 62 , Central image
angle = 2;
for k = 61:length(files)

    if k ~= 62
        im2 = k;
        im1 = 62;
        A = imread( fullfile(pathf, files(im1).name) ); % image 62
        B = imread( fullfile(pathf, files(im2).name) ); % image k,  compared image

        if k < 62
            % rotation angle
            angle = angle - 1;
        else
            % rotation angle
            angle = angle + 1;
        end
        tx = 0;             % x translation
        ty = 0;             % y translation
        r = cosd(angle);
        T = [1/r 0;
            0 1;
            tx ty];

        % Affine transformations
        stretch_lc = maketform('affine',T);
        resamp = makeresampler({'nearest','cubic'},'fill');

        x1_pos = fix( (T(1)*(size(B, 2))-(size(B, 2)))/2 + 1);
        x2_pos = fix( (T(1)*size(B, 2)) - (T(1)*(size(B, 2))-(size(B, 2)))/2 );
        y_pos = size(B, 1);

        [C, xdata, ydata] = imtransform(B,stretch_lc,resamp,...
            'XData', [x1_pos x2_pos],...
            'YData', [1 y_pos],...
            'FillValues', 255);
        Ct = C;
        Ao = A;

        % Cross-correlation starts
        % Fourier transform
        A = fftshift(fft2(A));

        [x, y] = size(A);
```

```
G = mat2gray(fspecial('gaussian',[x y],75)); % Gaussian filter
A = A .* G;

% Fourier transform
C = fftshift(fft2(C));
C = C .* G;                                  % Gaussian filter

% Compute cross-correlation
C = A .* conj(C);

% Apply inverse Fourier and inverse shift
C = abs(ifftshift(ifft2(C)));
figure, mesh(C)

% Compute offset from center of the Cross-correlation matrix
maxC = max(C(:));
[ypeak, xpeak] = find(C == maxC);               % Find Max

if 1
    % Cut Correlation in threshold
    C( C(:) > (maxC - 8e8) ) = maxC - 8e8;
    figure, mesh(C)
    maxC = max(C(:));
    [ypeaks, xpeaks] = find(C == maxC);        % Find all peaks Max

    % Compute center of mass
    idxp = find(C(:) == maxC);
    M = double(Ct(idxp));
    x_cen_mass = round((xpeaks'*M) / sum(M));
    y_cen_mass = round((ypeaks'*M) / sum(M));
end

figure, imshowpair(Ao,Ct,'falsecolor')
hold on;
plot(xpeak,ypeak,'r.')
plot(x_cen_mass,y_cen_mass,'.')
rh = rectangle('Position',[min(xpeaks),min(ypeaks),...
    max(xpeaks) - min(xpeaks),...
    max(ypeaks) - min(ypeaks)],...
    'Curvature',[.5,.5]);
set(rh,'EdgeColor','r','LineWidth',1)
return
% Offsets
[yc, xc] = size(C);
ych = fix(yc/2); xch = fix(xc/2);
if 0
    yoff = ypeak - ych;
    xoff = xpeak - xch;
else
    yoff = y_cen_mass - ych;
    xoff = x_cen_mass - xch;
end
offsets(:,iter) = [yoff; xoff]; % record offsets
iter = iter + 1;

D = ones(yc, xc) * 0.5;
D = uint8(D);
```

```
        if (xoff ~= 0 && abs(xoff) < xch)
            if yoff == 0
                yoff = 1;
            end
            if (yoff < 0 && xoff < 0)
                fprintf('yoff: -neg, xoff: -neg\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(1:yc-abs(yoff)+1, 1:xc-abs(xoff)+1) =...
                    Ct(abs(yoff):yc, abs(xoff):xc);

            elseif (yoff > 0 && xoff < 0)
                fprintf('yoff: +pos, xoff: -neg\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(abs(yoff):yc-1, 1:xc-abs(xoff)+1) =...
                    Ct(1:yc-abs(yoff), abs(xoff):xc);

            elseif (yoff < 0 && xoff > 0)
                fprintf('yoff: -neg, xoff: +pos\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(1:yc-abs(yoff)+1, abs(xoff):xc-1) =...
                    Ct(abs(yoff):yc, 1:xc-abs(xoff));

            elseif (yoff > 0 && xoff > 0)
                fprintf('yoff: +pos, xoff: +pos\n')
                fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                D(abs(yoff):yc-1, abs(xoff):xc-1) =...
                    Ct(1:yc-abs(yoff), 1:xc-abs(xoff));

            end
            fprintf('ANGLE from center = %d\n',angle)
            fprintf('------------------------\n')
            fprintf('------------------------\n')

        else
            fprintf('WARNING xoff is zero\n')
        end
        close all
        imshowpair(Ao,D,'falsecolor')
        return
        imwrite(D, fullfile('preAlignedTomo',sprintf('preAlignCen%d.tif',...    k)) )
    else
        offsets(:, iter) = [0; 0];
        iter = iter + 1;
        angle = 0;
    end
end
figure, plot(1:length(offsets),offsets(1,:),'.-')
set(gca,'Ytick',[min(offsets(1,:)):max(offsets(1,:))])
Yticks = get(gca,'Ytick');
set(gca,'YTickLabel',cellstr(num2str(Yticks(:))))
ylabel('y offsets')
axis tight
figure, plot(1:length(offsets),offsets(2,:),'.-')
ylabel('x offsets')
axis tight

save OFFS offsets
```

Appendix 2: Matlab program routines ; Manual features alignment

markFeatures.m

```
% mark points in the original images cropped earlier

clear; close all

pathf = 'preAlignedTomo/';
files = dir(fullfile(pathf, 'preAlignCen*'));

marks = zeros(4,length(files));

for k = 1:length(files)

    I = imread( fullfile(pathf, files(k).name) );
    close;
    imshow(I,[])
    p = ginput(5);        % Manual points choosing
    marks(:,k) = p(:);

end
save markers.mat marks    % saving manually marked features
```

alignMarks.m

```
% Alignment of the images from the saved markers (manually plotted features)
%

clear; close all
load markers.mat      % saved feature points marked manually
writefiles = 0;
drawpair = 1;

pathf = 'preAlignedTomo/';
files = dir(fullfile(pathf, 'preAlignCen*'));
I = imread( fullfile(pathf, files(62).name) );
if writefiles
    imwrite( I, fullfile('alignedtomo',sprintf('alignmark%d.tif', 62)) )
end
[yc, xc] = size(I);
imshow(I, [])
hold on;
plot(marks(1,:),marks(3,:),'r.',marks(2,:),marks(4,:),'b.');
plot(marks(1,51),marks(3,51),'gx',marks(2,51),marks(4,51),'gx',...
    'MarkerSize',12,'LineWidth',2);

% Find difference
displt = zeros(4,size(marks,2));
for i = 1:size(marks,2)
    displt(:,i) = marks(:,51) - marks(:,i);
end
```

```
% Plot difference
figure, plot(displt')
hold on;
plot([51 51],[min(min(displt,[],2)) max(max(displt,[],2))],'k-')
legend('x1','x2','y1','y2')
axis tight

% Choose which points to use, options:(1,2,3)
switch 1
    case 1 % first point
        offsets = zeros(2,size(marks,2));
        offsets(1,:) = displt(1,:);
        offsets(2,:) = displt(3,:);

    case 2 % second point
        offsets = zeros(2,size(marks,2));
        offsets(1,:) = displt(2,:);
        offsets(2,:) = displt(4,:);

    case 3 % both points
        offsets = zeros(2,size(marks,2));
        offsets(1,:) = round(mean([displt(1,:); displt(2,:)]));
        offsets(2,:) = round(mean([displt(3,:); displt(4,:)]));
end

% loop over images
iter = 1;

for k = 12:length(files)-11

    if k ~= 62

        I2 = imread( fullfile(pathf, files(k).name) );
        % Offsets
        xoff = offsets(1,iter);
        yoff = offsets(2,iter);

        D = ones(yc, xc) * 0.5;
        D = uint8(D);
        if xoff == 0
            xoff = 1;
        end
        if yoff == 0
            yoff = 1;
        end

        if (yoff < 0 && xoff < 0)
            fprintf('yoff: -neg, xoff: -neg\n')
            fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
            D(1:yc-abs(yoff)+1, 1:xc-abs(xoff)+1) =...
                I2(abs(yoff):yc, abs(xoff):xc);

        elseif (yoff > 0 && xoff < 0)
            fprintf('yoff: +pos, xoff: -neg\n')
            fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
            D(abs(yoff):yc-1, 1:xc-abs(xoff)+1) =...
                I2(1:yc-abs(yoff), abs(xoff):xc);
```

```
        elseif (yoff < 0 && xoff > 0)
            fprintf('yoff: -neg, xoff: +pos\n')
            fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
            D(1:yc-abs(yoff)+1, abs(xoff):xc-1) =...
                I2(abs(yoff):yc, 1:xc-abs(xoff));

        elseif (yoff > 0 && xoff > 0)
            fprintf('yoff: +pos, xoff: +pos\n')
            fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
            D(abs(yoff):yc-1, abs(xoff):xc-1) =...
                I2(1:yc-abs(yoff), 1:xc-abs(xoff));

        end
        fprintf('Aligned %d\n',k)
        fprintf('------------------------\n')
        fprintf('\n')

        if drawpair
            close all
            imshowpair(I,D,'falsecolor')
            hold on;

plot(marks(1,iter),marks(3,iter),'ro',marks(2,iter),marks(4,iter),'bo',...
                'MarkerSize',6,'Linewidth',2);

plot(marks(1,51),marks(3,51),'gx',marks(2,51),marks(4,51),'gx',...
                'MarkerSize',12,'LineWidth',2);
            pause()
        end

        if writefiles
          imwrite( D, fullfile('alignedtomo',sprintf('alignmark%d.tif', k)) )
        end
        iter = iter + 1;

    else
        iter = iter + 1;

    end

end
```

Appendix 3: Matlab program routines ; SIFT based alignment

get_marks.m

```
% Based on matches this funcation saves the SIFT features as marks
% Other SIFT algorithm files not included here and used as in their
% original form can be downloaded from
% http://www.cs.ubc.ca/~lowe/keypoints/.


clear; close all

switch 2
    case 1
        pathf = 'preAlignedTomo/';
        files = dir(fullfile(pathf, 'preAlignCen*'));
        marks = zeros(4,length(files));
        for k = 1:length(files)
            I = imread( fullfile(pathf, files(k).name) );
            close;
            imshow(I,[])
            p = ginput(2);
            marks(:,k) = p(:);
        end

    case 2 % Running this part once and saving
        %         pathf = 'preAlignedTomo/';
        %         files = dir(fullfile(pathf, 'preAlignCen*'));
        allmatches = [];
        locs = {};
        iter = 1;
        for k = 12:112 %length(files)-11
            if k ~= 62

                % calculating matches using match.m
                [num, matches, loc1, loc2] = match('preAlignCen62.tif',...
                    sprintf('preAlignCen%d.tif',k));
                allmatches(iter,:) = matches;
                locs{iter} = loc2;
                title(sprintf('image number %d',k))
                drawnow
                close
                iter = iter + 1;
            end
        end
        save matchlocs allmatches locs loc1

end

%%
close all;
clear
load matchlocs.mat
countlocs = allmatches > 0;
[sval, sidx] = sort(sum(countlocs),'descend');
```

```
zeroidx = ~countlocs(:,sidx(2)) .* ~countlocs(:,sidx(4));
missing = find(zeroidx)'
[sval2, sidx2] = sort(sum(countlocs(missing,:)),'descend');

marks = [];
marks(1,51) = fix(loc1(sidx(2),2));
marks(2,51) = fix(loc1(sidx(2),1));
marks(3,51) = fix(loc1(sidx(4),2));
marks(4,51) = fix(loc1(sidx(4),1));
marks(5,51) = fix(loc1(sidx(9),2));
marks(6,51) = fix(loc1(sidx(9),1));

iter = 1;
n = 1;
for i = 1:size(countlocs,1)

    if i == 51
        iter = iter + 1;
    end
    loc2 = locs{i};
    if (allmatches(i,sidx(2)) > 0)
        marks(1,iter) = fix(loc2(allmatches(i,sidx(2)),2));
        marks(2,iter) = fix(loc2(allmatches(i,sidx(2)),1));
    elseif (allmatches(i,sidx(4)) > 0)
        marks(3,iter) = fix(loc2(allmatches(i,sidx(4)),2));
        marks(4,iter) = fix(loc2(allmatches(i,sidx(4)),1));
    elseif (allmatches(i,sidx(9)) > 0)
        marks(5,iter) = fix(loc2(allmatches(i,sidx(9)),2));
        marks(6,iter) = fix(loc2(allmatches(i,sidx(9)),1));
    else
        idx = find(countlocs(i,:));
        if ~isempty(idx)
            marks(6+n,51) = fix(loc1(idx(1),2));
            marks(6+n+1,51) = fix(loc1(idx(1),1));
            marks(6+n,iter) = fix(loc2(allmatches(i,idx(1)),2));
            marks(6+n+1,iter) = fix(loc2(allmatches(i,idx(1)),1));
            n = n + 2;
        else
            marks(1,iter) = fix(loc1(sidx(2),2));
            marks(2,iter) = fix(loc1(sidx(2),1));
        end
    end
    iter = iter + 1;

end

% Features marking and plotting

figure,
I = imread('preAlignCen62.tif');
subplot(351)
imshow(I); hold on;
plot(marks(1,51),marks(2,51),'bx','LineWidth',2)
plot(marks(1,:),marks(2,:),'b.','LineWidth',2)
subplot(352)
imshow(I); hold on;
plot(marks(3,51),marks(4,51),'rx','LineWidth',2)
```

```
plot(marks(3,:),marks(4,:),'r.','LineWidth',2)
subplot(353)
imshow(I); hold on;
plot(marks(5,51),marks(6,51),'gx','LineWidth',2)
plot(marks(5,:),marks(6,:),'g.','LineWidth',2)
subplot(354)
imshow(I); hold on;
plot(marks(7,51),marks(8,51),'cx','LineWidth',2)
plot(marks(7,:),marks(8,:),'c.','LineWidth',2)
subplot(355)
imshow(I); hold on;
plot(marks(9,51),marks(10,51),'cx','LineWidth',2)
plot(marks(9,:),marks(10,:),'c.','LineWidth',2)
subplot(356)
imshow(I); hold on;
plot(marks(11,51),marks(12,51),'cx','LineWidth',2)
plot(marks(11,:),marks(12,:),'c.','LineWidth',2)
subplot(357)
imshow(I); hold on;
plot(marks(13,51),marks(14,51),'cx','LineWidth',2)
plot(marks(13,:),marks(14,:),'c.','LineWidth',2)
subplot(358)
imshow(I); hold on;
plot(marks(15,51),marks(16,51),'cx','LineWidth',2)
plot(marks(15,:),marks(16,:),'c.','LineWidth',2)
subplot(359)
imshow(I); hold on;
plot(marks(17,51),marks(18,51),'cx','LineWidth',2)
plot(marks(17,:),marks(18,:),'c.','LineWidth',2)
subplot(3,5,10)
imshow(I); hold on;
plot(marks(19,51),marks(20,51),'cx','LineWidth',2)
plot(marks(19,:),marks(20,:),'c.','LineWidth',2)
subplot(3,5,11)
imshow(I); hold on;
plot(marks(21,51),marks(22,51),'cx','LineWidth',2)
plot(marks(21,:),marks(22,:),'c.','LineWidth',2)
subplot(3,5,12)
imshow(I); hold on;
plot(marks(23,51),marks(24,51),'cx','LineWidth',2)
plot(marks(23,:),marks(24,:),'c.','LineWidth',2)
subplot(3,5,13)
imshow(I); hold on;
plot(marks(25,51),marks(26,51),'cx','LineWidth',2)
plot(marks(25,:),marks(26,:),'c.','LineWidth',2)
subplot(3,5,14)
imshow(I); hold on;
plot(marks(27,51),marks(28,51),'cx','LineWidth',2)
plot(marks(27,:),marks(28,:),'c.','LineWidth',2)
subplot(3,5,15)
imshow(I); hold on;
plot(marks(29,51),marks(30,51),'cx','LineWidth',2)
plot(marks(29,:),marks(30,:),'c.','LineWidth',2)

save markersift.mat marks      % Save SIFT features as marks
```

match.m

```
% This function takes two images, finds the SIFT features, and
% displays the lines which connect the matched points.
% This funcation finally returns the number of matches found and displayed.
% This Funcation orignally written by David Lowe [22].
% Function comments originally written by David Lowe [22].
% Other funcations are also partially suplemented by the original code and
comments by David Lowe [22].



function [num, matches, loc1, loc2] = match(image1, image2)

% Find SIFT keypoints for each image
[im1, des1, loc1] = sift(image1);
[im2, des2, loc2] = sift(image2);
% distRatio: Only keep matches in which the ratio of vector angles from the
% nearest to second nearest neighbor is less than distRatio (Reference: David
Lowe SIFT code)
distRatio = 0.5;

% For each descriptor in the first image, select its match to second image.
des2t = des2';                          % Precompute matrix transpose
for i = 1 : size(des1,1)
   dotprods = des1(i,:) * des2t;        % Computes vector of dot products
   [vals,indx] = sort(acos(dotprods));  % Take inverse cosine and sort
results

   % Check if nearest neighbor has angle less than distRatio times 2nd.
   if (vals(1) < distRatio * vals(2))
      matches(i) = indx(1);
   else
      matches(i) = 0;
   end
end

% Create a new image showing the two images side by side.
im3 = appendimages(im1,im2);

% Show a figure with lines joining the accepted matches.
figure('Position', [100 100 size(im3,2) size(im3,1)]);
colormap('gray');
imagesc(im3);
hold on;
cols1 = size(im1,2);
for i = 1: size(des1,1)
  if (matches(i) > 0)
    line([loc1(i,2) loc2(matches(i),2)+cols1], ...
         [loc1(i,1) loc2(matches(i),1)], 'Color', 'c');
  end
end
hold off;
num = sum(matches > 0);
fprintf('Found %d matches.\n', num);
```

align_marks_sift.m

```
% This function performs alignment
% alignment from saved markers based on the SIFT method
%

clear; close all
load markersift.mat
writefiles = 1;
drawpair = 0;

I = imread( 'preAlignCen62.tif' );
if writefiles
    imwrite( I, fullfile('alignedtomo',sprintf('alignmark%d.tif', 62)) )
end
[yc, xc] = size(I);
imshow(I, [])
hold on;
plot(marks(1,:),marks(2,:),'r.',marks(3,:),marks(4,:),'b.',...
    marks(5,:),marks(6,:),'g.',marks(7,:),marks(8,:),'m.',...
    marks(9,:),marks(10,:),'c.',marks(11,:),marks(12,:),'rs',...
    marks(13,:),marks(14,:),'bs',marks(15,:),marks(16,:),'gs',...
    marks(17,:),marks(18,:),'ms',marks(19,:),marks(20,:),'cs',...
    marks(21,:),marks(22,:),'r^',marks(23,:),marks(24,:),'b^',...
    marks(25,:),marks(26,:),'m^',marks(27,:),marks(28,:),'g^',...
    marks(29,:),marks(30,:),'c^');
plot(marks(1,51),marks(2,51),'ro',marks(3,51),marks(4,51),'bo',...
    marks(5,51),marks(6,51),'go',marks(7,51),marks(8,51),'mo',...
    marks(9,51),marks(10,51),'co',marks(11,51),marks(12,51),'rs',...
    marks(13,51),marks(14,51),'bs',marks(15,51),marks(16,51),'gs',...
    marks(17,51),marks(18,51),'ms',marks(19,51),marks(20,51),'cs',...
    marks(21,51),marks(22,51),'r^',marks(23,51),marks(24,51),'b^',...
    marks(25,51),marks(26,51),'m^',marks(27,51),marks(28,51),'g^',...
    marks(29,51),marks(30,51),'c^',...
    'MarkerSize',14,'LineWidth',2);

% Find difference
displt = zeros(30,size(marks,2));
for i = 1:size(marks,2)
    for j = 1:size(marks,1)
        if marks(j,i) ~= 0
            displt(j,i) = marks(j,51) - marks(j,i);
        end
    end
end
% Plot difference
figure, plot(displt(1:2,:)')
hold on;
plot([51 51],[min(min(displt(1:2,:),[],2)) max(max(displt(1:2,:),[],2))],'k-
')
legend('x1','y1')
axis tight
figure, plot(displt(3:4,:)')
hold on;
plot([51 51],[min(min(displt(3:4,:),[],2)) max(max(displt(3:4,:),[],2))],'k-
')
```

```
legend('x2','y2')
axis tight

% Find distance
distpts = zeros(2,size(marks,2));
for i = 1:size(marks,2)
    n = 1;
    for j = 1:2:4
        if marks(j,i) ~= 0 && marks(j+1,i) ~= 0
            distpts(n,i) = sqrt((marks(j,51) - marks(j,i))^2 + ...
                (marks(j+1,51) - marks(j+1,i))^2);
        end
        n = n + 1;
    end
end
% Plot distance
figure, plot(distpts(1,:)','r-')
hold on;
plot(distpts(2,:)','b-')
plot([51 51],[min(min(distpts,[],2)) max(max(distpts,[],2))],'k-')
legend('d1','d2')
axis tight

% Choose the best images or all images
% close all
switch 1
    case 1

        tmarks = marks(1:4,7:90);
        offsets = zeros(2,size(marks,2));
        idx1 = find(displt(3,:) == 0);
        idx2 = find(displt(3,:) > 0);
        offsets(1,idx1) = displt(1,idx1);
        offsets(2,idx1) = displt(2,idx1);
        offsets(1,idx2) = displt(3,idx2);
        offsets(2,idx2) = displt(4,idx2);
        offsets = offsets(:,7:90);
        idxp = idx2 - 6;
        % loop over images
        iter = 1;
        newmarks = zeros(2,size(tmarks,2));

        for k = 7:90

            if k ~= 62

                I2 = imread(sprintf('preAlignCen%d.tif',k));
                % Offsets
                xoff = offsets(1,iter);
                yoff = offsets(2,iter);

                D = ones(yc, xc) * 0.5;
                D = uint8(D);
                if xoff == 0
                    xoff = 1;
                end
                if yoff == 0
```

```
    yoff = 1;
end

if (yoff < 0 && xoff < 0)
    if ismember( iter, idxp )
        newmarks(:,iter) = tmarks(3:4,iter) -...
            abs(offsets(:,iter));
    else
        newmarks(:,iter) = tmarks(1:2,iter) -...
            abs(offsets(:,iter));
    end
    fprintf('yoff: -neg, xoff: -neg\n')
    fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
    D(1:yc-abs(yoff)+1, 1:xc-abs(xoff)+1) =...
        I2(abs(yoff):yc, abs(xoff):xc);

elseif (yoff > 0 && xoff < 0)
    if ismember( iter, idxp )
        newmarks(1,iter) = tmarks(3,iter) -...
            offsets(1,iter);
        newmarks(2,iter) = tmarks(4,iter) +...
            offsets(2,iter);
    else
        newmarks(1,iter) = tmarks(1,iter) -...
            offsets(1,iter);
        newmarks(2,iter) = tmarks(2,iter) +...
            offsets(2,iter);
    end
    fprintf('yoff: +pos, xoff: -neg\n')
    fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
    D(abs(yoff):yc-1, 1:xc-abs(xoff)+1) =...
        I2(1:yc-abs(yoff), abs(xoff):xc);

elseif (yoff < 0 && xoff > 0)
    if ismember( iter, idxp )
        newmarks(1,iter) = tmarks(3,iter) +...
            offsets(1,iter);
        newmarks(2,iter) = tmarks(4,iter) -...
            offsets(2,iter);
    else
        newmarks(1,iter) = tmarks(1,iter) +...
            offsets(1,iter);
        newmarks(2,iter) = tmarks(2,iter) -...
            offsets(2,iter);
    end
    fprintf('yoff: -neg, xoff: +pos\n')
    fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
    D(1:yc-abs(yoff)+1, abs(xoff):xc-1) =...
        I2(abs(yoff):yc, 1:xc-abs(xoff));

elseif (yoff > 0 && xoff > 0)
    if ismember( iter, idxp )
        newmarks(:,iter) = tmarks(3:4,iter) +...
            abs(offsets(:,iter));
    else
        newmarks(:,iter) = tmarks(1:2,iter) +...
            abs(offsets(:,iter));
```

```
                    end
                    fprintf('yoff: +pos, xoff: +pos\n')
                    fprintf('yoff = %d, xoff = %d\n',yoff,xoff)
                    D(abs(yoff):yc-1, abs(xoff):xc-1) =...
                        I2(1:yc-abs(yoff), 1:xc-abs(xoff));

                end
                fprintf('Aligned %d\n',k)
                fprintf('------------------------\n')
                fprintf('\n')

                if drawpair
                    close all
                    imshowpair(I,D,'falsecolor')
                    hold on;

plot(tmarks(1,iter),tmarks(2,iter),'ro',tmarks(3,iter),tmarks(4,iter),'bo',...
                        'MarkerSize',6,'Linewidth',2);

plot(tmarks(1,45),tmarks(2,45),'gx',tmarks(3,45),tmarks(4,45),'gx',...
                        'MarkerSize',12,'LineWidth',2);
                    drawnow
                    pause()
                end

                if writefiles
                    imwrite( D,
fullfile('alignedtomo',sprintf('alignmark%d.tif', k)) )
                end
                iter = iter + 1;

            else
                newmarks(:,iter) = tmarks(1:2,45);
                iter
                iter = iter + 1;

            end

        end

        figure,
        imshow(I, [])
        hold on;
        idx = 1:size(newmarks,2);
        idx(idxp) = [];
        plot(newmarks(1,56),newmarks(2,56),'rx',...
            newmarks(1,idx),newmarks(2,idx),'r.',...
            newmarks(1,idxp(1)),newmarks(2,idxp(1)),'bx',...
            newmarks(1,idxp),newmarks(2,idxp),'b.',...
            'MarkerSize',12,'LineWidth',3);


        % Find difference again
        displt2 = zeros(size(newmarks));
        for i = 1:size(newmarks,2)
            if ismember( i, idxp )
                displt2(:,i) = newmarks(:,idxp(1)) - newmarks(:,i);
```

```
        else
            displt2(:,i) = newmarks(:,56) - newmarks(:,i);
        end
    end
    % Plot difference
    figure, plot(displt2')
    hold on;
    plot([56 56],[min(min(displt2,[],2)) max(max(displt2,[],2))],'k-')
    legend('x1','y1')
    axis tight

    % Find distance again
    distpts2 = zeros(1,size(newmarks,2));
    for i = 1:size(newmarks,2)
        if ismember( i, idxp )
        distpts2(i) = sqrt((newmarks(1,idxp(1)) - newmarks(1,i))^2 + ...
        (newmarks(2,idxp(1)) - newmarks(2,i))^2);
        else
            distpts2(i) = sqrt((newmarks(1,56) - newmarks(1,i))^2 + ...
                (newmarks(2,56) - newmarks(2,i))^2);
        end
    end
    % Plot distance
    figure, plot(distpts2')
    hold on;
    plot([56 56],[min(min(distpts2,[],2)) max(max(distpts2,[],2))],'k-')
    legend('d')
    axis tight

    % Compute Error
    idx1 = idx1(7:90);
    idx1(idxp) = [];
    RMSE1 = mean( sqrt((distpts2(idx) - distpts(1,idx1)).^2) )
    RMSE2 = mean( sqrt((distpts2(idxp) - distpts(2,idx2)).^2) )
    RMSE = RMSE1 + RMSE2

    save newmarks displt displt2 distpts distpts2 RMSE

end
```