INFORMATION EXTRACTION ON BIOLOGICAL, ORGANIC AND INORGANIC MATERIALS USING IMAGE PROCESSING TECHNIQUES

MUHAMMAD JEHANGIR

Information Extraction on Biological, Organic and Inorganic Materials Using Image Processing Techniques

by

© Muhammad Jehangir

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Computational Science Programme

Memorial University of Newfoundland

May, 2006

St. John's Newfoundland

Your file Votre référence
ISBN: 978-0-494-19370-9
Our file Notre référence
ISBN: 978-0-494-19370-9

Abstract

Information extraction from images is essential to research in many science faculties today, from computer science to bioinformatics and medicine. My research focuses on the analysis of structural changes in biological, organic and inorganic materials under a variety of conditions. Therefore, I process images from before and after an event (such as the addition of ions) to quantify exactly how much change is occurring in the material.

My major focus is to write code to automate finding objects/structures (in an image) and calculating their sizes, heights, orientations, and distributions. The data comes from atomic force microscope measurements, which produce multi-layered two-dimensional arrays of data (i.e. sets of three-dimensional images). The image processing involves noise reduction, background leveling, object identification, and then the calculation and display of statistical information. The code used to automate these processes and to present the features to the user is written in the IgorPro Scripting Languange.

The processing of an image proceeds iteratively, where an initial identification of objects/structures in an image leads to a definition of background and noise, which can then be adjusted to better identify the objects. Background leveling can be done vertically as well as horizontally, as per the requirements of a specific image. Distribution (spacing) of objects is calculated in different ways for isotropic versus non-isotropic arrangement of objects. The information about the objects in the images is presented in comparison graphs and summary tables.

Acknowledgements

Thanks to Dr. Y. Zhao and N. Zhou (Department of Chemistry), Dr. J. Robinson and M. Hayley (Department of Biochemistry), Dr. J. Shirokoff and Sanjeev Vasisht (Engineering), Dr. A. Yethiraj (Physics and Physical Oceanography), and K. Soper and M. Sun (Merschrod group, Chemistry) for samples and/or images. I also acknowledge financial support from the Natural Science and Engineering Resouce Council (Canada) and the Canada Foundation for Innovation.

Contents

List of Figures

Chapter 1

Introduction

Computer Imaging can be defined as the acquisition, visualization and processing of information by computer. Image processing is computer imaging wherein the application involves a human being in the visual loop (images are to be examined and acted upon by humans). Some of the major challenges in image processing are the separation of image data from noise, background from objects of interest, and the processing of image data to extract desired information.

The application of imaging and its processing is a key area of research in the fields of medicine, security systems, biochemistry, geography, engineering and physics [1]. Medical imaging is one of the fastest growing areas in medical science. In recent times most of the diagnosis includes: imaging of the damaged tissues and processing those images to know about the magnitude of damage and the appropriate response to it is determined [8]. Security is one of the major concerns in our rapidly changing world. Image processing is used in most secure systems, as in the form of biometric authentication, like authorized entry to some secure building. This security system

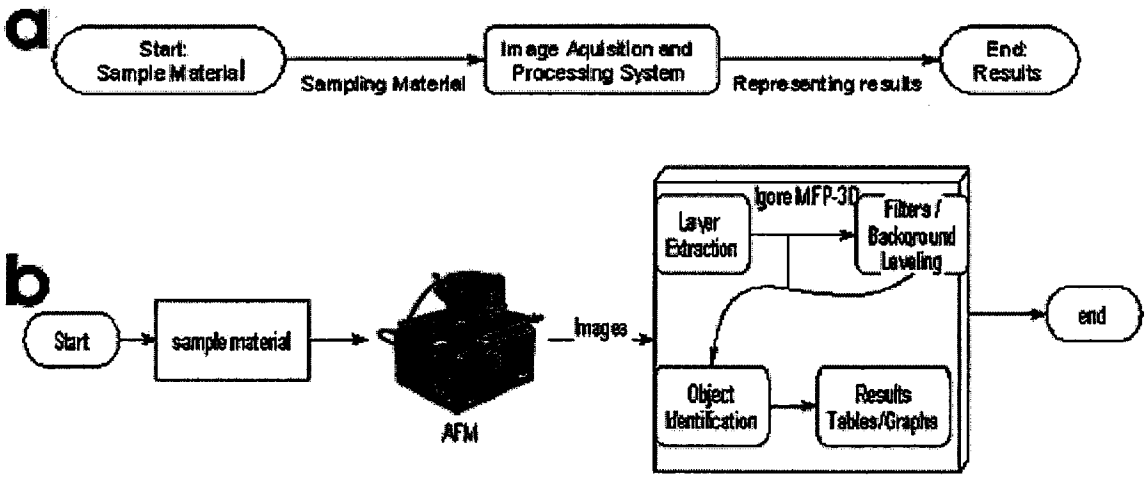


Figure 1.1: (a) Black Box View of System and (b) White box view of system

normally uses a fingerprint, retina or face recognition system to permit authorized entry [9]. In geography, images of earth structures below the surface are analyzed for finding sources of oil or any possible changes occurring inthe structure of earth crust, i.e. that can lead to earth quakes [10]. These are a few applications of imaging and image processing, but the scope of image processing is not limited to the mentioned fields.

Image processing includes image acquisition, image compression, image enhancement and image understanding [1]. Image Acquisition is the process of obtaining images of various objects (such as human being, animal, tree, location, car or any tangible object) [1,2]. The scope of this research deals with images of proteins, polymers, and metal nanoparticles under a variety of chemical and physical conditions. These images are obtained using Atomic Force Microscopy (AFM) [3,4], a technique used to image nano-scale materials. My approach to image processing is illustrated in the schematics of Figure 1.1.

Image Compression is reducing the massive amount of data that represents an image. This is done by removing unnecessary or redundant data from the image. In our research, the image is compressed by selecting the appropriate layer of data from a multilayer image.

Image Enhancement involves improving a selected image visually. Enhancement methods are problem specific. For example, an enhancement method used to improve a satellite image's quality may not be suitable for the enhancement of medical images [5,6]. Our research uses noise filters (such as mean filters, median filters, and Fourier transform filters) and background leveling as enhancement tools. Background leveling levels the uneven background of an image to avoid losing small objects and helps in finding the true heights of objects.

Image Interpretation is the transformation of image information into a format understandable by humans. For example if an image is processed to find a human face, then extracting the face from an image and matching it with the faces from databases will be its interpretation [7].

In this thesis my research is focused on developing routines to automate and facilitate the processing of biochemical images to find and represent the size/distribution of objects/structures (in an image) in summary tables or graphs, as outlined in Figure 1.2. The image processing starts with imaging the sample (proteins in various solutions) by AFM. Our goal is to find structures and structural changes occurring in proteins and the reasons for these changes. The summary about structures in the image, represented in the form of tables or graphs that tells us about changes occurring and its causes.

One of the key features of this research is that we compress images as our first

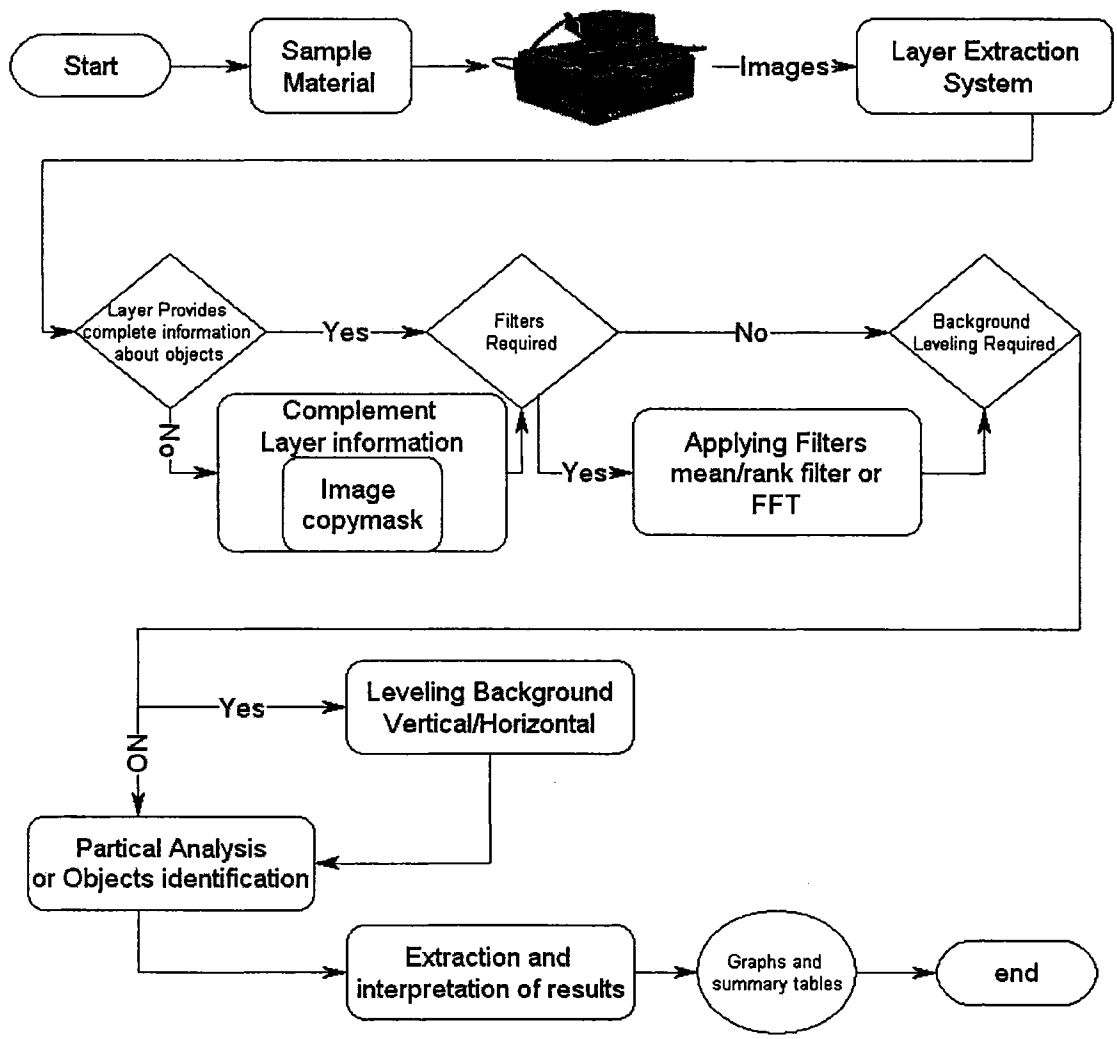


Figure 1.2: System functional flow diagram

step. That leads to the processing of the minimum required data for the latter phases of the process. To improve the time efficiency of our system, we filter the image information from three layers of data, where each layer is a 3-D (three-dimensional) image represented by a 2-D array of data values, as shown in Figure 1.3. We convert the 3-D image into a 2-D binary array, which provides us information about structures (or objects) in the images. The next step is to make the image ready for structures/object identification. This step involves applying filters in the case of noise and leveling image background if it is not even. An algorithm (steps to identify objects in the image) is then used to identify the structures/objects in it and information about these structures/objects is represented in the form of tables and comparison graphs. The future of this research is to simulate the changes and to process images that are obtained from imaging instruments other than atomic force microscopy.

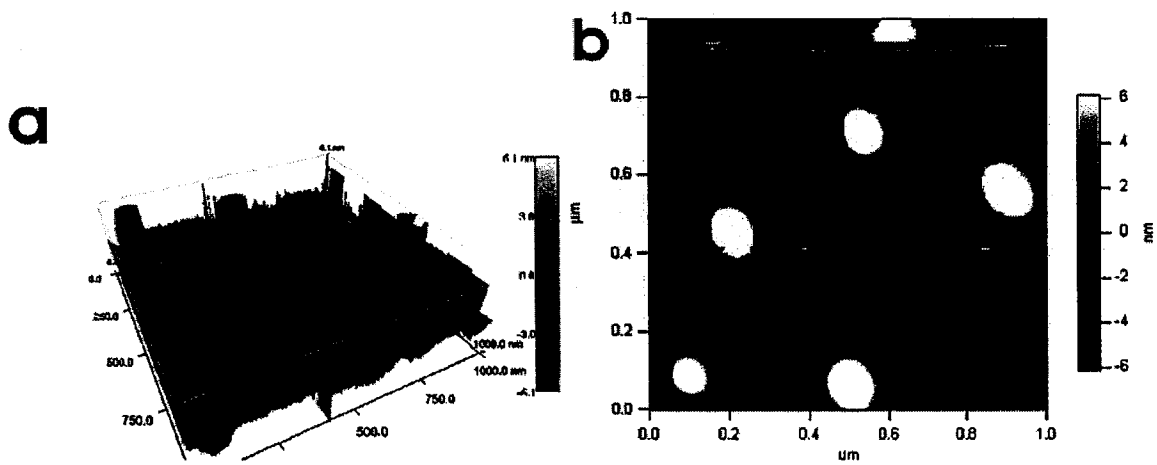


Figure 1.3: Two views of an AFM image; (a) shows the 3-D view of image of polymer on mica and (b) shows the height layer of same image.This image ©Ming Sun, 2005. Used with permission.

The thesis is divided into five chapters dealing with various issues and functionalities of the system where my code serves two purposes:

1. to expose already available services (by WaveMetrics [11]) in a user friendly fashion; and

2. to add desired services which are not already available.

The list of services provided or exposed by Igor (WaveMetrics [11]) are Layer extraction, Noise filters (*i.e.* mean filters, median filters, FFT filters) and particle analysis. The services implemented in this research are copying image mask, leveling of uneven background (*i.e.* vertical and horizontal leveling), calculating objects heights, calculating the average distances between objects/structures (for isotropic and non-isotropic cases), recording and representation of results, and extension of the system to non-Igor images.

The second chapter, "System Functionality", gives a brief overview of the whole system, *i.e.* features, limitations and applications of the system. Chapter 3, "Image processing", corresponds to details of the functionality: what types of services are provided, how these services work, and their advantages and limitations. This chapter also discusses the algorithms used for providing different services during the processing of an image. Chapter 4, "Results", discusses the image analysis for a series of case studies in biochemistry and material sciences. The results tell us exactly what types of changes occur in the structure and distribution of materials if combined with any other material or behavior of materials at different temperatures. This chapter also includes details about the processing of SEM images and records statistical and graphical results. Last but not least, Chapter 5, "Conclusion", deals with system advantages

and efficiencies and also discusses the limitation of different services provided. This part of the thesis also suggests future directions for research.

Each chapter includes numerous references and examples for the materials presented. The material is presented in a conceptual and application-oriented manner so that it gives an immediate understanding of how each of the topics fits into the overall system. The appendices of the thesis include key word explanations, a manual of the services, various sets of processed images, pseudo algorithm of services provided, and fully commented code of services.

Chapter 2

An Approach towards Segmentation of Images with 3-D Information

2.1 Introduction

Image segmentation is one of the focuses of computer vision and image processing. Segmentation is to distinguish between the objects of interest and "the rest." This latter group is also referred to as the background. Segmentation leads to information extraction from an image. This information extraction from the images is utilized in various fields of science and engineering. [1, 2] For example, segmentation and object extraction are widely used in security systems based on finger print, retina or face recognition. In medicine, molecular imaging makes it possible to reveal the activity of different molecules inside the body at different scales.

We apply image processing and segmentation to the field of biomaterials. In our experiments, we create images of surface of materials as proteins and enzymes using AFM (atomic force microscopy) [3, 4]. We use AFM to create an image of the surface of materials containing nano-scale objects and to observe how these materials/surfaces respond to different solutions and different environments. We analyze and process these images having nano-scale objects and observe the changes at different conditions. One of the key aspects of our image processing is that we process two-dimensional arrays of data and extract three-dimensional information. We use Igor [11] as the image processing language as it is also used to acquire the data. The approach we use for image segmentation and processing is targeted for images scanned through AFM. However, the routines we develop can be used for images scanned using SEM (scanning electron microscopy).

2.2 Image Processing

The process of segmentation and image processing can be further divided into several steps:

- layer extraction (with option of maintaining inter-layer correlation),
- noise removal,
- background leveling,
- object identification, and
- presentation and recording of results

2.2.1 Layer Extraction

The first step in processing the scanned image is to extract the layer of interest for further processing. The AFM images are of three layer format as shown in Figure 2.1, but the information of our interest can be extracted from only one layer in most of the cases. So we extract the favourable layer that contain information of our interest from image and process it for information extraction to save processing time. This technique helps in speeding up the image processing and information extraction process. Correlation between layers can be maintained where necessary to apply image information from one layer to another, such as applying boundaries defined by friction to the object height information.

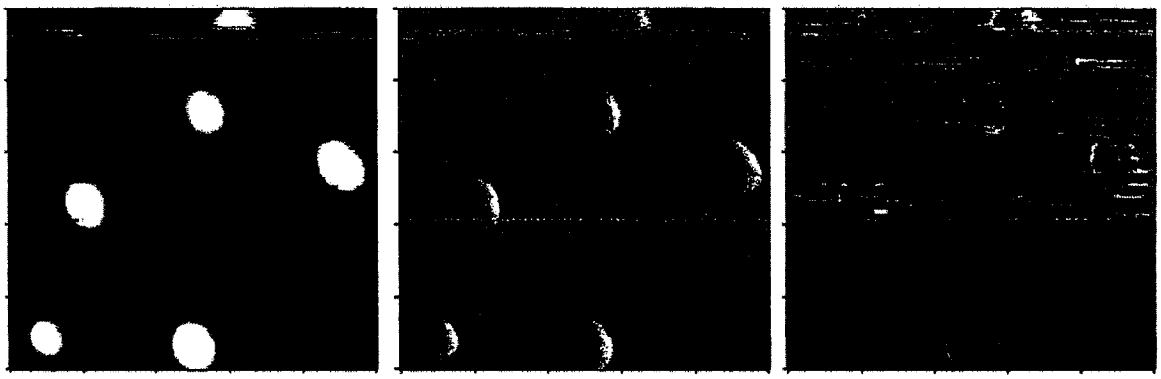


Figure 2.1: Shows the image of polymer on mica, where the height trace represent heights of objects, the amplitude trace represents error in feedback mechanism, and phase trace represents elasticity of sample. This image ©Erika Merschrod, 2005. Used with permission.

2.2.2 Noise Removal

Generally, several noise sources can affect the AFM image (i.e. environmental noise, roughness of the support surface [12]). The noise that most commonly affects an AFM image is due to a single source that leads to distributed spots. This noise can be classified as impulsive, and can be filtered out using a median filter. Sometimes the noises are uniformly distributed in the whole image and can be eliminated using a mean filter. There are cases where regular noise is superimposed over the image pixel lines; we handle it by removing the superimposed noise through a Fast Fourier Transformation (FFT) filter. Figure 2.2 shows the same image of collagen fibers but the superimposed noise in Figure 2.2a is removed in 2.2b using an FFT filter.

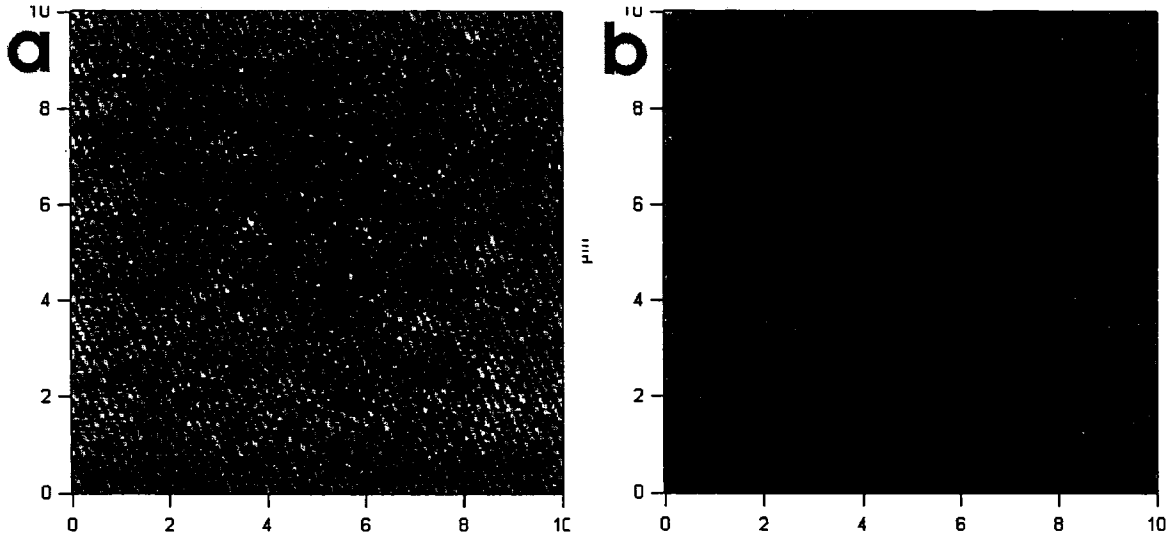


Figure 2.2: A fast-fourier transform (FFT) filter of image (a) removes high frequency noise and results in image (b). This image ©Ming Sun, 2005. Used with permission.

2.2.3 Leveling Background

One of the focuses of image segmentation and processing is to find the true heights of objects in an image. Unfortunately, most of the AFM images have uneven backgrounds [12]. This uneven background leads to inaccurate objects'/structures' identification, for example missing small objects in object identification.

Figure 2.3a shows that the central part of the background of the image is high (brightness represents height), so it will make a huge object in the center of the image. We need a uniform background to obtain true heights of the objects in an image. To level the background of our image, we apply our image background leveling algorithm. This algorithm levels the background as per requirement of the image (vertically or horizontally). The algorithm selects a pixel line in the image for leveling, and then applies the differences of that pixel line and average of that pixel line to the whole image matrix. Our results show that the leveling background algorithm leads to good results in object identification and height calculation. Figure 2.4 shows the vertical flattening of the first image into the second image and similarly in Figure 2.3 the results of horizontal flattening can be seen.

2.2.4 Object Identification

Once the noise is filtered and the background is leveled the image is ready for object identification. The object identification service can be provided with a minimum area for an object as a threshold (filter the objects with lower area than provided threshold area). The area is measured in pixels. Particle analysis is accomplished by first converting the data from its original format into a binary representation where the

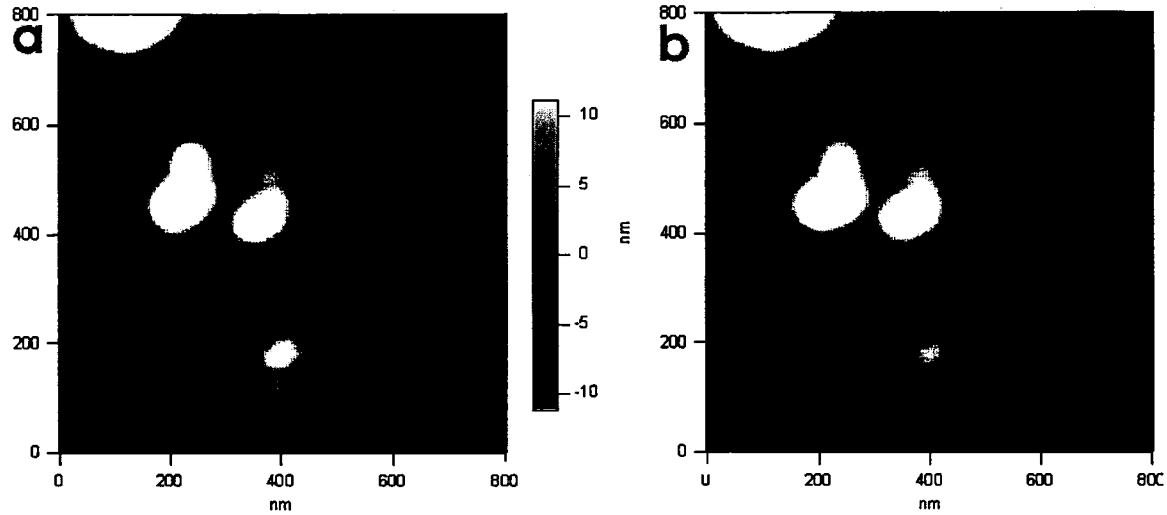


Figure 2.3: A Horizontal Flattening of image (a) (gold particles on a silicon surface) levels the background and results in image (b). This image ©Erika Merschrod, 2005. Used with permission.

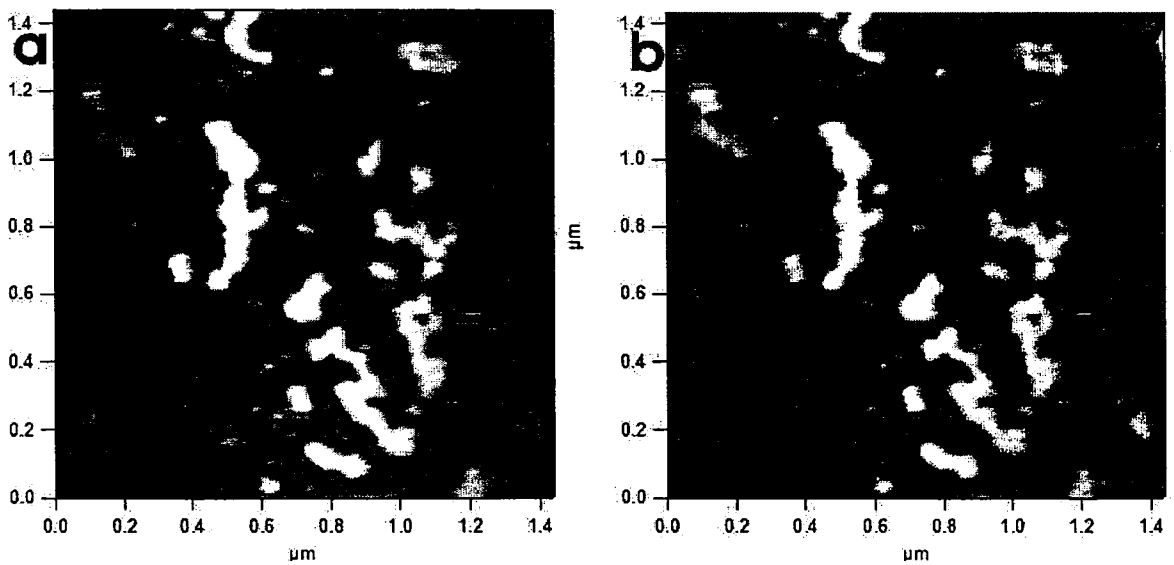


Figure 2.4: A Vertical Flattening of toposome (a type of protein) [13] image (a) levels the background and results in image (b). This image ©Ming Sun, 2005. Used with permission.

particle is designated by zero and the background by any non-zero value [11,14]. The algorithm searches for the first pixel or voxel that belongs to a particle by considering the difference of background and objects' pixel point values, then grows the particle from that seed while keeping count of the area, perimeter and count of pixels or voxels in the particle. Figure 2.5 shows the outlines of objects with red boundary lines. The image represents a film polymerized on a mica substrate.

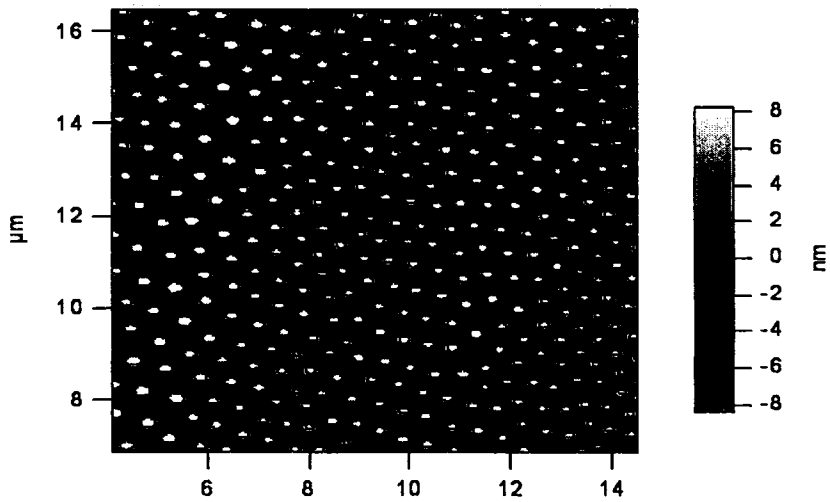


Figure 2.5: Particle analysis of image showing polymer grown on mica substrate. This image ©Erika Merschrod, 2005. Used with permission.

2.2.5 Object distribution in the image

One of our research requirements is to find how objects and structures are distributed in sample. To serve this purpose I wrote an algorithm for average distance finding. The algorithm tells us about the average distances between neighboring objects/structures. A special case of this algorithm is for anisotropic objects such as the

pattern shown in Figure 2.6. In this case, users of the system must select the image to find the objects distribution in it. The algorithm is then provided with an angle representing a line on the image. The algorithm draw lines perpendicular to each point of the angled line and calculates distances between objects on each perpendicular line. The algorithm can also optimize the user-provided line to provide more accurate distances. The result of the algorithm provides us with average distance between every two neighboring objects/structures. Figure 2.6 shows the input line

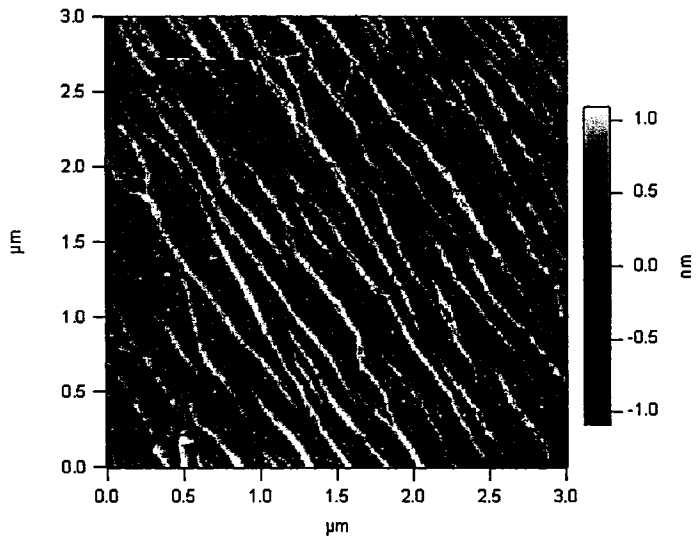


Figure 2.6: An image of a collagen pattern showing user defined input line a and one of the generated perpendicular lines b used to find average distribution/distances of anisotropic structures. This image ©Ming Sun, 2005. Used with permission.

provided by the user *via* a selected angle (45° in this case) as line a, with the line marked b representing one of the perpendicular lines.

2.2.6 Recording Results

The service of object identification outlines the objects in the AFM images. The next step is to calculate the desired statistics about the image and record them. The statistics about processed images, such as their objects' heights, sizes, and orientation, are recorded in graphical and textual formats. Sample graphical output is shown in Figure 2.7. A summary text file is also saved showing the image statistics as the number of objects in the image, their sizes, heights, rectangularity, standard deviation and kurtosis.

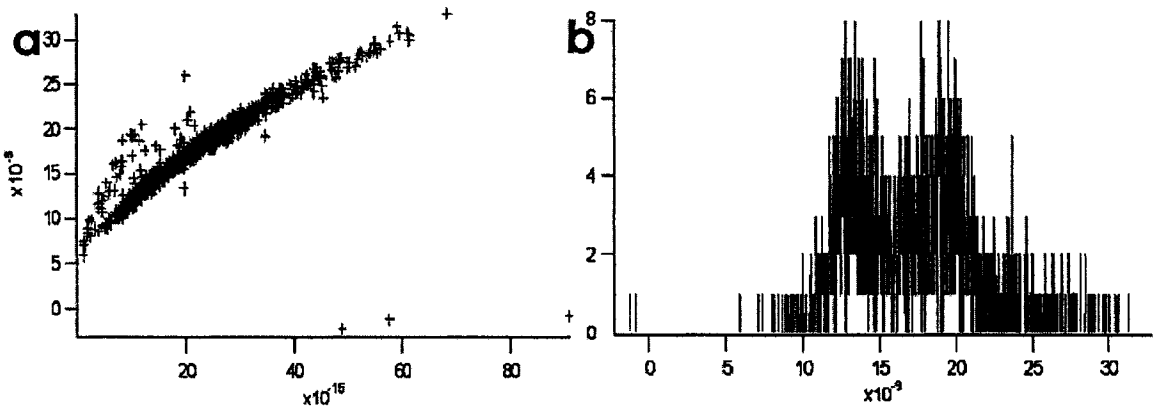


Figure 2.7: Graphical representation of the object statistics in the image from Figure 2.5 (showing polymer grown on mica substrate). (a) objects' areas aginst heights (b) objects' heights histogram.

The graphs in Figure 2.7 show that the distribution of polymers grown on a mica substrate is mostly uniform [15]. A quick glance at the Figure 2.7a also clearly tells us that three crosses with low heights and high areas must be scan line noise. Hence we can say these graphs tell us about noises and erroneous data present in images.

The text summary file contains information about the processed image: number of objects in image, average height and area, maximum and minimum height and area, standard deviation, skewness, and kurtosis of the objects' heights and areas.

Chapter 3

Image Processing

One of the major challenges for image processing is the reduction of data. Images normally have an enormous amount of data, from kilobytes to megabytes. In most cases much of this information is not required to solve an image processing problem. The first step of our processing determines exactly what information is necessary.

3.1 Layer Extraction

The layer extraction mechanism is a procedure of compressing the image for processing. The AFM images we use in our research represent the surface of materials, which are of the three-layer format [3]. These layers represent height, error in the feedback mechanism (which highlights edges) and elasticity/friction of the sample [16,17]. Each of the layers is an image, and the information we are trying to extract from the image are its objects'/structures' heights, area and distribution [3, 16–18].

All the information of interest usually can be extracted from the height layer only. Therefore, we reduce the image data into data representing the height layer of

the image. This is a simple process that can be performed by selecting the available service of layer extraction from the service menu for a specific image. Layer extraction speeds up the image processing and analysis process by about ten times.

However, sometimes the information in the height layer is not complete, and cannot be used to extract accurate information about the image. In such situations a "copy mask" service is used to complement the missing information of the height layer from any other layer. In the example given in Figure 3.1, the object boundaries are copied from an auxiliary layer (friction layer) and are pasted on top of the layer with data of interest (height layer). In the height layer in Figure 3.1b, the dark part of the image represents small heights and light regions corresponds to larger heights.

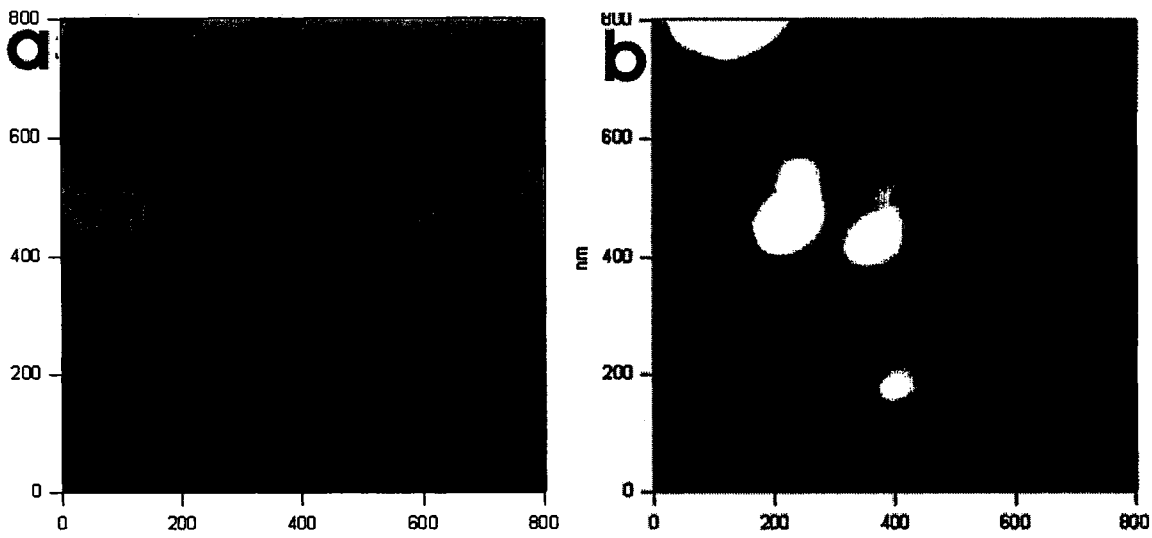


Figure 3.1: Copying Mask from Image (a) (friction map) to Image (b) (height map) identifies objects (such as that indicated by the arrow in (b)) which would not be seen in a mask generated directly from(b). Images represent gold particles on a silicon surface. This image ©Erika Merschrod, 2005. Used with permission.

Figure 3.1a, has clear objects, but Figure 3.1b is the layer favorable for processing. The copy mask service saves the boundaries of objects in Figure 3.1a and pastes those boundaries on the image (layer) in Figure 3.1b. Otherwise, small objects like the one with the arrow in Figure 3.1b would not appear in height mask. The copy mask service is provided in MFP-3D software (by wavematrics) but the idea of copying mask and pasting it on top of other layers to complement missing information was adopted during the research.

3.2 Noise Removal

Noise is any undesired information that contaminates an image. In typical images the noises can be categorized as Gaussian ("normal"), uniform, or salt-pepper ("impulse") noises. The sources of noise can be the object of focus and/or analog to digital conversion. The major sources of noise in AFM images are environmental noise and roughness of the support surface [12]. In our imaging process the support surface is very uniform so our images normally have environmental noises only. The image processing tool provides noise filtering services to get rid of the noise. The filters most often used for noise removal in this research are order filters (such as median filters), mean filters, and Fast Fourier Transformation filters [19].

3.2.1 Order Filters

These filters are based on the order statistic of an image. This filtering technique arranges all the pixels in sequential order, based on their value (mostly gray scale). The most useful of the order filters is median filter. The median filter starts with

a window of size N from the top left of the image and arranges the pixels within, by bringing the median valued pixels to the center of the window [20]. The window moves from left to right and goes to the second row after reaching the end of the first row. Figure 3.2 gives an idea of the working of median filter for N = 3.

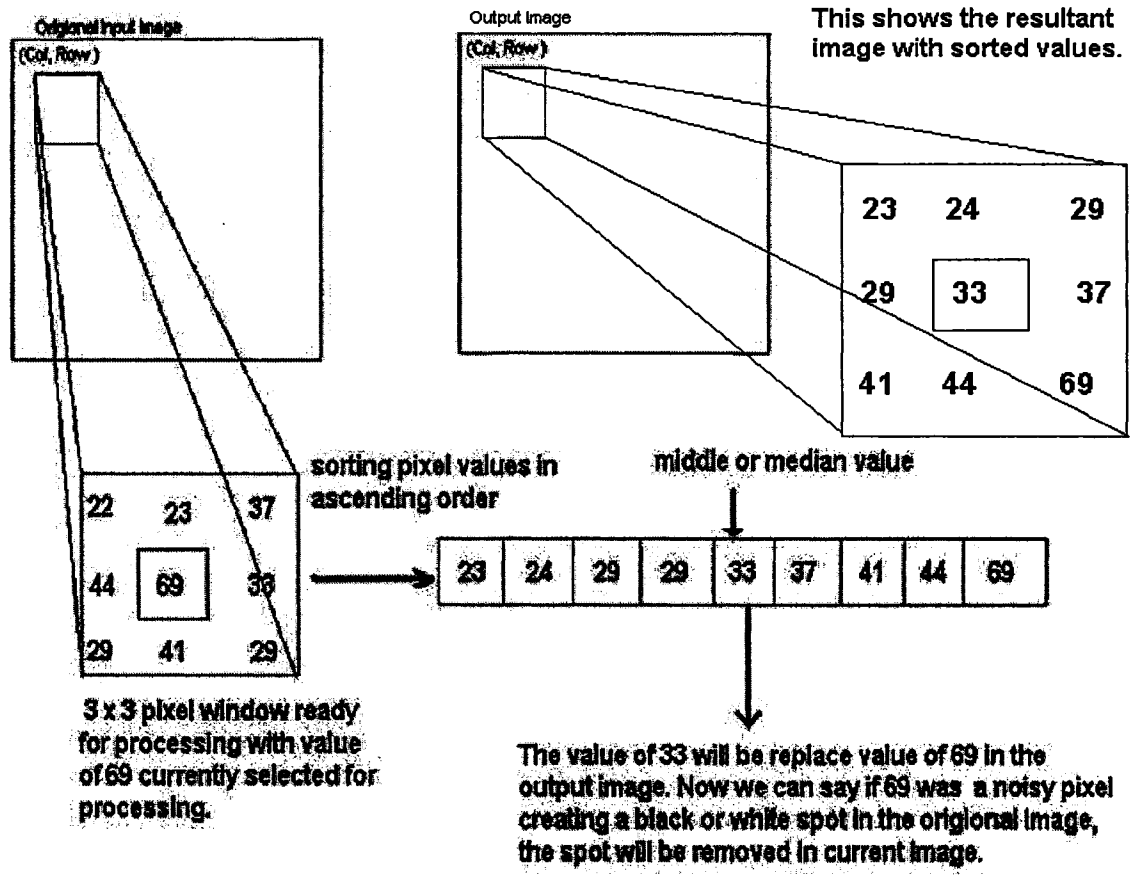


Figure 3.2: Moving window median filter example, inspired from image in thesis of Jason Waltman, Wittenberg University, 2001.

The median filter is effective in removing salt-pepper noises (black and white spots on an image). Figure 3.3 shows the application of the median filter on the image with

salt-pepper noise.

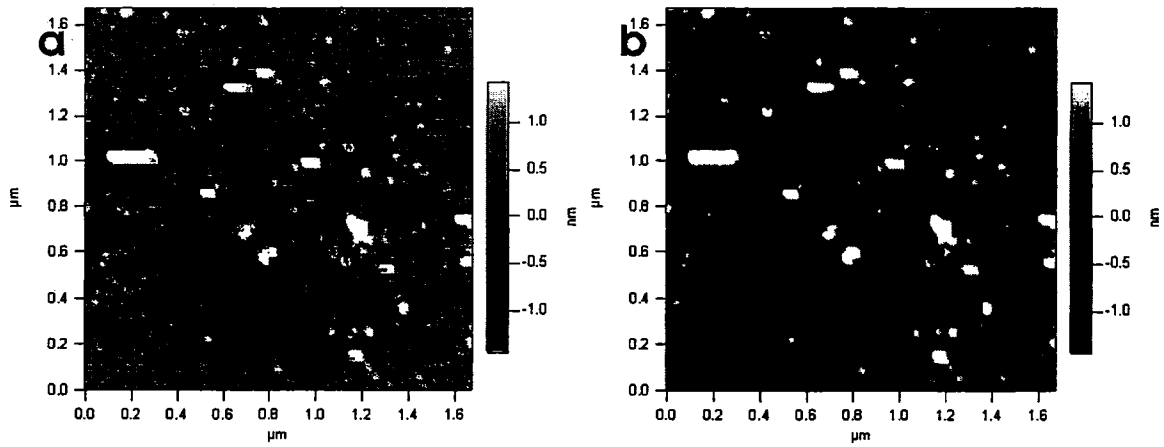


Figure 3.3: Applying median filters on image (a) an image with salt-pepper noise, removes salt-pepper noise and results in image (b). This image ©Ming Sun, 2005. Used with permission.

3.2.2 Mean filters

The mean filter is applied when the images have uniformly distributed noises [20]. The filter requires a window size N, and then the filter finds out some form of average within an N×N window, using the sliding window concept to process the whole image. The most basic of the mean filters is the arithmetic mean filter which finds the average of pixel values x_i in the window:

$$\overline{x} \equiv \frac{1}{N} \sum_{i=0}^{N} x_i$$

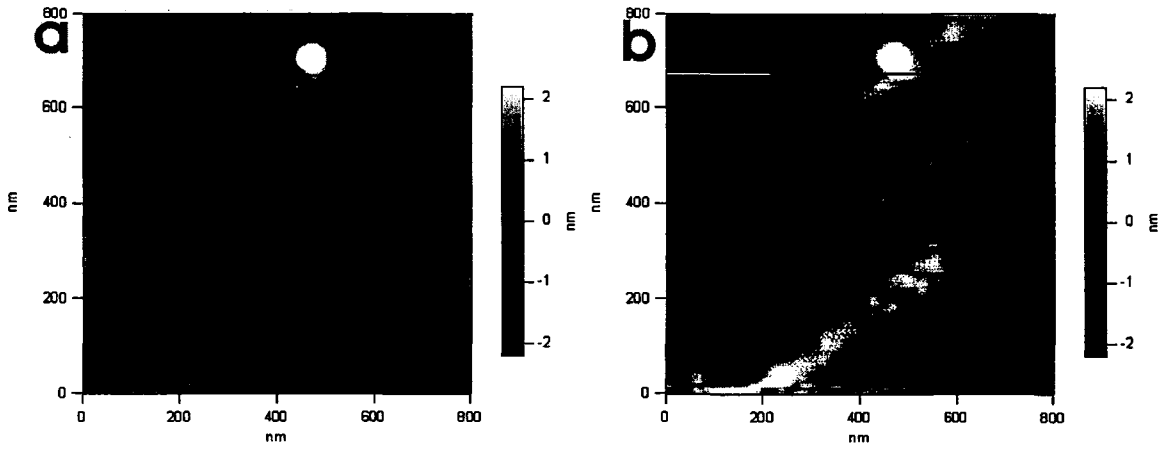


Figure 3.4: Applying mean filters on image (a) an image with uniform noise, removes noise and results in image (b). This image ©Ming Sun, 2005. Used with permission.

3.2.3 Fast Fourier Transformation Filter (FFT Filter)

The Fourier Transformation maps image data into frequency space through transformation equations. In our research we apply the commonly used "Fast Fourier Transformation" (FFT)) [19] on images to filter high frequency noises when periodic noise in the image is superimposed over the image pixel lines. We handle it by removing particular frequency ranges. Our image data (pixel points) numbers 2^n (e.g. 256x256, 512x512) and the FFT works efficiently with 2^n data points.

$$\text{Fast Fourier Transformation } I\ [t_1]\,[n] \equiv \sum_{i=0}^{N-1} f[t_1]\,[k]\, e^{\frac{\mathrm{i}2\pi \mathrm{kn}}{N}}$$

The FFT is applied on each column of original image, where I is the resultant transformed column for the original image column f. The images in Figure 3.5 show the same collagen fibers but the superimposed noise in figure 3.5a is removed in 3.5b using Fast Fourier Transformation [11, 21].

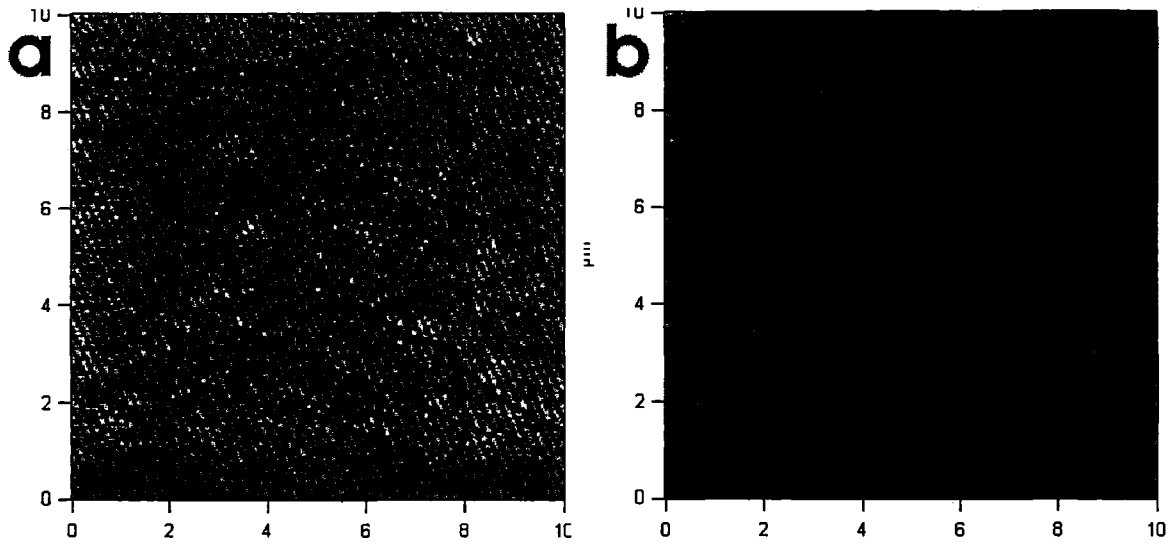


Figure 3.5: A fast-fourier transform (FFT) filter of image (a) removes high frequency noise and results in image (b). This image ©Ming Sun, 2005. Used with permission.

3.3 Flattening Image

One of our research goals is to find the exact information about structures in images, i.e. their heights, areas and distribution. To find the accurate heights, areas and distribution of structures/objects in an image, an even background is highly desirable [16]. Unluckily most of our images have the problem of an uneven background. Results have shown that there are cases when small objects fade out as compared to the high part of uneven backgrounds and in some cases the ups/heights of the background become objects or parts of objects. To get rid of this issue we have a service (algorithm) of background flattening in our working system. The service asks the user of the system to provide the type of algorithm suitable for the image at hand (vertical flattening algorithm or horizontal flattening algorithm). The user then inputs the best pixel line to provide the guidance for the algorithm. The algorithm

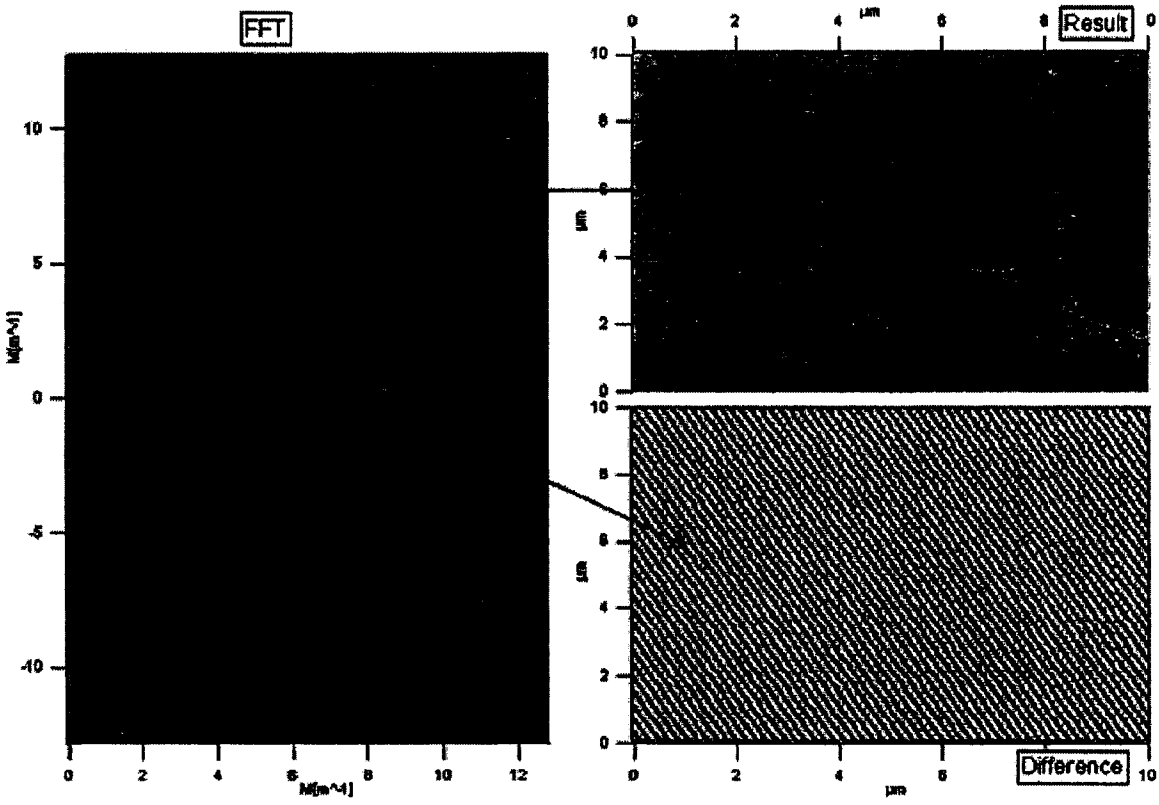


Figure 3.6: Application of FFT on the image of the collagen fibers where section with title "FFT" represents frequency , "Result" section shows the image after transformation, and "Difference" section represents the difference between the original and final images. The section of frequency that is filtered out is shown boxed with the arrow directing to difference image.

averages out unwanted heights and depths according to the user desired mechanism.

The step by step working of the flattening algorithm is as follows:

1. The user selects an image for flattening
2. The user provides the information about the type of flattening required for the current image (Vertical flattening / Horizontal flattening).
3. The user selects a pixel line in the selected image to guide the algorithm
4. The algorithm:
   (a) finds the average of the selected pixel line (row or column of image matrix),
   (b) calculates a vector representing the difference of average to the actual values of the selected pixel line (avoiding exception values), and
   (c) applies this difference value (average - pixel values) vector to the whole image data vertically or horizontally as required.

Our observations tell us that this algorithm leads to reliable results of true heights, areas and distribution of objects.

Figure 3.7a (equivalent to Figure 2.3) shows an image with an uneven background as there is a dark band (representing depth in the image) horizontally between 400 and 500 micrometers. Figure 3.7b shows the same image but the background is leveled by applying the horizontal background leveling algorithm.

Analogous to the horizontal flattening in Figure 3.7, Figure 3.9 (equivalent to Figure 2.4) shows the vertical flattening of the first image into the second image. Without this flattening, the central part of the background of the image in Figure

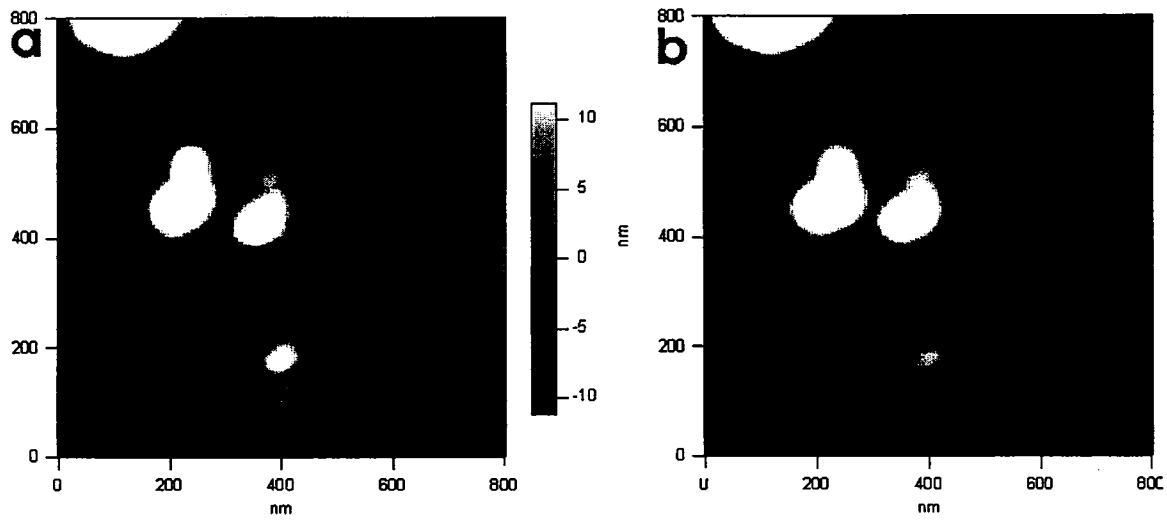


Figure 3.7: A Horizontal Flattening of image (a) (gold particles on a silicon surface) levels the background and results in image (b). This image ©Erika Merschrod, 2005. Used with permission. *(Equivalent to Figure 2.3.)*

3.7a is high (brightness represents height), so it will make a huge object in the center of the image.

An illustration with a simple (or small) matrix follows. As shown in Figure 3.8 Table 1 represents the image matrix, and the third column of image data is selected by the user as ideal for horizontal leveling. As higher values represent objects, we can clearly see there is an object or part of an object in our selected line. We have to skip the data points belonging to the objects and use the previous or next values instead. Then we calculate the difference of the average of the selected column and store it in Vector 1. Applying Vector 1 (adding to matrix vertically) to the image (Table 1) results in Table 2 (image with uniform background).

The algorithm is highly customized and can process huge amounts of data in minimal time. For example an image with data size 1028 x 1028 (image data matrix

11	13	12	11	12
13	7	198	7	10
11	10	12	9	10
7	5	6	8	6
9	10	8	9	7

Table 1:
Original Image Data

-2
-2
-2
4
2

Vector 2:
Normalization Vector

9	11	10	9	10
11	5	196	5	8
9	8	10	7	8
11	9	10	12	10
11	12	10	11	9

Table 2:
Image Data After Normalization

Figure 3.8: A demonstration of image leveling on image data. Table 1 represents origional Image data, Vector 1 represents user selected image line candidate for image leveling, and Table 2 shows normalized image data

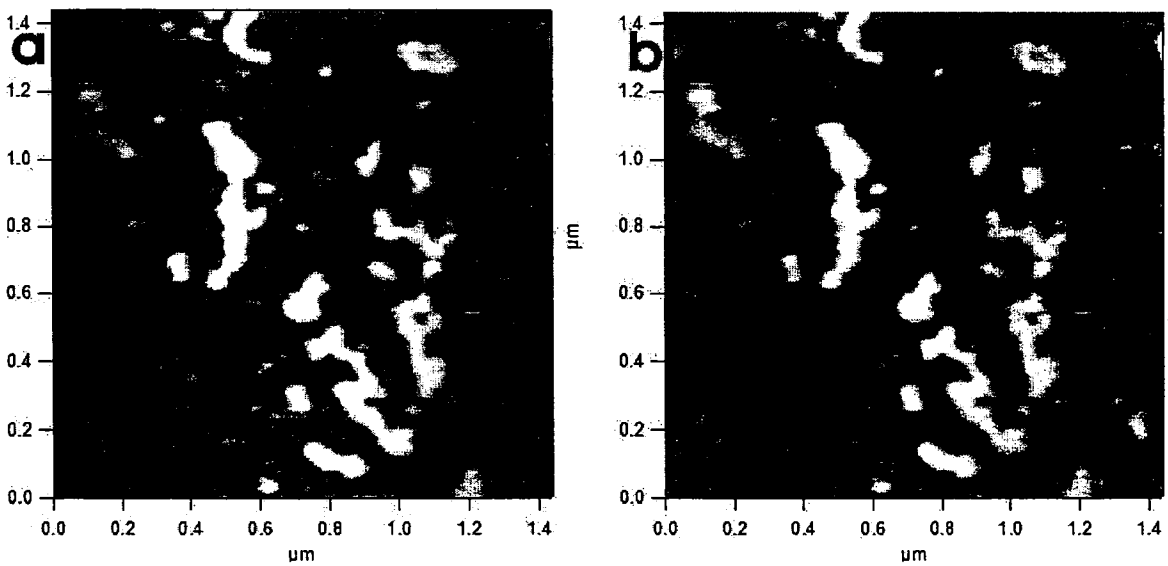


Figure 3.9: A Vertical Flattening of toposome (a type of protein) [13] image (a) levels the background and results in image (b). This image ©Ming Sun, 2005. Used with permission. *(Equivalent to Figure 2.4.)*

size) can be processed and flattened in a fraction of second (0.03 sec "processing speed") on a computer with an average speed processor. Furthermore, the algorithm is effective and leads to help in identifying true structures (the heights, areas and distribution of objects in an image) that were difficult to find without this algorithm.

3.4 Particle Analysis

Once the image is refined by removing all anomalies and/or noises it is ready for finding objects or structures. Particle analysis is accomplished by first converting the image data from its original format into a binary representation, where the particle is designated by zero and the background by any non-zero value [11,14]. The algorithm searches for the first pixel that belongs to a particle and then grows the particle from that seed while keeping count of the area, perimeter and count of pixels in the particle. The margin of error in object boudaries identification is approximatly 5 nm. The first particle pixel is calculated as follows:

1. Select an arbitrary threshold value to differentiate between background and objects of interest.

2. Create mask with object pixels set to 1 and background pixels set to 0.

3. Apply mask to obtain a flattened background.

4. Check correlation between images before and after mask, with optimal correlation defined as $\frac{image \times mask}{image} \approx 1$.

5. Adjust the mask based on correlation feedback and repeat from step 1 untill correlation is approximately equal to 1.

If you use additional flags the algorithm will compute additional quantities for each pixel that belong to the particle [11]. The particle analysis service provides the user with an interface to input the threshold for filtering out unwanted objects. The user can provide a minimum area (in term of pixels) to filter out the objects with areas less than provided the threshold value. The objects/structures exceeding the threshold value found in the image are numbered starting from the bottom left corner of the image. Figure 3.10 (equivalent to Figure 2.5) shows the outlines of objects with red boundary lines. The image represents a film polymerized on a mica substrate [15].

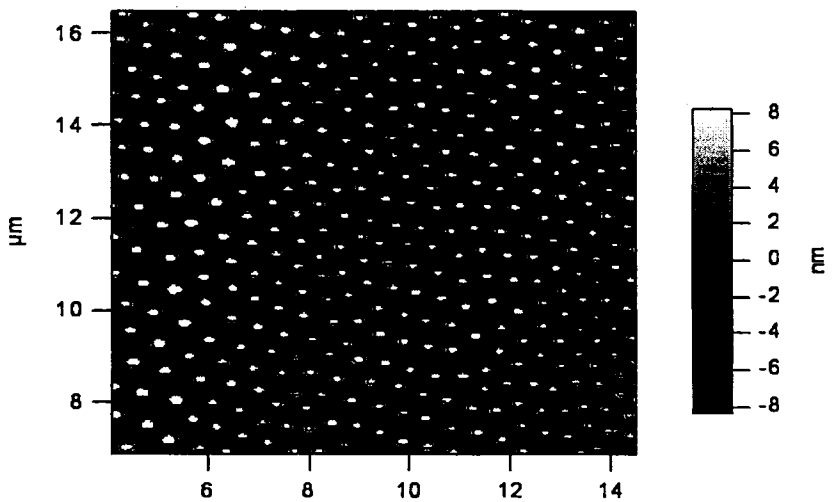


Figure 3.10: Particle analysis of image showing polymer grown on mica substrate. This image ©Erika Merschrod, 2005. Used with permission. *(Equivalent to Figure 2.5.)*

Once objects are identified and marked with object counters, the following values are calculated for each object.

1. Object area: the area (in pixels) for each particle as the number of pixels

inscribed in an object

2. Object perimeter: the perimeter (in pixels) for each particle. The perimeter calculation involves estimates for 45° edge that results in no-integer values.

3. Object circularity: the circularity is the ratio of the square of the perimeter to ($4 \times \pi \times$ object area). This value approaches unity (one) for a perfect circle.

4. Object rectangularity: the rectangularity is the ratio of the area of the particle to the area of the inscribing (non-rotated) rectangle. The ratio is $\pi/4$ for a perfect circular object and unity for a non-rotated rectangle.

5. The standard deviation, skewness and kurtosis of object areas and/or heights are calculated according to following formulae:

$$\text{Standard Deviation } \sigma \equiv \sqrt{\frac{1}{Vnpnts-1}\sum (Y_i - Vavg)^2}$$

$$Skewness \equiv \frac{1}{Vnpnts-1}\sum_{i=0}^{Vnpnts-1}\left[\frac{(Y_i-\overline{Y})}{\sigma}\right]^3$$

$$Kurtosis \equiv \frac{1}{Vnpnts-1}\sum_{i=0}^{Vnpnts-1}\left[\frac{(Y_i-\overline{Y})}{\sigma}\right]^4 - 3$$

where

Vnpnts: number of objects/structures.

Vavg: average of Y values. Where Y can be objects heights or areas

Standard Deviation: standard deviation of Y values

One of the desired results of our research is to calculate the height of each object. If we use the binary image it is not possible to calculate the heights of the objects,

so we process the original image data to find the highest point of each object in the image. In the original image data the pixel values of objects are positive values and the background of the image consists of negative pixel values [11]. Also, the bigger positive values represent higher parts of objects rather than smaller positive values. The algorithm devised to find the height of objects follows the same route for finding objects as that of the object marker *i.e.* it identifies objects and marks them with a number. The pseudo code for the algorithm calculating the heights/peaks of objects is as follows.

1. Start from bottom left of an image and move right searching for positive values.
2. When a positive value is encountered, the system checks if the positive value is higher than the threshold value for the previous object identification algorithm.
3. If the identified value belongs to an object/structure, increment the object counter.
4. Move in the direction of higher values either top, right or top right (45 ° from top).
5. Stop where the value is maximum from its next and previous values.
6. Write this value against the object counter as its peak value/height of the object.
7. Skip the positive value of this object and move right.
8. When a background pixel value (negative pixel value) is reached, move right and repeat the process from step 2.

9. When the end of an image is reached on the right side, the routine moves up by one pixel line and starts from left to move right and repeat the process from step 2.

10. Repeat from step 2 until the end of the image is reached.

This algorithm is fast and takes about 0.03 seconds to calculate heights of objects in an image with more than a hundred objects. The loops are managed effectively and are broken with a break statement on reaching the height for an object to avoid unnecessary iteration of loops. After applying this algorithm we have a summary table with the object counter and its height for a certain image.

ROC0			1.62
Point	heightVector		
14	1.75407e-08		
15	1.67151e-08		
16	1.74212e-08		
17	1.71302e-08		
18	1.75886e-08		
19	1.71293e-08		
20	1.88922e-08		
21	1.70735e-08		
22	1.84214e-08		
23	1.30375e-08		
24	1.62546e-08		

Figure 3.11: The output table shown in the screen capture here lists object counters (points) against their heights in meters for the image in Figure 3.10

Another feature of particle analysis is to find the distribution of objects. To see the distribution of the objects our system has two services:

1. Minimal distance calculation between neighboring objects

2. Average distances between structures in images

Our images normally have one of two different types of structures/objects. The first are regular or irregular circle-shaped objects (as shown in Figure 3.1and 3.11). The second are rod or rope like structures (as shown in Figure 3.12). The first service of distance calculation between neighboring objects is applied on images having circle like objects. Similarly, the service of average distance calculation is applied when dealing with the images having rod or rope like structures.

3.4.1 Minimal distance calculation between neighboring objects

This algorithm tells us how the objects are distributed in the image. The result of this algorithm is the average distance, the minimal distance, and the distances between any two neighboring objects. The result in this algorithm is calculated as the displacement between neighboring peaks. The pseudo algorithm for this service is as follows:

1. Start from bottom left of the image and search for object peak.

2. If a peak is encountered initialize the distance counter with zero.

3. Move right and search next peak and increment distance counter for every non-peak object pixel point or background pixel point.

4. If there is no background point between two peaks consider the peak with higher positive value as peak and adjust distance counter accordingly.

5. Repeat step 3 until reaching next peak.

6. On reaching second peak save the distance counter value in distance summary table and initialize counter with zero and go to step 3.

7. If right end of image is reached go to one level up.

8. Repeat from step 2, until the end (top right) of image is reached.

The result of this algorithm is a summary table with information about objects distribution.

3.4.2 Average distances between structures in images

This algorithm calculates the distances between the structures (rod or rope like structures) in AFM images. In this service the user provides an angle and the system draws a line with the user input angle from right-bottom of the image. Then the system draws perpendicular lines to the user defined line. The system then calculates the distances between structures on each perpendicular line and takes its average.

The result of this algorithm is average and minimal distances between structures for current image. The distance is separation between the peaks of structures. The pseudo algorithm for this service is as following:

1. User inputs an angle to draw a line from right-bottom of the image

2. The system draws lines perpendicular on the user input line

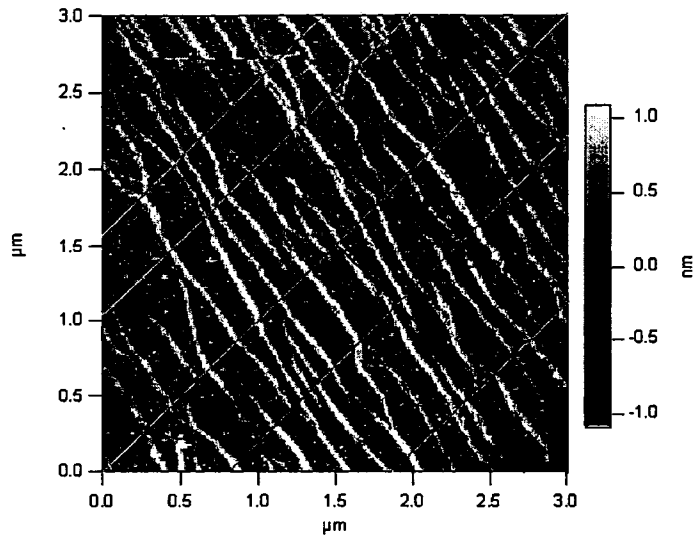


Figure 3.12: Collagen image having red line showing user input angle line and sea green lines are perpendicular lines to user input angled line. This image ©Ming Sun, 2005. Used with permission.

3. The system starts to find objects on perpendicular draw line starting from top-left of top-right according to distribution of lines

4. If a structure is found on perpendicular line find and record its distances

5. Go to next perpendicular line below processed

6. Go to step 4, until the right bottom or left bottom of image is reached

7. Calculate the average of recorded distances between structures

This algorithm tells us exactly how structures are distributed in some images. We can see the effect of external factors on some material by imaging material and then applying the external factor (such as temperature change, adding some chemical solution) on the same material and image it again. If we look at the distribution

before and after change we can easily say how objects behave they converge, diverge, grow or shrink because of change. Furthermore, when a user provides the system with some angle (i.e. 30 °), the system calculates the distribution of structures for a range of [input ° +2, input ° -2] (28-32°) and the results show what angle gives more accurate results for distribution of structures.

3.5 Extension for non-Igor Images

Our system was designed to process Igor images but it has the ability to process specific SEM (Scanning Electron Microscopy) images with minimal interaction from the user. The first step is to import the image to Igor-MFP 3D and then ask the user to set the scale for the image. The user of the system provides the size of one pixel for the SEM image he/she wants to process. The system then converts the image dimension to $2^n \times 2^n$ dimension. Now the image can be treated as an AFM image, and be processed to find the areas of objects/structures in the image.

First of all an image is imported to Igor environment and scaled as shown in Figure 3.13 b. Then we can now process it, like Igor native images as explained in previous sections.

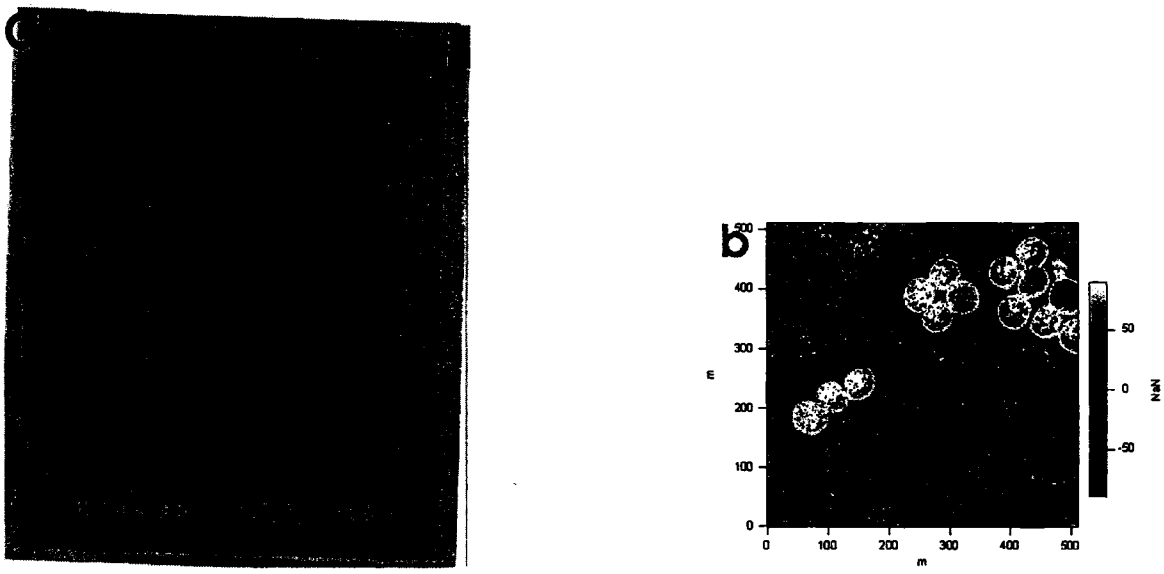


Figure 3.13: SEM images a, shows raw SEM image and b, shows same images scalled and ready for processing. This image ©Anand Yethiraj, 2005. Used with permission.

Chapter 4

Results and Discussion

Our research focuses on automating the analysis of physical changes occurring in a variety of proteins and polymers. We record statistical and graphical information of the sample material before and after the change. By comparing statistical data and graphical information, we deduce the exact amount of change (in size, shape, distribution and/or aspect ratio) occurring in the sample material. Furthermore, we discuss the structure of different sample materials in a variety of chemical compounds. The type of materials and their detailed structural analysis are given below.

4.1 Toposome

In this section we present the results of an analysis of calcium binding to toposome. Toposome is generally found in the cytoplasm of the sea urchin egg and embryo and is thought to repair damage in the plasma membrane [22]. In Figure 4.1 we image the protein toposome, and then add calcium to the same sample, and image it again.

The statistical results of surface images (as in Figure 4.1) of (4.3×10^{-06} mg/ml

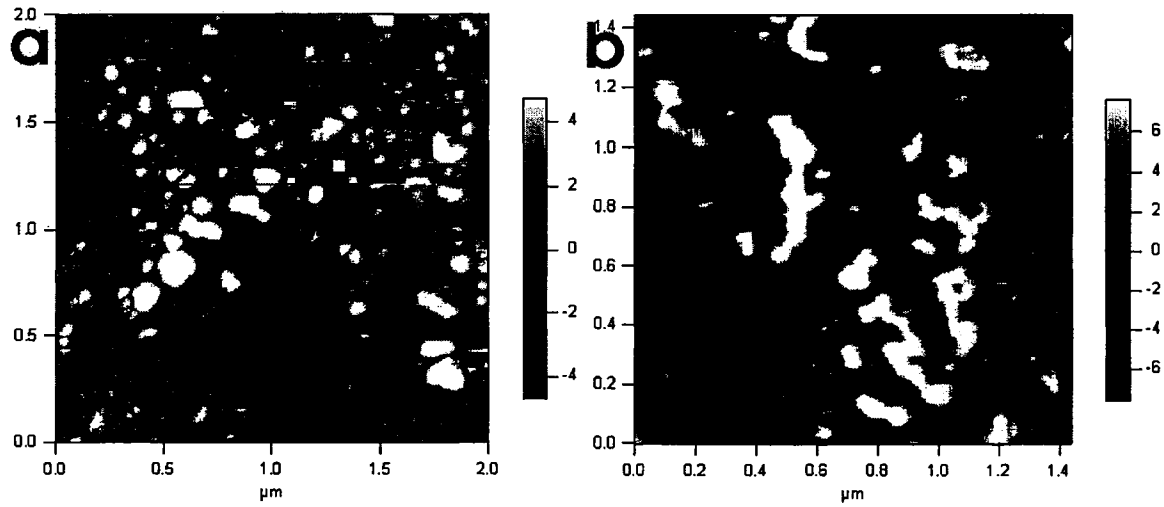


Figure 4.1: Toposome Images a)without Calcium solution b) with Calcium solution. This image ©Ming Sun, 2005. Used with permission.

concentration) toposome in 100μM calcium solution (Figure 4.1b) and without the calcium solution (Figure 4.1a) show that in calcium solution the toposome aggregates significantly, perhaps as a repair mechanism for the egg. The average area of objects in calcium solution increased by an amount 7.19×10^{-16} m^2 (or 719 nm^2) (about 28%), that is from 2.542×10^{-15} m^2 (or 2542 nm^2) to 3.261×10^{-15} m^2 (or 3261 n m^2). On the other hand, the effect on height is small as the increase in the average height is approximately by 5%, which is from 4.6×10^{-9} m (or 4.6 nm) to 4.8×10^{-9} m (or 4.8 nm).

The number of particles in the higher concentrated calcium solution is smaller so that it appears that particles aggregate laterally to make bigger particles. Also, a smaller increase in height shows that particles do not aggregate on top of other particles. Furthermore, the rectangularity of the objects is increased by 12.23% in calcium solution. Rectangularity is the ratio of the area of the particle to the area of

the inscribing rectangle. This suggests aggregation resulting from a structural (shape) change in the protein.

Results of this experiment leads to the observation that toposome undergoes structural changes by adding the mentioned concentration of calcium. The uncertanities/errors in area and height calculation will be due to error in the selection of threshold value for the selection of object pixel and background pixel. Our observation tells us that different concentrations of calcium lead to different types of changes. According to J. J. Robinson [22], with low concentrations of calcium the toposome undergoes a secondary structural change which facilitates protein binding to membranes. As increasing amounts of calcium bind, the toposome undergoes a change in tertiary structure which helps this protein to cause or drive membrane-membrane interactions. These interactions are required for a number of biological processes during development.

The Figure 4.2 is the same toposome sample as in Figure 4.1 but with a higher concentration of calcium (500μM concentration). As is clear from the image, the higher concentration of calcium introduced a great amount of change in structure and distribution of toposome [23]. The toposome objects adopted different type of shapes, *e.g.* circular, rectangular and horse shoe types. Our system is unable to tell us about the exact amount and type of changes occurring due to the higher calcium concentration because the structures in the image adopt a variety of shapes and it is difficult to characterize the smaller objects in a field of larger ones.

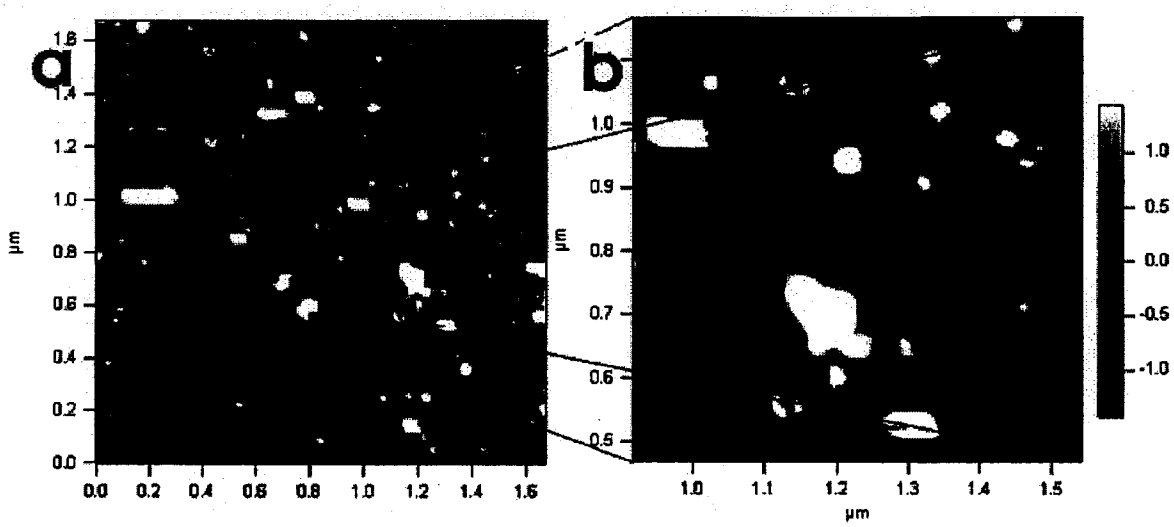


Figure 4.2: (a) Toposome image with higher calcium concentration, (b) zooming view to show clear view of horseshoes, irregular and regular structures. This image ©Ming Sun, 2005. Used with permission.

4.2 Polymers

The growth of polymer on a mica substrate is the focus of the polymer discussion and results [24]. After growing a polymer film on a mica substrate, the sample is imaged using AFM. Observation of graphical and numerical results of Figure 4.3a tells us that polymer growth on the mica substrate is quite uniform. The image on the right, Figure 4.3b, shows some uneven polymerization on a larger scale possibly due to local variation in polymer film thickness and evaporation rate. Below we show the statistical details for the image on the left.

The statistical graphs in Figure 4.4 left and right show that the distribution of polymer nano-particles grown on a mica substrate is mostly uniform with a tight bimodal distribution.

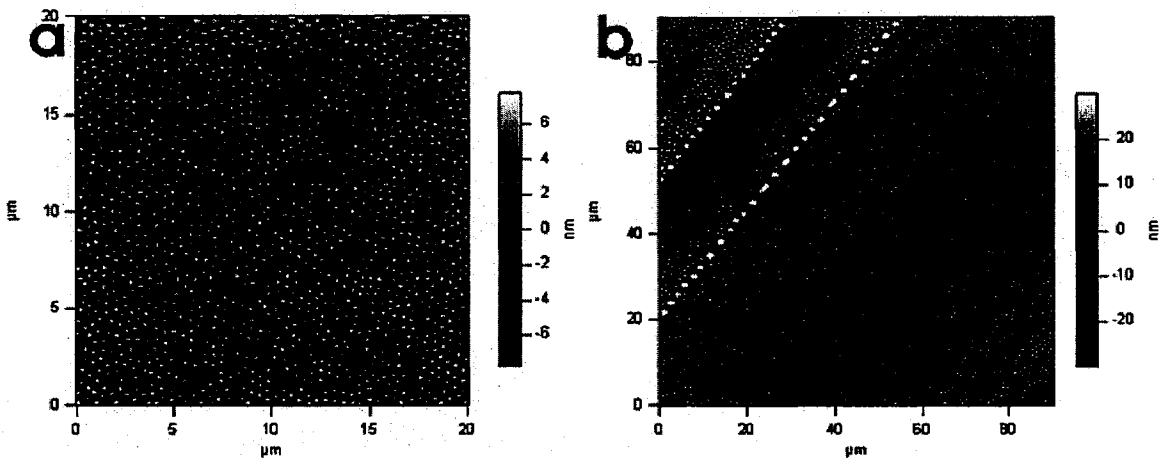


Figure 4.3: Result of even polymerization shown in image (a) and uneven polymerization in image (b) of film on mica substrate. This image ©Erika Merschrod, 2005. Used with permission.

Polymer particles with areas of more than $5.0\times10^{-14}m^2$ (or 50000 n m^2) could be quite high and should certainly be non-zero. Hence these graphs isolate noise and erroneous data present in the images. Also, the graph tells us that polymers on a mica substrate grow taller with increase in their area i.e. particles with area in a range of 0-4.0 $\times10^{-14}m^2$ (or 0-40000 nm^2) have a height of $5\text{-}20\times10^{-09}$m (or 5-20 n m) but when the area is $5.0\text{-}6.0\times10^{-14}m^2$ (or 50000-60000 n m^2) the object's height ranges $28\text{-}34\times10^{-09}$m (or 28-34nm). So the growth in area and height of polymers on mica substrate is directly proportional. A quick glance at the Figure 4.4 also clearly tells us about outliers in data. Marked by the circle are three particles with a large area but very small height, indicators of scan-line noise.

The statistical summary file lays out statistical information in tabular form. As the table in Figure 4.5 shows, the object growth for the polymer sample is uniform. The average area appears to be the midpoint between maximum and minimum area

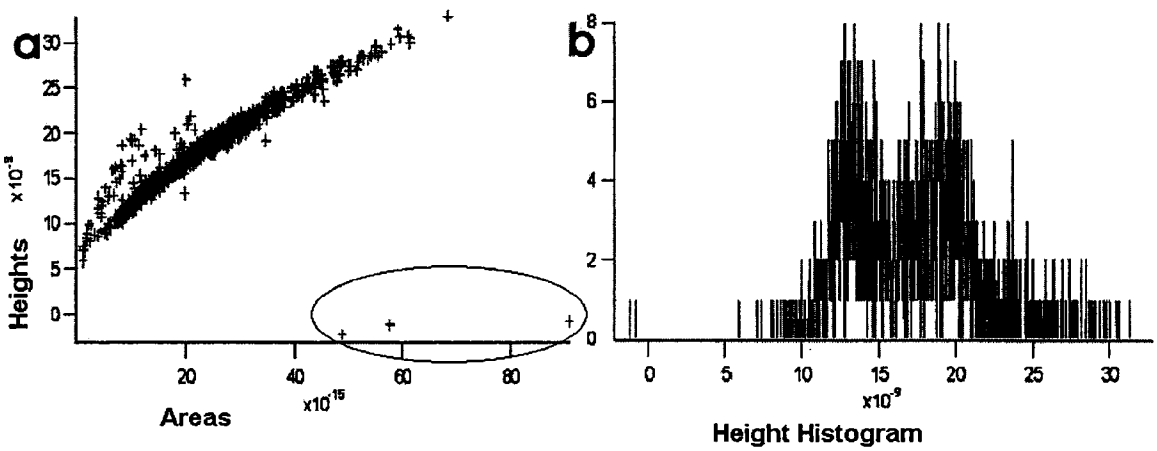


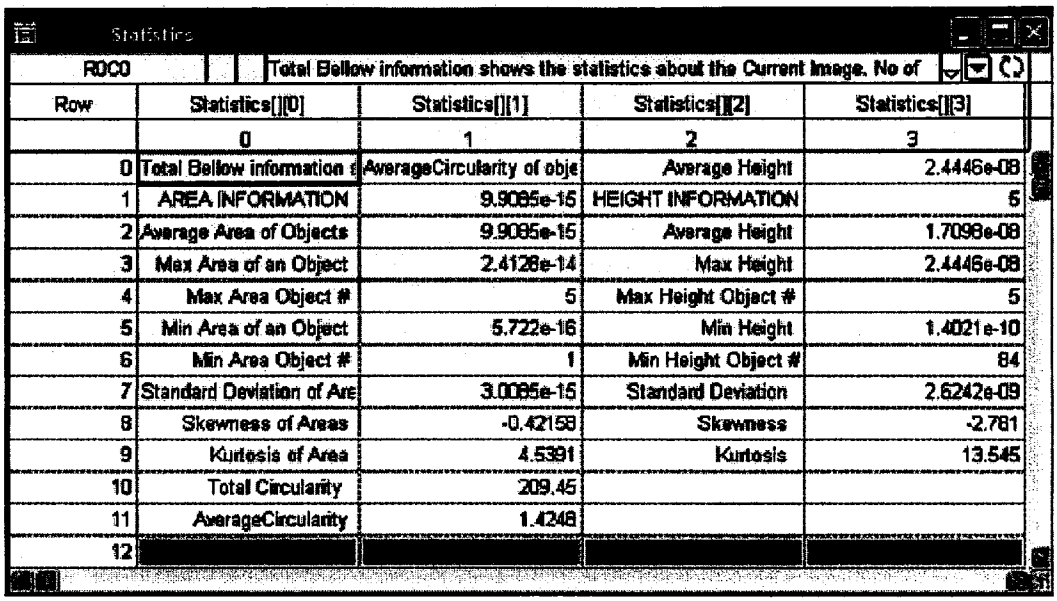
Figure 4.4: Graphical representation of Image (showing polymer grown on mica substrate) statistical results a) objects areas aginst heights b) objects height histogram.

objects. The table also shows that the heights of the objects is mostly uniform, the maximum height of any object being very close to the average heights.

4.3 Collagen Microfibrils

Collagen is one of the most abundant proteins in animals' bodies, and collagen fibrils are the most important tensile reinforcing element in animals' tissues [25]. Among more than 20 different types of monomers, Type I collagen is the most abundant one found in animals and is also a focus of research [25, 26]. The size and shape of the fibrils are important to determine tissues' functions since the hierarchical spatial arrangement of collagen fibrils can dominate the mechanical and physical properties of tissues. In our research we study the shape and distribution of collagen micro fibrils.

From Figure 4.6, it is obvious that collagen makes rope-like structures. The above



Statistics

ROI0 | Total Bellow information shows the statistics about the Current Image. No of

Row	Statistics[][0]	Statistics[][1]	Statistics[][2]	Statistics[][3]
	0	1	2	3
0	Total Bellow information s	AverageCircularity of obje	Average Height	2.4446e-08
1	AREA INFORMATION	9.9085e-15	HEIGHT INFORMATION	5
2	Average Area of Objects	9.9085e-15	Average Height	1.7098e-08
3	Max Area of an Object	2.4128e-14	Max Height	2.4446e-08
4	Max Area Object #	5	Max Height Object #	5
5	Min Area of an Object	5.722e-16	Min Height	1.4021e-10
6	Min Area Object #	1	Min Height Object #	84
7	Standard Deviation of Are	3.0085e-15	Standard Deviation	2.6242e-09
8	Skewness of Areas	-0.42158	Skewness	-2.781
9	Kurtosis of Area	4.5391	Kurtosis	13.545
10	Total Circularity	209.45		
11	AverageCircularity	1.4248		
12				

Figure 4.5: Statistical summary for the image in Figure 4.4

image is from a collagen solution of 2 mg/ml concentration. To find the exact spatial distribution of these fibers in the sample, we calculated the average distances of neighboring structures. For the image in Figure 4.6, the average distance between neighboring structures is 8.8×10^{-08}m (88 nm). The average distances help us calculate the structure and structural changes at a different solution and for different concentration of collagen.

The images in Figure 4.7 show that collagen at higher concentration grows more structures (not taller structures) than at low concentration. So the distribution of structures of collagens can be controlled by increasing or decreasing the concentration of collagen in the sample solution used for experimenting. The average distance between neighboring structures for image 4.7a is 4.65×10^{-07}m (or 465 nm). When the concentration was increased from 0.3 mg/ml to 1 mg/ml the average distance between neighboring structures reduced to 1.29×10^{-07}m (or 129 nm) and for concentration 3

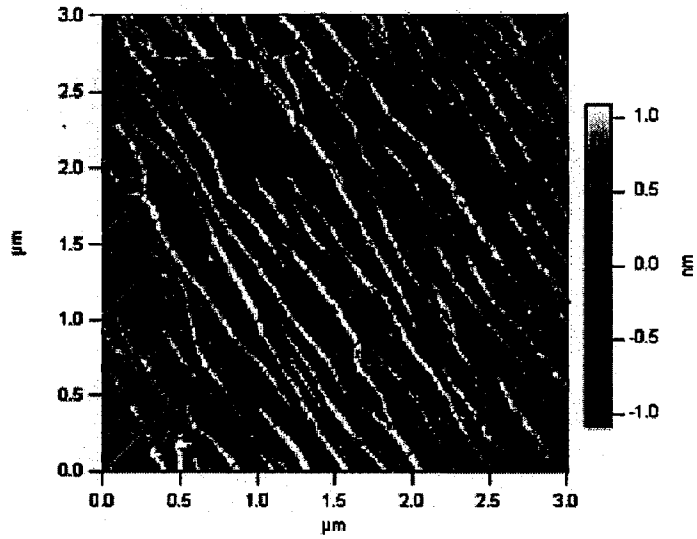


Figure 4.6: Collagen micro fibril image. The red and sea green lines are user defined lines drawn to calculate the distribution of collagen structures in sample. This image ©Ming Sun, 2005. Used with permission. *(Equivalent to Figure 3.12.)*

mg/ml as shown in image 4.7c, the average distance was 8.6×10^{-08}m (or 86 nm).

4.4 Limitations of the System

Our system is capable of handling and processing a variety of AFM images. However, during our experimentation we were unable to process images of the gold sample with a noise superimposed over the objects. The nature of noise superimposed over particles is different than previously mentioned noises, as it is a signal of noise superimposed on top of objects. We have objects of different heights, so in case of applying a threshold on height to get rid of the noise, that will either cut the tops from taller objects or it will leave noise on top of objects with smaller heights. For example, we tried to filter the high frequency noise using Fourier transformation but

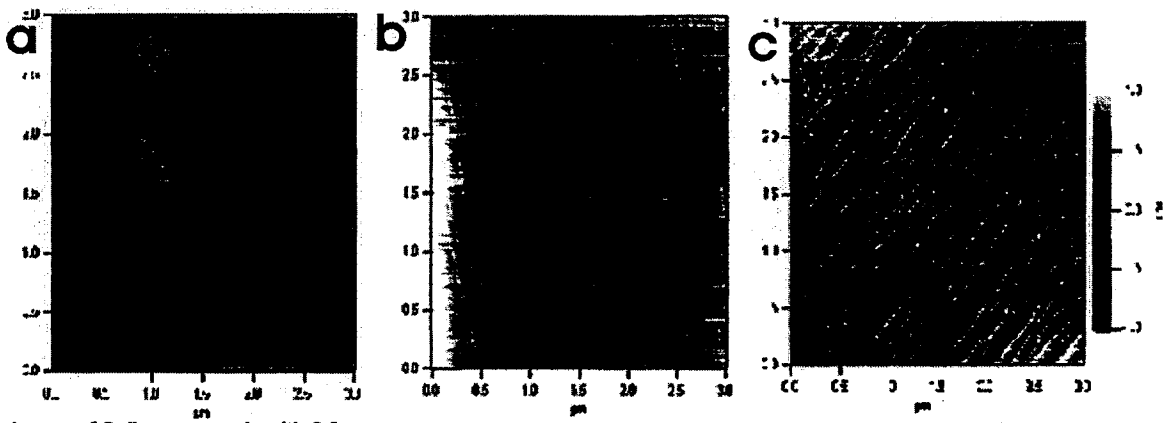


Figure 4.7: Images of Collagen at different concentration with calcium. This image ©Ming Sun, 2005. Used with permission.

the transformation left us with a distorted image as shown in Figure 4.8b.

In future we can design specific filters for the problem in hand to get rid of this type of noise. For example, we could ask the user about maximum height (threshold value) of objects/structures in the image and then set all the pixel values higher than user defined threshold to threshold values.

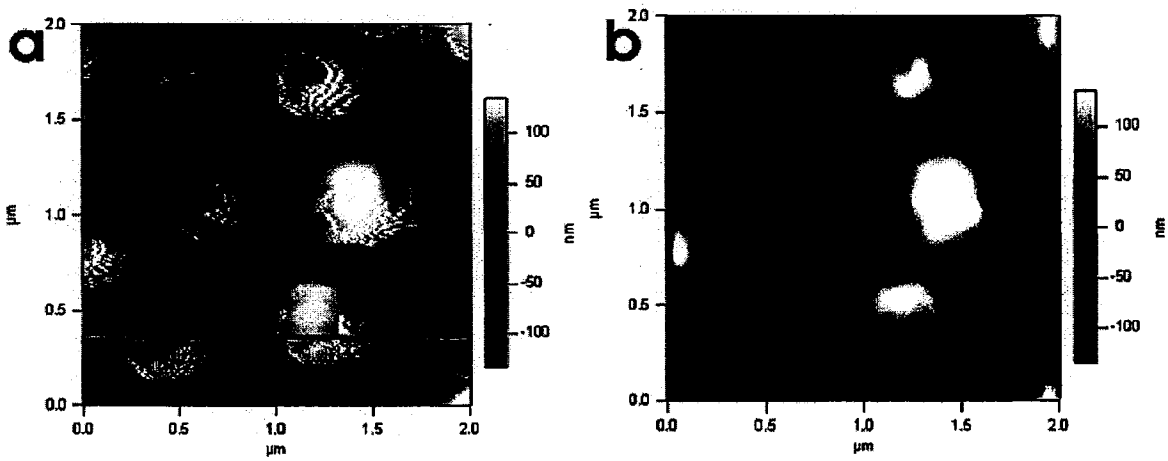


Figure 4.8: (a) Gold particle on silicon surface with noise lying on top of particles. (b) The same image after applying FFT (Fast Fourier Transformation) filter. This image ©Erika Merschrod, 2005. Used with permission.

Chapter 5

Conclusions and Future Direction

The developed system is a combination of robust and user-friendly services provided by Igor-MFP3D or by specially written code customized to serve our purpose, *i.e.* image analysis for nano-scale bio-chemical images. A very user-friendly interface is provided in the form of menus or option boxes.

The major focuses of the services I developed are accuracy of information extraction, speed of processing, user friendliness and robustness. Our system tells us about the exact amount of change occurring if we increase concentration of certain solution or add other materials to our solution. The accuracy of results is confirmed by comaparing with manual calculations and already-present information. Our system processes the images of the sample before and after the change and then generates a statistical summary table or graph indicating the exact amount of change occurring in the sample structure.

One of the key features of our system is effective and faster processing of images with huge amounts of data points. In our research, the images we need to process have

multiple layers with a variety of information associated with each layer. To process the images faster, we extract the layer of relevant data and process it. This layer extraction mechanism increases our processing capability by a large degree. In cases where the extracted layer does not have all of the desired information, we complement the missing information by adding to it one of the other layers of the image. The idea of layer extraction and subsequent correlation for processing the AFM images was adopted in this research and was not available previously. We observe that this technique decreases our processing time more than threefold.

Furthermore, our system is designed to a higher degree of user friendliness and is robust to use. All the services provided are easy to use with a click of the mouse either on a menu or in a user option dialog box. The system functionality and usability is explained through screen-capture images in the user manual. The system is design to respond consistently and intelligently to undesirable user inputs. For example, if the user provides nothing where the system asks for the file name, or if the file provided by the user does not exist, the system will ask the user again to provide the correct file name.

Our system is designed for AFM images, but it is also capable of processing non-AFM images. A few user-defined services can import, scale and process SEM images in a semi-automated fashion. Future directions for this research would be to make the system flexible to support and process all types of images. In my opinion, our system can easily process optical microscope images if the already existing services are tuned accordingly. This could be particularly useful for fluorescence images where you stain different parts of your sample with different coloured fluorescent tags. These could be thought of as different layers for your sample: one red, one green, one blue. We

can use our image processing techniques to compare statistics between layers to check for co-localization of various parts of the sample by providing new interfaces and by tuning and automating already existing services.

Another potential area of future research will be to design filters for images with low-frequency noise superimposed over the object signal. The images of gold particles on a silicon surface have this type of noise. We can get rid of problems by asking the user to enter the threshold value for the maximum height of objects/structures for an image and then mark all the pixel higher than threshold with the threshold.

Furthermore, currently our image processing system is unable to categorize statistical information about objects or structures in images with multiple types of structures/objects in a single image. As a next step, we can design services that can take care of this issue, *i.e.* to find exactly how many objects are of what shape (circular, rectangular, horse shoe, rope like) and exactly what kind of changes occur in these objects/structures due to changes in the sample. To find the shapes of objects in automated services, a comparative analysis of area and cross-section will be extremely helpful.

In conclusion, our highly automated and effective system finds statistical information about objects/structures in bio-chemical material samples. Also, the system records any changes which occur in these samples due to the addition of any other material or any change in concentration or temperature.

Bibliography

[1] Umbaugh, S. E. *Computer Imaging: Digital Image Analysis and Processing;* CRC Press: Prentice Hall PTR, Upper saddle river, NJ 07458, 2005.

[2] Danaher, M.; Hopkins, L. *International Symposium for Information Science* **2002,** *8,* 79-83.

[3] Binnig, G.; Quate, C. F.; Gerber, C. *Physical Review Letters* **1986,** *56,* 930-934.

[4] Liou, S. P.; Jain, R. C. **1990,** *1,* 201-203 Digital Object Identifier 10.1109/ICPR.1990.118091.

[5] Barrett, H. H. *Optical Society J. Opt* **1990,** *7,* 1266-1278.

[6] Yang, X.; Lo, C. P. *Photogrammetric Engineering and Remote Sensing* **2000,** *66,* 967-980.

[7] Samal, A.; Ivengar, P. A. *Pattern Recognition* **1992,** *25,* 65-77 ISSN:0032-3203.

[8] Warfield, S. K.; Kaus, M.; Jolesz, F. A.; Kikinis, R. Tensor Controlled Local Structure Enhancement of (CT) Images for Bone Segmentation. In *Medical Image Computing and Computer-Assisted Intervention,* Vol. 1496; Springer Verlag: Boston, MA, 1998 ISSN:0302-9743.

[9] Osuna, E.; Freund, R.; Girosi, F. *IEEE Computer Society Conference on Computer Vision and Pattern* **1997,** p. 130.

[10] Douglas, T. R.; Solmon, S. C.; Purdy, G. M. *Journal of Geophysical Research* **1994,** *99,* 135-157 ISSN:0148-0227.

[11] WaveMetrics, Inc., WaveMetrics, Inc Lake Oswego, OR 97035, USA "Igor Pro Manual", 5.03 ed.; 2004.

[12] Elisa, F.; Luca, B.; Macii, E.; Zuccheri, G. *Information Technology in Biomedicine* **2005,** *9,* 508-517.

[13] Mayne, J.; Robinson, J. J. *Biochem Cell Biol* **1998,** *76,* 83-88.

[14] Software for Image acquisition and Processing by Asylum Research, http://www.asylumresearch.com.

[15] Zhou, N.; Merschrod, E.; Zhao, Y. *J. Am. Chem. Soc* **2005,** *127,* 14154-14155.

[16] Almqvist, N.; Bhatia, R.; Primbs, G.; Desai, N.; Banerjee, S.; Lal, R. **2004,** *86,* 1753-1762.

[17] Giessibl, F. J. *Reviews of Modern Physics* **2003,** *75,* 949-983.

[18] "Atomic Force Microscopy", http://www.bfrl.nist.gov/nanoscience/BFRLAFM.htm.

[19] Fisher, R.; Perkins, S.; Walker, A.; Wolfart, E. "Fourier Transformation: Image Processing Learning Resources", 2003 http://homepages.inf.edu.ac.uk/rbf/HIPR2/hiprtop.htm.

[20] Moore, M. S.; Mitra, S. K. *Nonlinear Image Processing Proceeding of SPIE* **1999,** *3646,* 56-66.

[21] "Igor Pro", Software by wavematrics, IGOR Pro is an extraordinarily powerful and extensible scientific graphing, data analysis, image processing and programming software tool for scientists and engineers.

[22] Mayne, J.; Robinson, J. J. *FEBS Journal* **2005,** 272.

[23] Bowman, C. N.; Chair, P. "Division of Polymer Chemistry ABSTRACTS", 224th ACS National Meeting, Boston, MA, 2002 Abstracts.

[24] Biscarini, F.; Zamboni, R.; Samori, P.; Ostoja, P.; Taliani, C. *The American Physical Society* **1995,** *B 52,* 14868-14877.

[25] Kielty, C. M.; Grant, M. E. *Connective Tissue and its Heritable Disorders* **2003,** *2003,* 159-221.

[26] Batge, B.; Notbohm, H.; Diebold, J.; Hartwig,; Lehmann,; Bodo, M.; Deutzmann, R.; Muller, P. K. *European Journal of Biochemistry* **1990,** *192,* 153.

Appendix A

User Manual

A.1 Introduction

Nanotechnology is the one of the fastest growing research focuses for scientists of various interests. State of the art devices are used to observe the tiny (nano-scale) structures and their behaviors. We can apply this technology to study the systems of biological or chemical interest. One question we often ask is, how these structures reacts/responds to different environments and situations? We can answer this question using Atomic Force Microscopy.

Atomic Force Microscopy (AFM) is used to measure three dimensional nanoscale objects. It provides data in the form of 2-D arrays (3-D images), that is clear enough to be observed with human eye, and easy for analysis and processing. We use AFM to collect data on sample surfaces. Our samples have atomic or molecular structures and our images show their response to temperature changes, different solutions, and other physical parameters.

The first step is to scan the sample's surface area using AFM. This process give us AFM images. These images are then processed in Igor [21] (a mathematical package) to collect information. Igor provides an extremely large set of operations for the analyzing and processing of images. Our focus is specific and cannot be served directly by native Igor operations alone; therefore we need extra image analyzing and processing services within the Igor environment to serve our purpose. These services are named as Igor "Add-ons". Some of these are provided by Asylum Research [14] (the AFM manufacturers). These services are mainly responsible for getting statistics about scanned images.

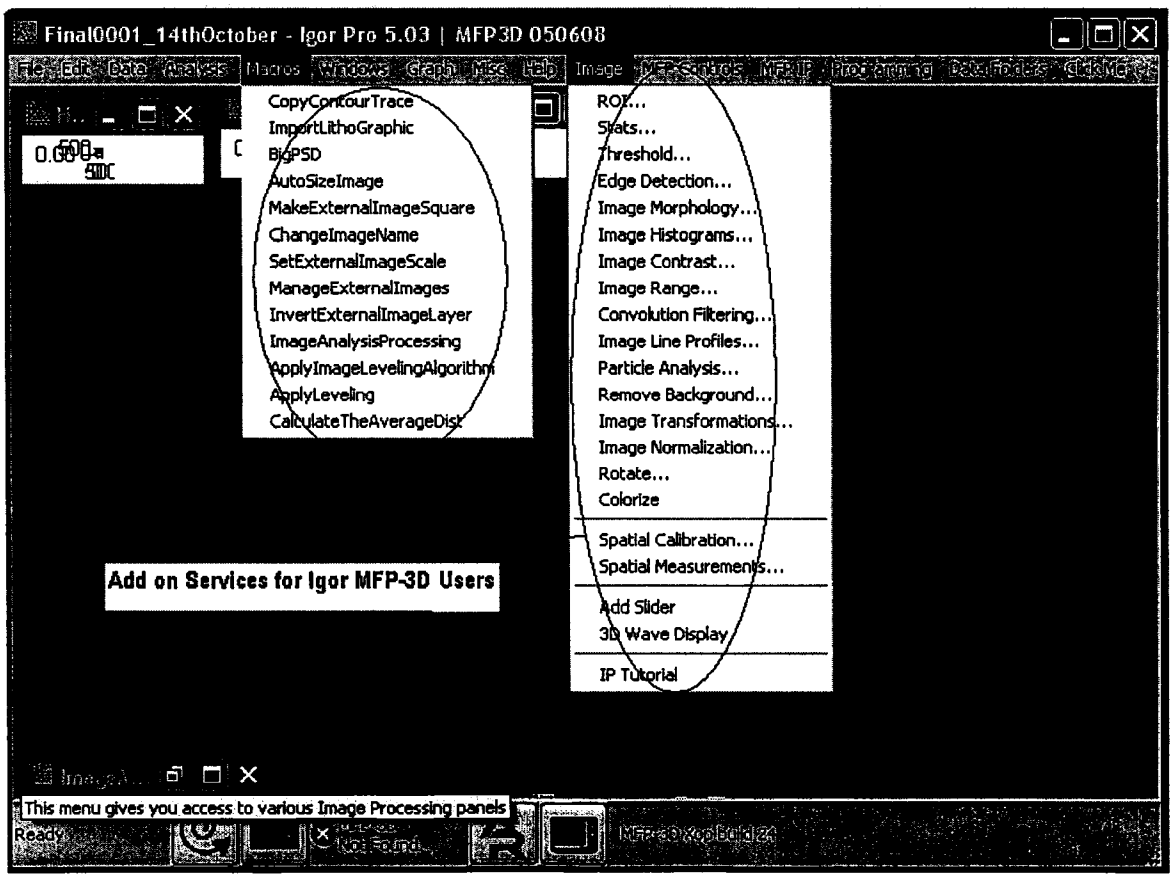


Figure A.1: Srvices exposed and created by user define procedure.

Our goal is to extract the statistics about the height, size and distribution of objects in the images. These statistics are then represented in the form of tables and graphs. Furthermore, we want to bring the scanned SEM (scanning electron microscopy) images into Igor for information extraction. Images that are scanned at a high resolution are easy to process and analyze.

A.2 Installation

To install the "Add-on" services Igor users are required to add procedure "ImageAnalysisProcedure" to their experiment. The user can observe extra options as shown in Images below after saving experiment. To install the "add-on" the user has to add "ImageAnalysisProcedure.ipf" to their experiment by going to the menu File$\longrightarrow$ Open File $\longrightarrow$ Procedure. Igor users can apply these services and utilize them during the processing of their images.

A.3 Usage

The first step in using Igor's image processing services is to load the desired images in the experiment. Then open the procedure file "ImageAnalysisProcedure" (by going to menu File $\longrightarrow$ Open File $\longrightarrow$ Procedure) in the current experiment and compile it by clicking the "compile" button at lower left. Before analyzing and processing it is highly recommended to save the experiment. Image processing and analysis is available to users in the form of various services. The usage of these services is explained in the following sections.

- Generate Image Statistics
- Apply Image Leveling Algorithm
- Calculate average distances between structures in an image
- Import an Image to Igor (SEM Image)

A.4 Generate Image Statistics

A.4.1 Flattening

The first step for generating image statistics is to flatten the image and extract the suitable layer of the image for processing. The image must be selected for processing to generate statistics. The first step is to flatten the image by the Asylum-provided flattening service. To perform that operation first click the button with caption "M" on top right of selected image. A new window will appear having tabs at the top. Select the tab with caption "Flatten" and then click the button "Flatten". Set the flattening order to zero.

After flattening the image click the "Auto" button at the top of the image. Image analyzing and processing routines can be applied now. To save memory during processing, the required layer will be extracted and processed.

A.4.2 Layer Extraction

To extract a layer from an Image, the user has two options. One can use the built-in command "extract layer" found in the dropdown list "Commands" on the image.

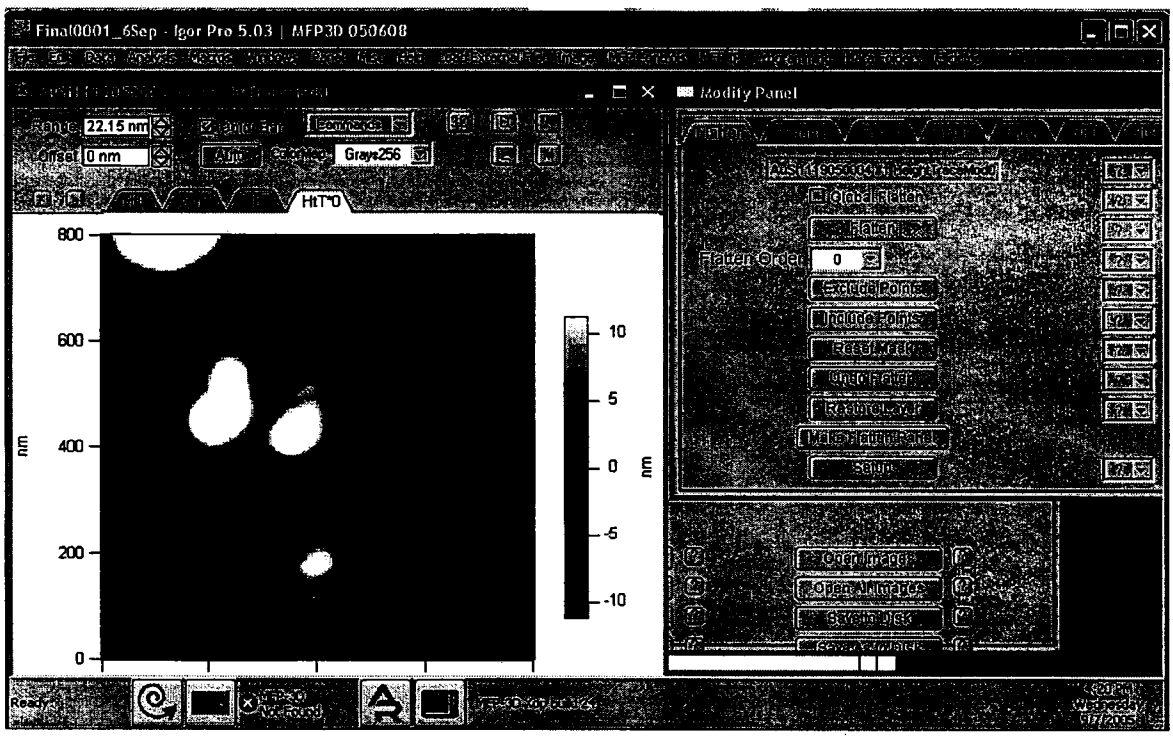


Figure A.2: Flattening the image before layer extraction

Otherwise, the user can select the image and then click the user service Extract layer" from the Macros menu. The extract layer process is explained with the image below:

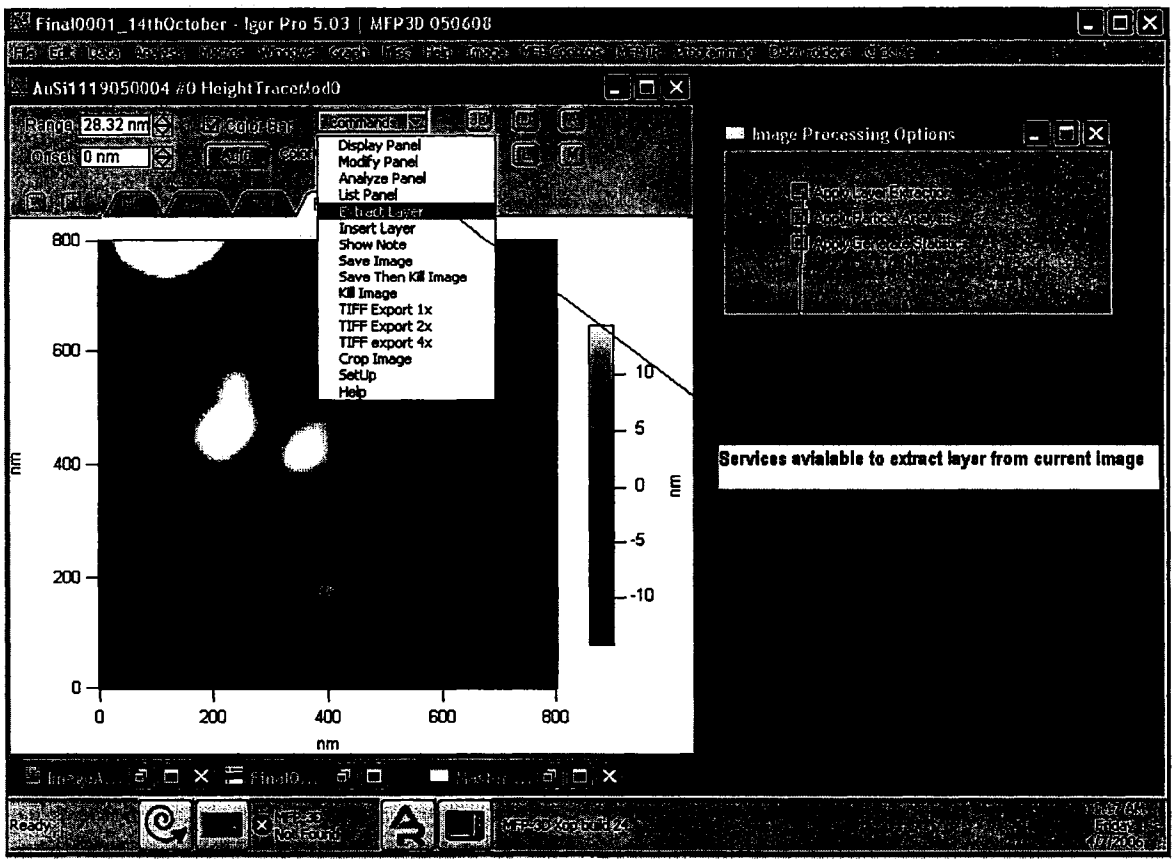


Figure A.3: Screen capture of Layer Extraction services for desired image

A.4.3 Noise Removal

In case of noise in the "LayerData" a noise filter will be applied by clicking "M" on top right of the image and then select the tab with caption "Filter". Median or rank filters will be used in case of impulse (salt & pepper) noise while Gaussian filters are used if Gaussian (uniform, normal, distributed) noise exists.

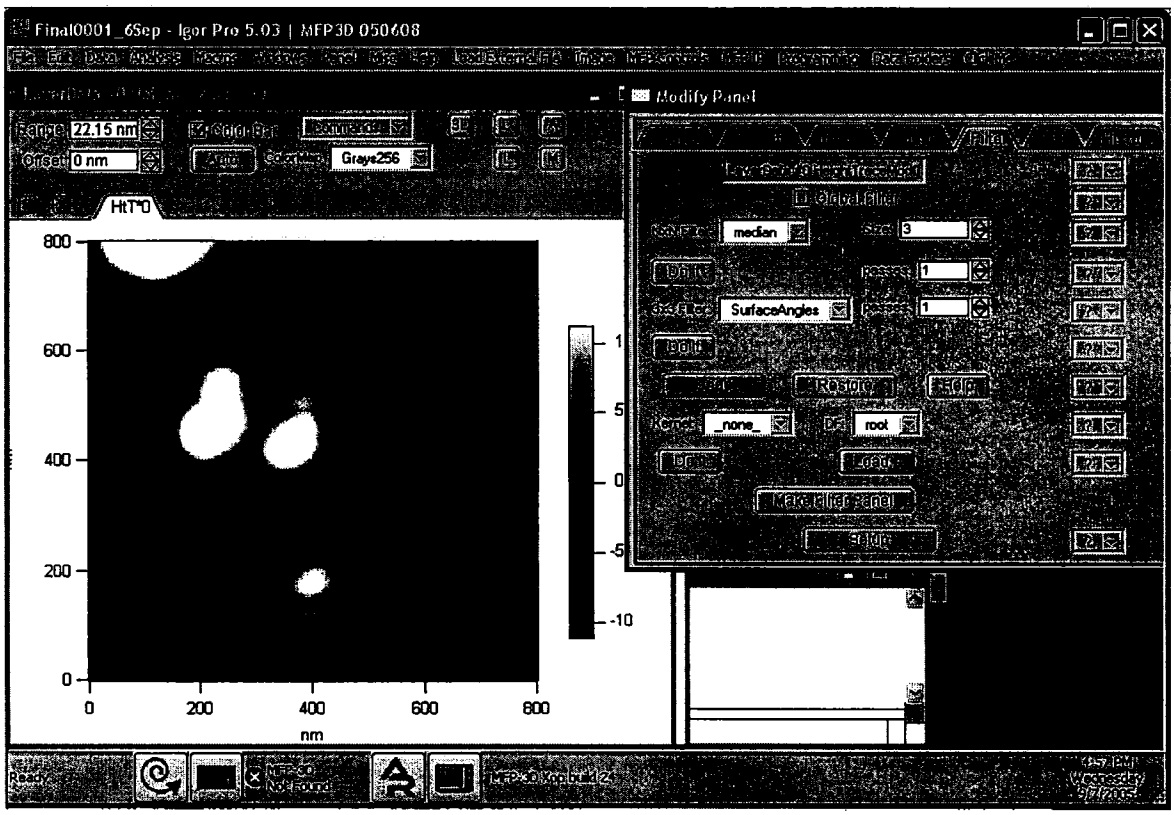


Figure A.4: Screen capture shows different filters that can be applied on Layer data according to the requirement of image

A.4.4 Image Statistics

The image is now ready for analysis and processing, so select the particle analysis option from the menu option "image" as shown in the image below. Then select the analysis parameters (normally the default values are fine) and click the button do it. The objects inside the image will be encircled as shown in the image below. If the boundaries of objects are not acceptable click the "Remove Overlay" button and change the parameters. If the user wants to make objects with numbers in the view, click the check box "label".

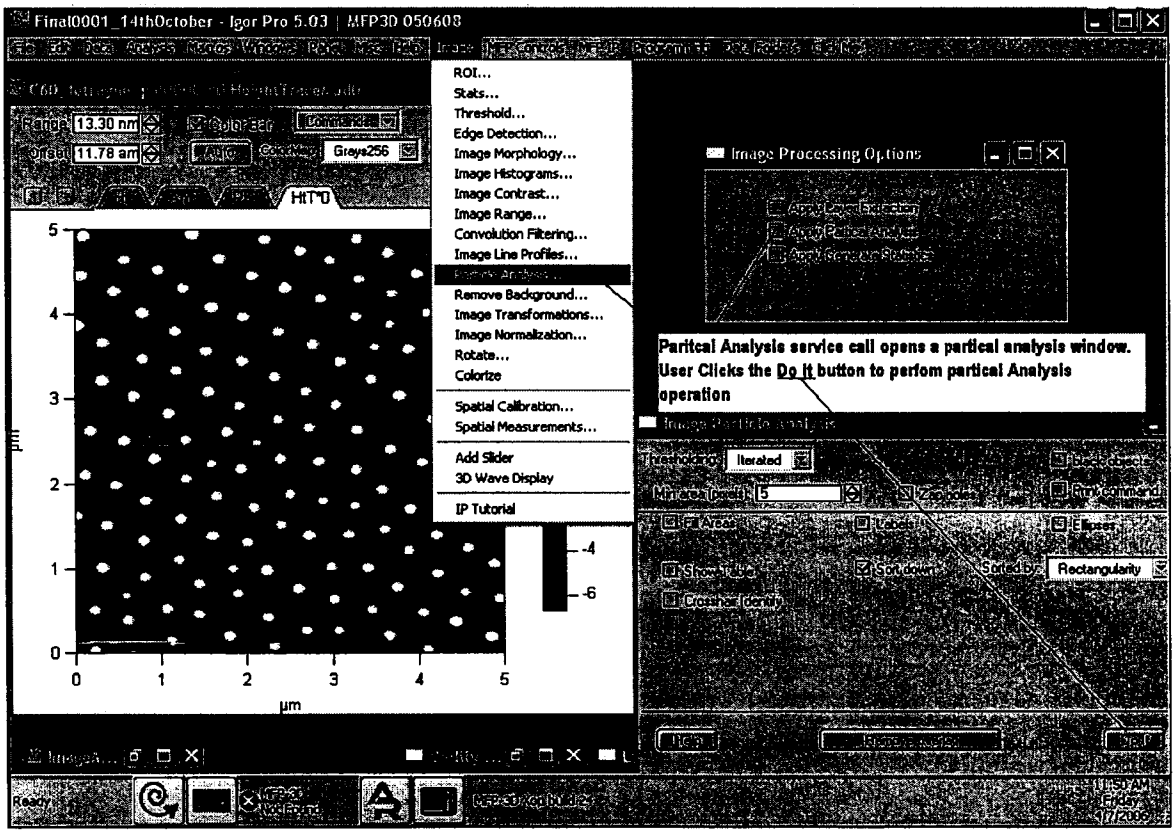


Figure A.5: Screen capture showing particle analysis process

The statistics about the image are generated and can be represented in the form of graphs and a summary file by clicking "ShowImageStatistics" from menu option "Macro" or from the user menu box by clicking Apply Generate Statistics. The user will be prompted to provide the directory for saving the text summary of the image objects.

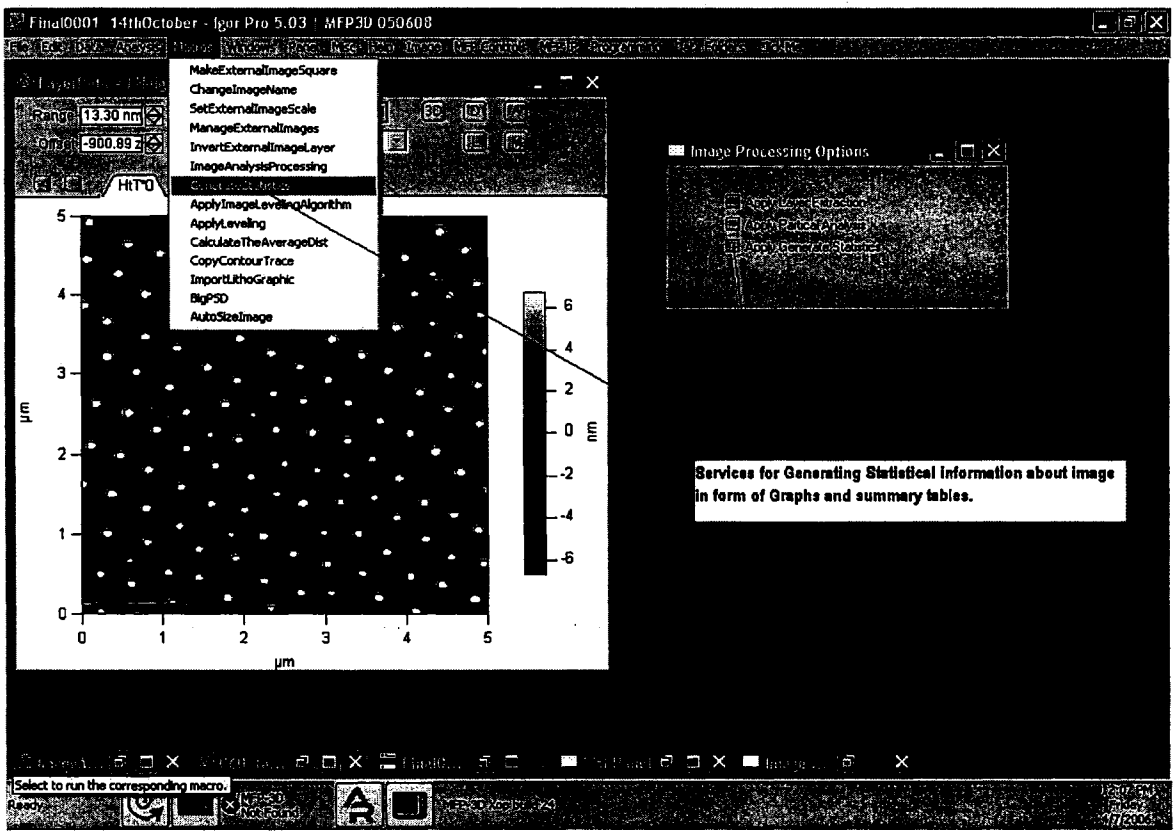


Figure A.6: Screen capture showing the services for generating Image statistics

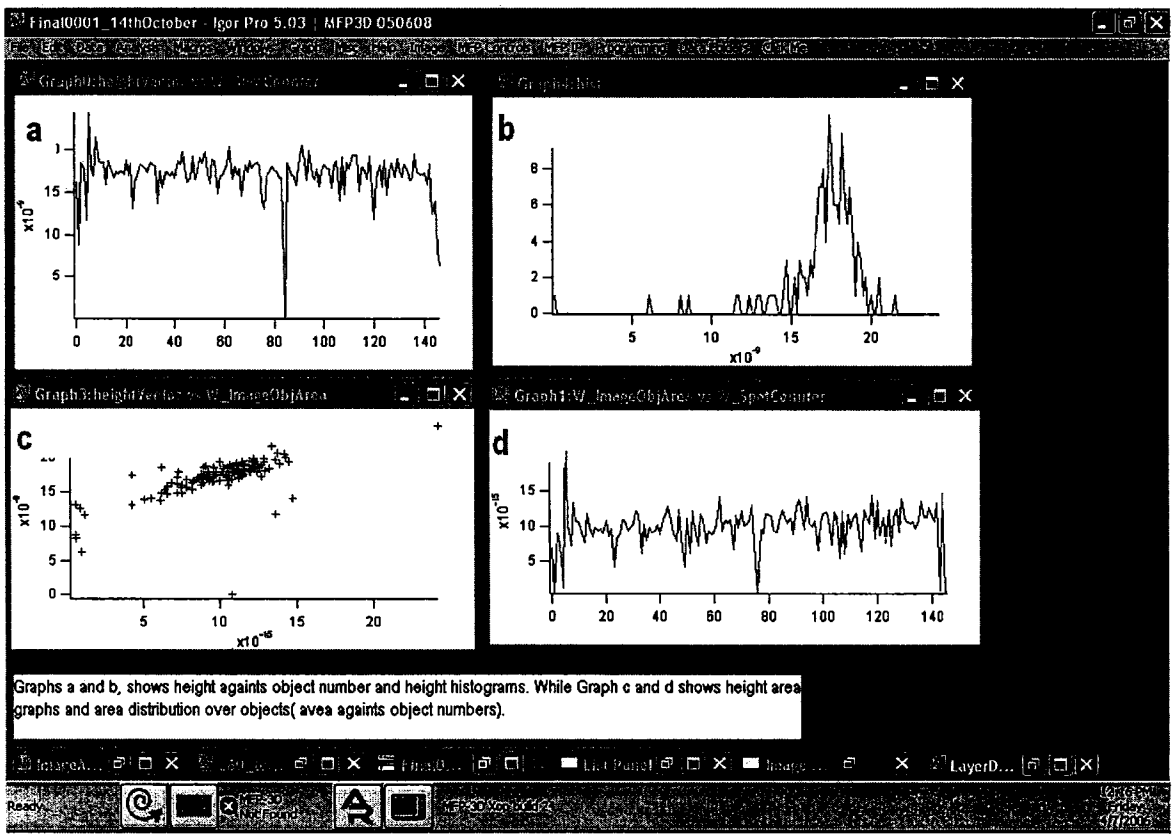
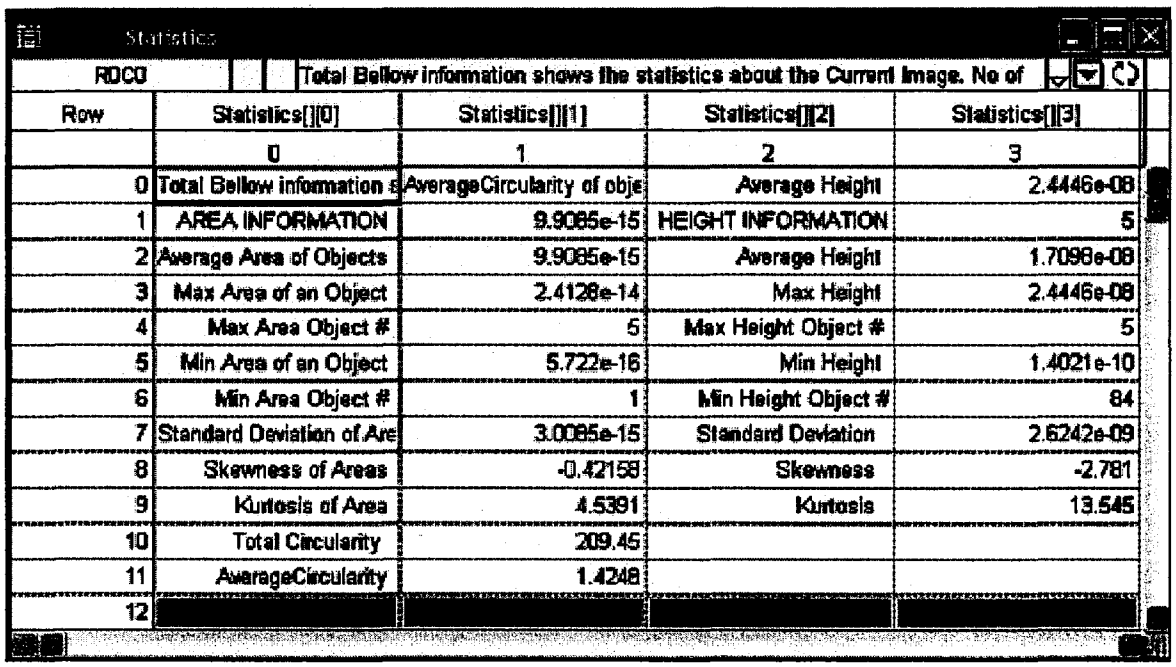


Figure A.7: A screen capture showing the graphical representation of information about the processed image



Statistics

RDCO | Total Bellow information shows the statistics about the Current Image. No of

Row	Statistics[][0]	Statistics[][1]	Statistics[][2]	Statistics[][3]
	0	1	2	3
0	Total Bellow information s	AverageCircularity of obje	Average Height	2.4446e-08
1	AREA INFORMATION	9.9065e-15	HEIGHT INFORMATION	5
2	Average Area of Objects	9.9065e-15	Average Height	1.7098e-08
3	Max Area of an Object	2.4128e-14	Max Height	2.4446e-08
4	Max Area Object #	5	Max Height Object #	5
5	Min Area of an Object	5.722e-16	Min Height	1.4021e-10
6	Min Area Object #	1	Min Height Object #	84
7	Standard Deviation of Are	3.0065e-15	Standard Deviation	2.6242e-09
8	Skewness of Areas	-0.42158	Skewness	-2.781
9	Kurtosis of Area	4.5391	Kurtosis	13.545
10	Total Circularity	209.45		
11	AverageCircularity	1.4248		
12				

Figure A.8: A screen capture showing statistical information in tabular format

A.5 Apply Image Leveling Algorithm

After applying the built-in "Flatten" algorithm, if the background of the image is still not level (i.e. there are some sharp bands or the background which have different intensity), the image leveling algorithm can be applied. To solve the problem of uneven background, our system provides two services for flattening: vertical flattening and horizontal flattening. The user of the system can apply these flattening algorithms by selecting a line from the background (ideal candidate line for flattening) and level the image background according to that line.

The "ApplyImageLevelingAlgorithm" option can be selected from the "Macro" menu to apply the appropriate leveling algorithms on the desired layer of data. The difference between the images before and after applying the leveling algorithm can be

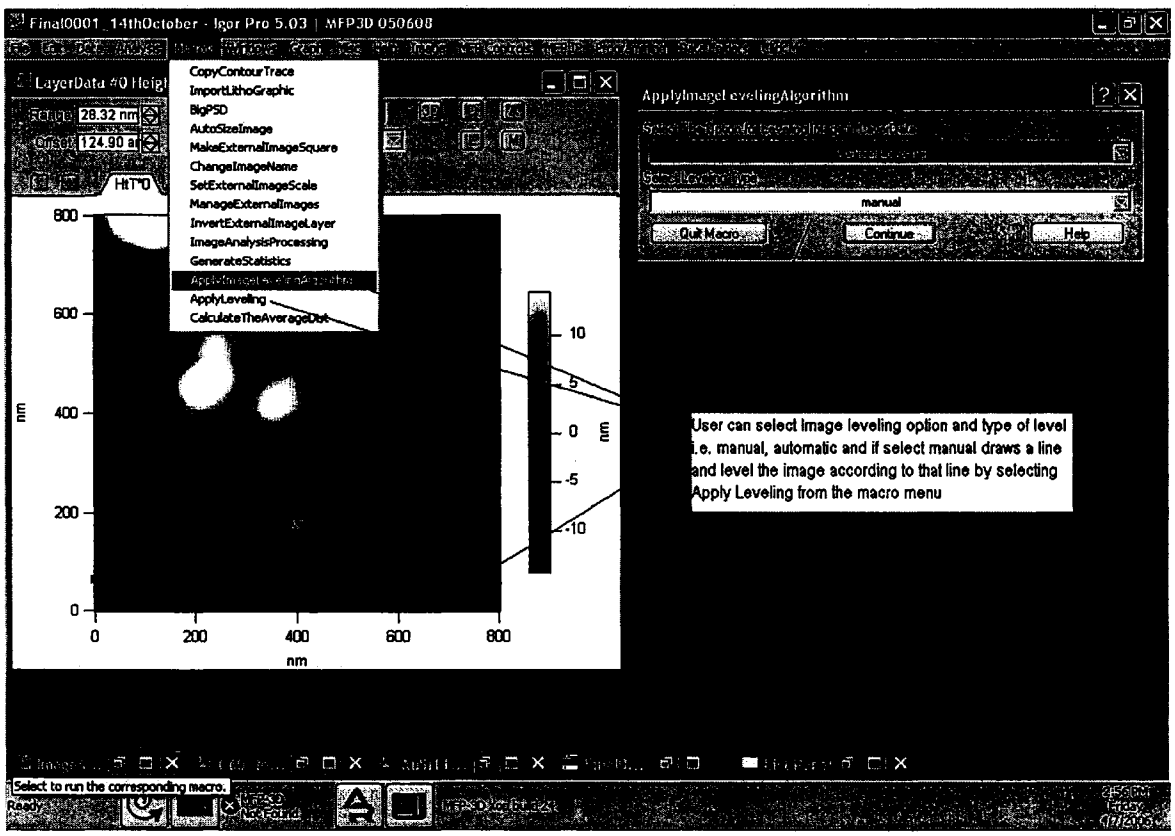


Figure A.9: Screen capture illustrates how to applying leveling on the image layer

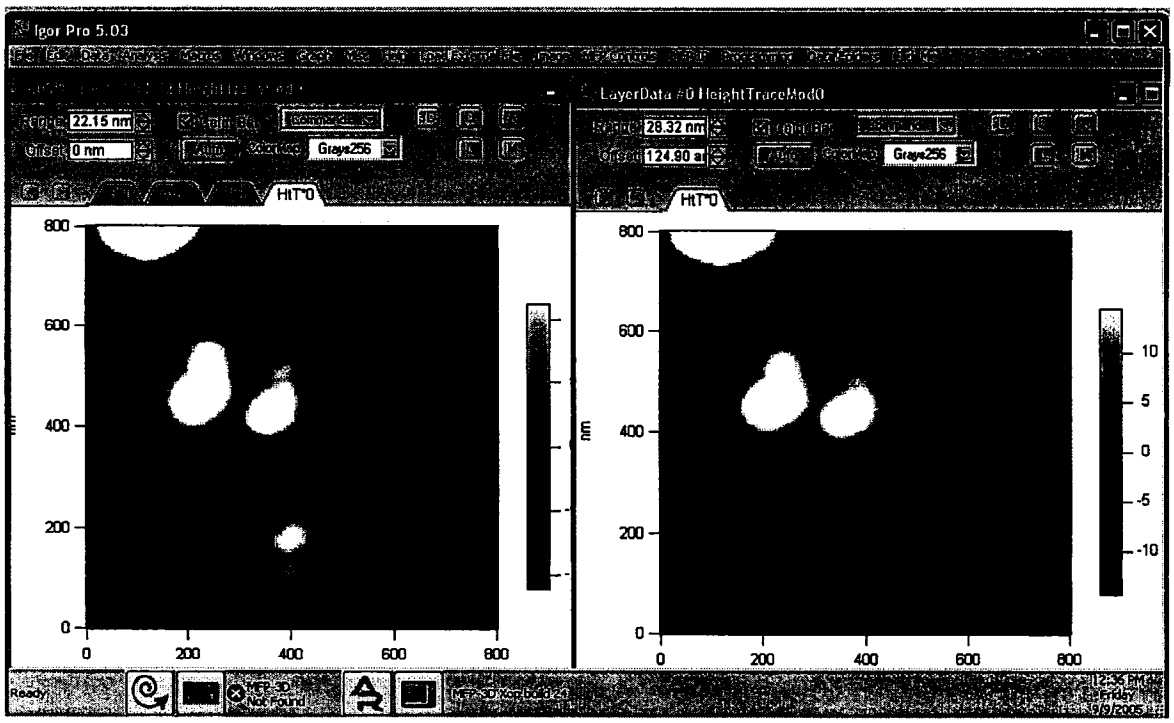


Figure A.10: Screen capture shows the result after applying horizontal leveling

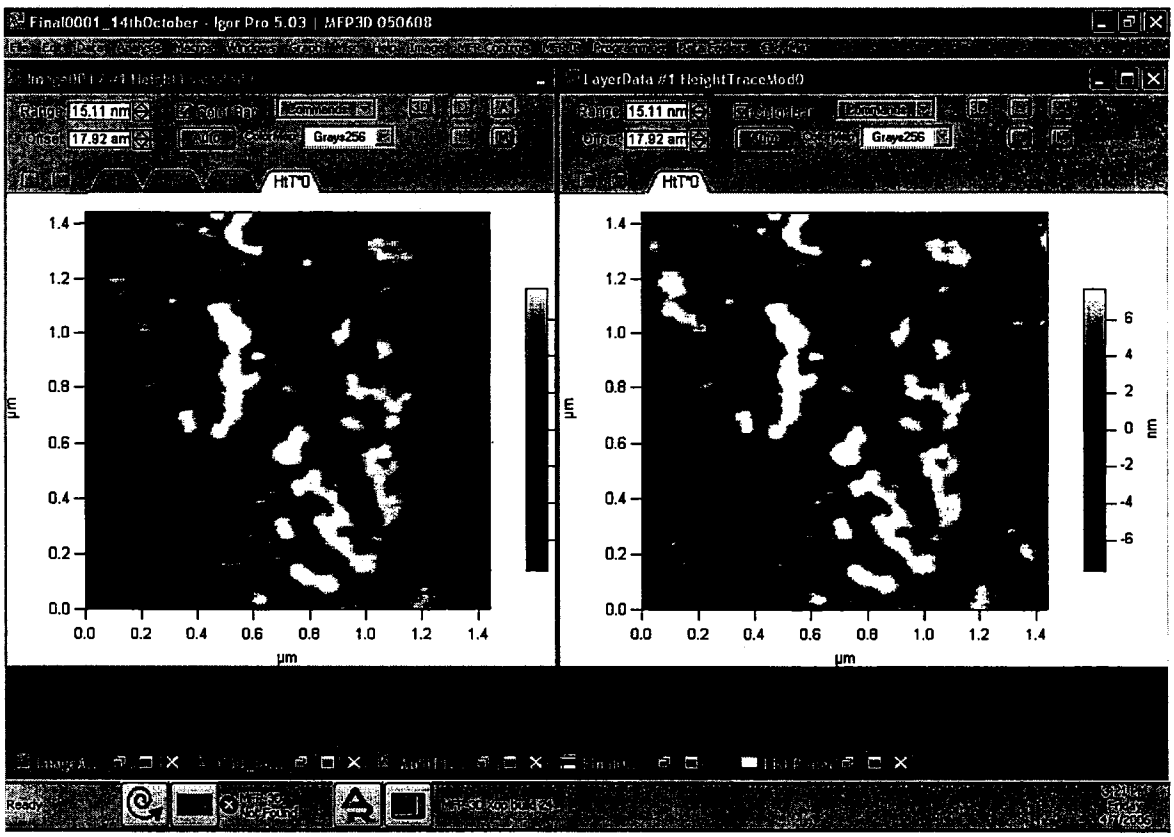


Figure A.11: Screen capture shows the result of applying vertical leveling

observed by looking at Figures A.10 and A.11 above. The extraction of true objects is also easier when the image background leveling algorithm is applied in the case of an image which has an imbalanced background.

A.6 Calculate average distances between structures in an image

This service calculates average distances between structures in an image by finding average distances between neighboring structures. The service ask the user to input an angle for a line and then the system finds structures along a series of lines normal to the user-input line. The system then calculates the distances between those structures.

The result of average distances for an image is calculated. The system normally calculates the average distance of structures for 5 angles closer to user input angle line, *i.e.* from [user angle – 2] to [user angle + 2]. This result will help the user to find a good candidate angle for finding the structure distribution in an image.

A.7 Import an Image to Igor (SEM Image)

To analyze an external analog image – such as a Scanning Electron Microscope (SEM) image on polaroid film – in Igor, the image must first be digitized with a scanner and saved in any standard graphics file format. To import the image, use the menu option "Data" then "Load Image" from "Load waves" as shown in Figure A.13. In the dialog box which appears, click the "File" button to select a file in the "Open File" dialog.

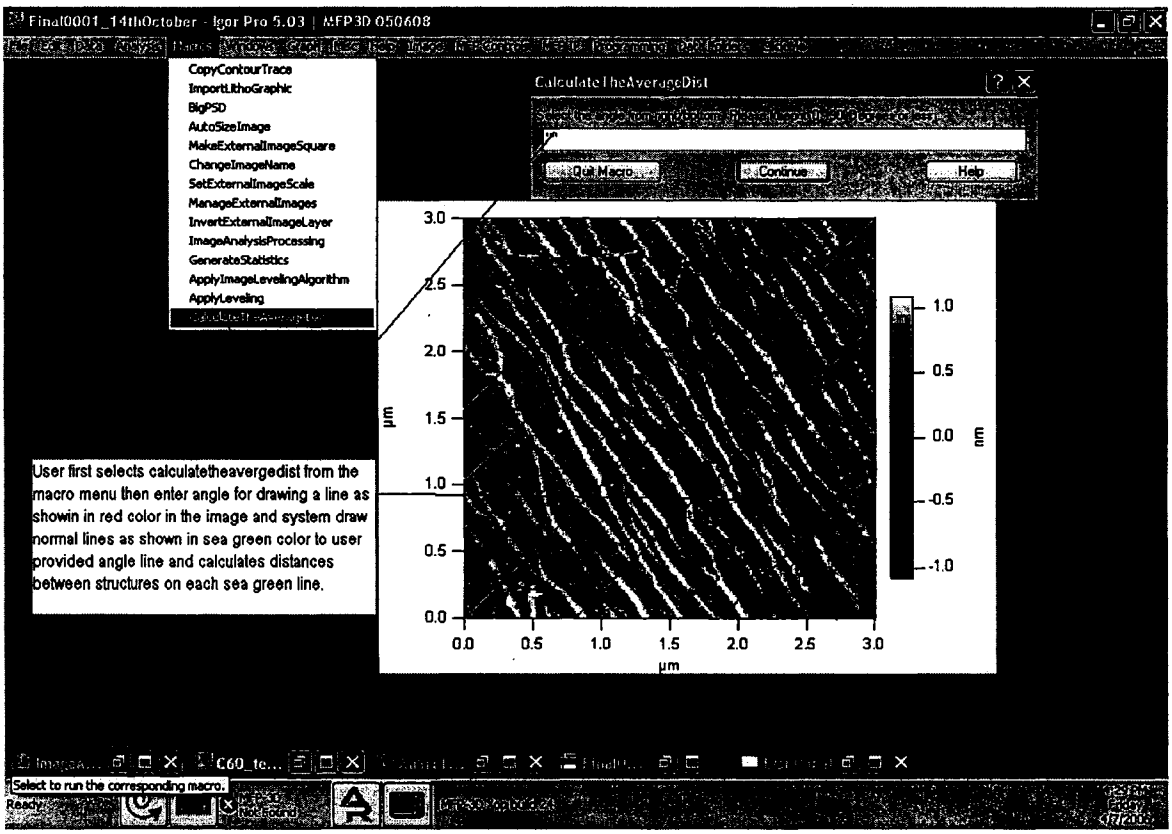


Figure A.12: Shows how to calculate the average distances of structures for some image

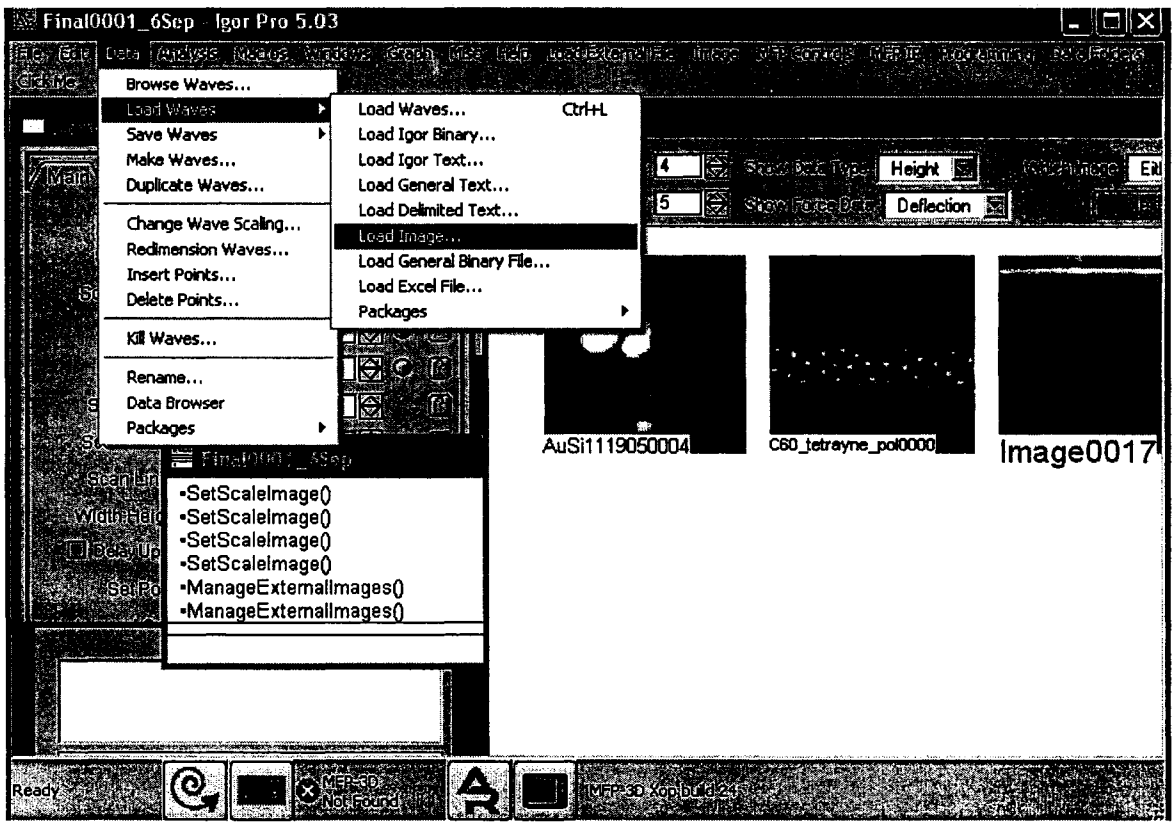


Figure A.13: Screen capture shows how to load an external (SEM) image

Do give some name to the file you want to import by clicking the radio button "Use the name". Do check the checkbox Display Image to see the import image.

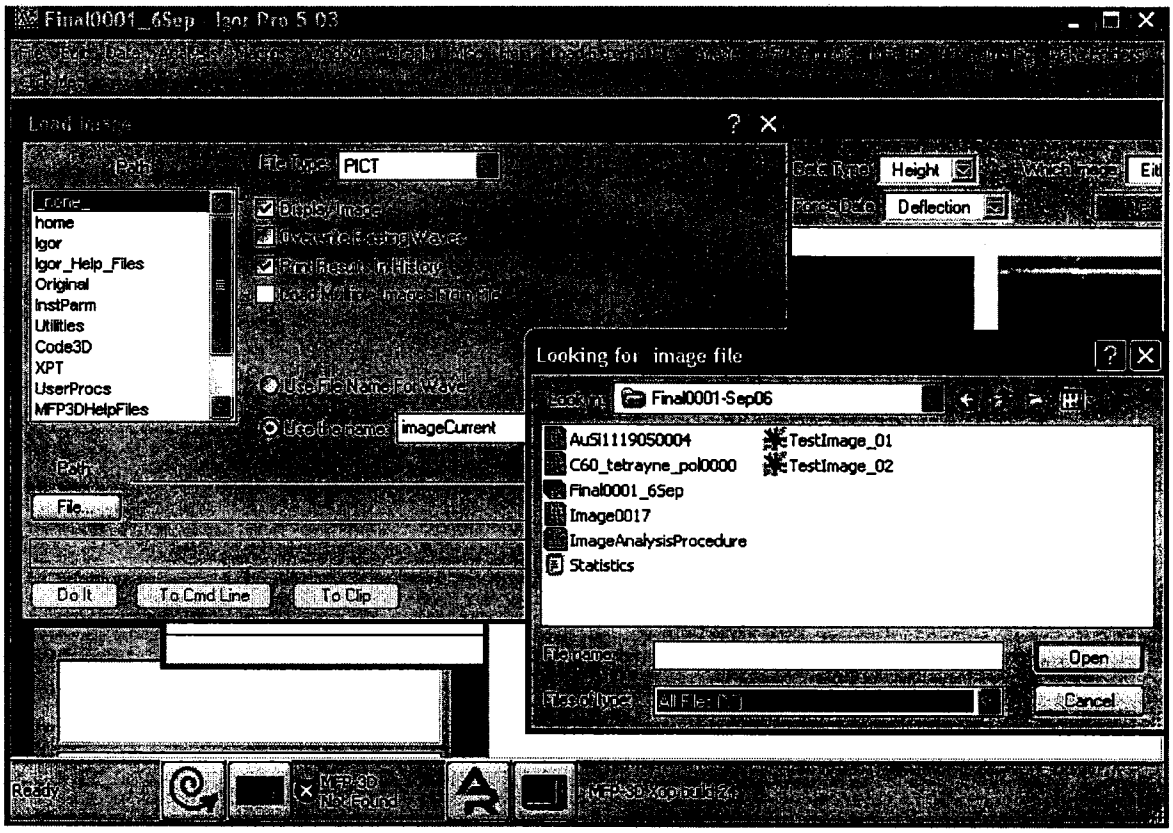


Figure A.14: Screen capture shows how to load an external image and save it so that it can be visible in MFP-3D software.

Note: If you are currently working with some images in the experiment, the image will be loaded into Image folder. In case a newly-loaded image is not loaded into the Image folder, it can be dragged and dropped to the Image folder from the root folder in the data browser. You can open the data browser by going to menu option "Data" and then "Data browser".

A.7.1 Set Image Scale (SEM Images)

This operation is available for images collected from tools other than the MFP-3D AFM. These images (*i.e.* SEM images) require scales setting to represent true size information. Menu option "SetImageScale" under "Macro" will perform this functionality. The image is selected from the drop down list for scaling. The size of the actual pixel will be provided for scaling of image. In our images the size of the pixel is constant so we set that value as default, but for a different set of images the pixel size will be different, so the field that takes the value of actual pixel size is kept editable. The continue button will perform the scaling and the result can be seen by viewing the image with the true dimensions of the image.

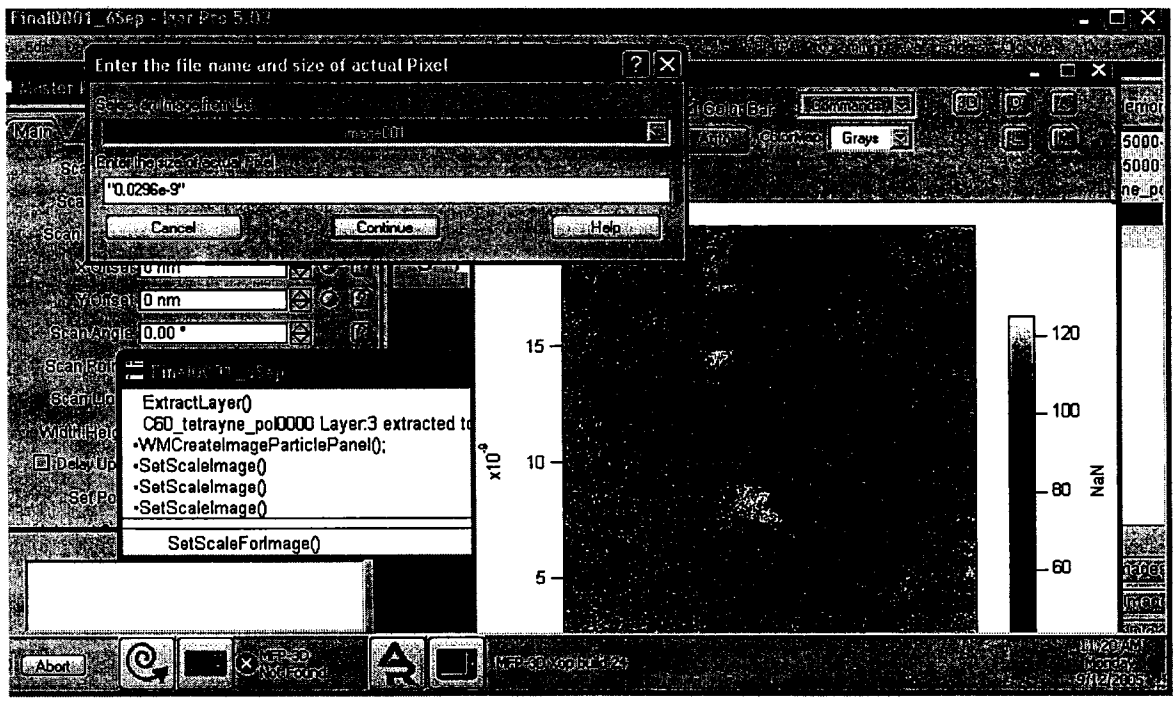


Figure A.15: A screen capture illustrating how to set the scale for a non-Igor (or SEM) image

A.7.2 Manage External Images (SEM images)

SEM images can be copied into a dummy image layer for processing using this operation. The user can either just copy the image directly to some Igor dummy layer or copy the inverted image (see below) into an Igor dummy layer. The image can be selected from the dropdown image list and the conversion mechanism should be selected and performed by clicking "continue". The new layer can be seen as "myLayerData" after refreshing images in the list panel.

A.7.3 Invert Layer Data (SEM images only)

Flipping/Inversion of an image is sometime required during image processing since the SEM negatives are often scanned in for their higher resolution relative to the polaroid prints. This can be done by clicking the "InvertLayerData" option from "Macro". Any SEM image in the layer data will be inverted. *Note: This option does not work for AFM images and should not be applied on AFM images.*

A.7.4 Make Image Square (SEM images only)

SEM images are normally not square and do not contain 2^n number of image points. Igor deals with images of $2^n \times 2^n$ number of data points more efficiently, so the Make Image Square option shrinks the image to suitable (2^n number of point. This option can be selected by selecting "MakeImageSquare" under menu option "Macros". Then the user has to select the image from the dropdown images list. The resultant square (2^n points) image is stored in LayerData.

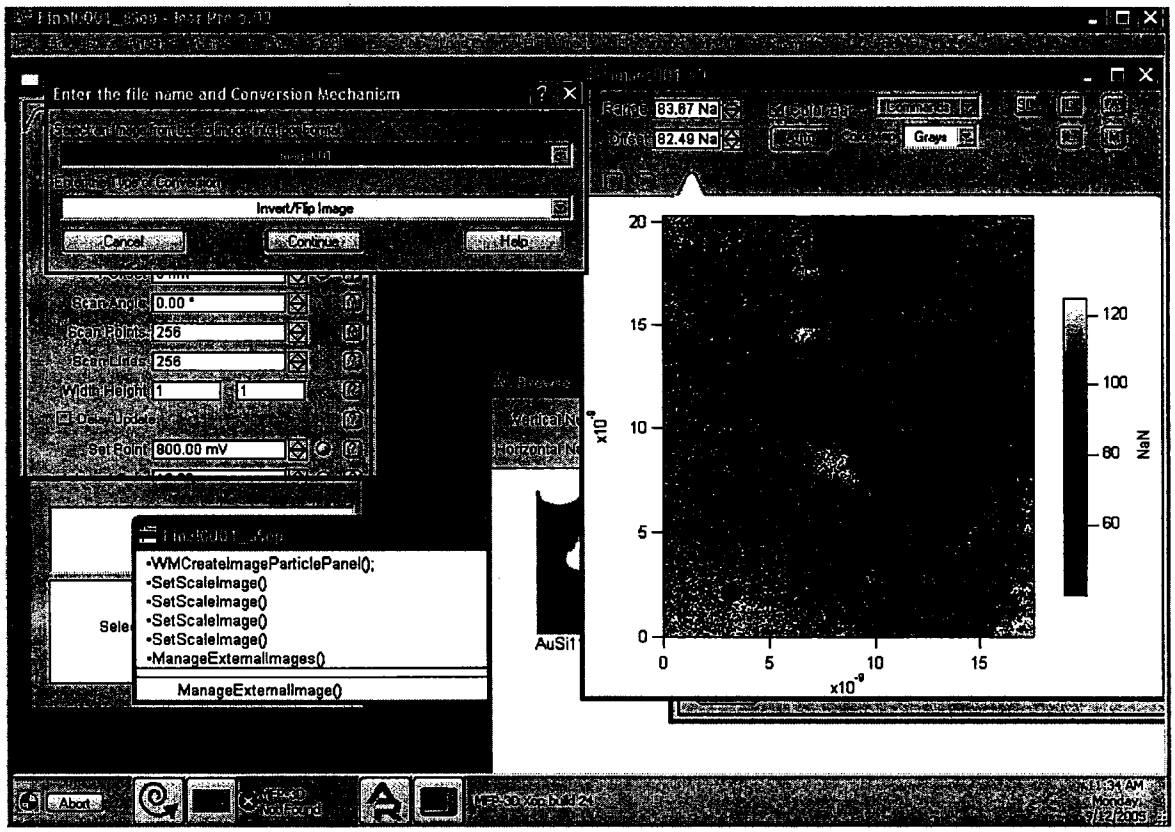


Figure A.16: A screen capture showing how to invert an image so that it will bring out objects from the background

A.8 Feedback:

Any feedback about these services or manual should be provided to Dr. Erika Merschrod (erika@mun.ca) or Jehangir by their email addresses or in person.

Appendix B

Code

All the code used and written in this research is written in IgorPro Scripting Language. The code calls the Image Processing procedures exposed by MFP-3D (wavematrix). The services given below are developed during this research.

```
#include <All IP Procedures>
#pragma rtGlobals=1 // Use modern global access method.
// During our process we tested these services on different SEM images
// successfully.  Every macro calls a function that does the
// the functionality closely related to name of function/service.
macro MakeExternalImageSquare()
    makeImageSquare()
end
//This Macro set is responsible for making non-igor images change
// in a way that we can process it just like igor native images.

macro ChangeImageName()
    renameImage()
end
macro SetExternalImageScale()
    SetScaleForImage()
end
macro ManageExternalImages()
    ManageExternalImage()
end
macro InvertExternalImageLayer()
    InvertLayer()
end
//End of ExternalImages Macros

// Macro that calls the user defined processes option box
macro ImageAnalysisProcessing()
    checkboxfunction()
end
//macro call generate statistics
macro GenerateStatistics()
    doGenerateStatistics()
end

// This function draws the box with Title Image Processing Options and
// then applies those processes according to user responses
Function checkboxFunction()
    NewPanel /k=1 /W=(150,50,450,180) as "Image Processing Options"
```

```
    Variable/G gRadioVal= 1
    CheckBox check0,pos={52,25},size={78,15},title="Apply Layer
     Extraction",value= 0,mode=0,proc =doLayerExtraction;
    CheckBox check3,pos={52,45},size={78,15},title="Apply Partical Analysis"
    ,value= 0,mode=0,proc = doAnalyzeParticles;
    CheckBox check4,pos={52,65},size={78,15},title="Apply Generate
     Statistics",value= 0,mode=0,proc = doGenerateStatistics;
End

Function doLayerExtraction(ctrlName,checked) : CheckBoxControl
    String ctrlName
    Variable checked
    if (checked == 1)
    ExtractLayer()
    endif
end

Function doAnalyzeParticles(ctrlName,checked) : CheckBoxControl
//This service checks if user selected to do analyze particles.
// If yes, it will open image analyze particle window
    String ctrlName
    Variable checked
    if (checked == 1)
         WMCreateImageParticlePanel();
    endif
end
Function doGenerateStatistics(ctrlName,checked) : CheckBoxControl
//This service checks if user selected to call generate image
 //statistics. It then calls the function Generate Image Statistics
    String ctrlName
    Variable checked
    if (checked == 1)
         GenerateImageStatistics();
    endif
end

// This macro asks the user about the type of leveling algorithm to
```

```
// apply and what mode s/he wants to apply
// Either to input image leveling information or use default
// normalization techniques
macro ApplyImageLevelingAlgorithm( Levelingoption,levelType)
    String Levelingoption,levelType
    Prompt Levelingoption,"Select The Option for Leveling Image in
    LayerData",popup,"Vertical Leveling;Horizontal Leveling;_none_"
    Prompt levelType,"Select Leveling Type",popup,"manual;automatic;"
    if (stringmatch(Levelingoption,"Vertical Leveling") ==1)
        if(stringmatch(levelType,"automatic")==1)
            ImageVerticalLevel(0)
        else
            GraphWaveDraw/O/L/R
// Here system asks the user to draw the input line
        endif

        else stringmatch(Levelingoption,"Horizontal Leveling") ==1)
        if(stringmatch(levelType,"automatic")==1)
            ImageHorizontalLevel(0)
        else
            GraphWaveDraw/O/B/T
// Here system asks the user to draw the input line

        endif
    endif
End

// When user selects to draw a line for leveling the image, the system
// then asks the user again to apply the line in leveling.  Although
//it's an added burden for the user, we have to ask the user twice.  We
// cannot declare global variables, so we cannot store the user
// response in macro as we cannot declare static data type in macros.

// declare global variables so cannot store the user response in macro
 // as we cannot declare static data type in macros.
Macro ApplyLeveling(Levelingoption)
    String Levelingoption
    Prompt Levelingoption,"Select The Option for Leveling Image in
```

```
    LayerData",popup,"Vertical Leveling;Horizontal Leveling;"
    if (stringmatch(Levelingoption,"Vertical Leveling") ==1)
        doUserLeveling("Vertical")
    else
        doUserLeveling("Horizontal")
    endif
end
// Call to to the user selected funtion for User input selected line.
Function ApplyUserLeveling(Ltype)
    string Ltype
//store the leveling type either vertical or horizontal
    wave userLine = root:Images:W_YPoly0
    variable dimension  = DimSize(root:Images:LayerData,0 )
    variable userRC =  floor(((1+ userLine[0])/2)* dimension)
     //Calculate the user defined row or column in image for the drawn line
    if (stringmatch(Ltype,"Vertical") ==1)

        ImageVerticalLevel( userRC)
    else
        ImageHorizontalLevel(userRC)

    endif
end
// This function is responsible for leveling the image vertically when
//user asks for vertical leveling of the current image layer.
Function ImageVerticalLevel(row)
 //Layer Select for normalization default value is zero in case of
 //automatic
    variable row
    wave imageWave = root:Images:LayerData
    String notes =  note (root:Images:LayerData)
// reading current image notes into a string for further processing
    variable l_row , tot_rc , s_column , averagevalue, sumLine,column

    tot_rc = DimSize(root:Images:LayerData,0 )
// this variable saves the value of total number of rows or columns as
// images are square so row = column
```

```
    Make/O/N=(tot_rc) userSelRow
// Make a vector to save the user selected Image line for
// normalization.
    userSelRow = 0

// This loop adds the pixel values in user selected line for
//calculating average values.
    for (column =0 ;column<tot_rc ;column+=1)
        sumLine += imageWave[row][column]
    endfor
    averagevalue= sumLine/row
// Average of user selected image line values

// This loop calculates the difference of average of selected line to
 //actual values in selected line and normalization vector is created.
    for (column =0 ;column<tot_rc ;column+=1)
        userSelRow[column ] = averagevalue - imageWave[row][column]
    endfor

// This loop applies normalization vector on the whole image under
//consideration that is layer data.
    for (l_row =0 ;l_row<tot_rc ;l_row+=1)
        for (s_column =0 ;s_column<tot_rc ;s_column+=1)
            imageWave[l_row][ s_column ] = imageWave[l_row]
            [s_column] +userSelRow[ s_column ]
        endfor
    endfor

// Return memory of unrequired user variables , strings and vectors to
 // the Processor.
    Killvariables /A/Z
    KillStrings /A/Z
    Killwaves userSelRow
end
//End of Vertical Leveling

// This function levels the image when user selects horizontal
//leveling/flattening according to user request
```

```
Function ImageHorizontalLevel(column)
variable column
// User Selected line or zero for default if user select automatic
wave imageWave = root:Images:LayerData
// Load the current Image Layer into User wave for processing
String notes =  note (root:Images:LayerData)
// reads notes from the current Image for getting image detail
variable tot_rc, row, s_column , averagevalue, sumLine
//Local variables declearign storing temp data and loop controls
tot_rc = DimSize(root:Images:LayerData,0 )        // This variable
//saves rows or columns numbers as square images in our case so we have
// one variable for both rows and columns

Make/O/N=(tot_rc) userSelRow
// Make a vector to save the user selected Image line for normalization.
    userSelRow = 0

    variable dummycol
// Dummy variable to store the column value for temporary usage
    dummycol = column

    for (row =0 ;row<tot_rc ;row+=1)
        if(imageWave[row][column] > 0)
// This checks if there is an object in user selected line so don't
 //make it part of user
// selected vector (line ) for image leveling
            do
                column +=1
            while  (imageWave[row][column] < 0)
// This skips all the values of particle in selected line

            sumLine += imageWave[row][column]
// Add the previous value for every particle point instead of particle
 //pixel value
            column = dummycol
// Helps remember control of old point on line before approach of a
//particle in
// User selected line or default zero indexed line
```

```
        else
            sumLine += imageWave[row][column]
// For regular scenario just add the value for normalization vector
        endif
endfor

averagevalue= sumLine/tot_rc
// Average of User Selected line

// This loop creates the " average - Selected line vector " for
//normalizations of the current image
for (row =0 ;row<tot_rc ;row+=1)
        if(imageWave[row][column] > 0)
            userSelRow[row] = userSelRow[row-1]
        else
            userSelRow[row] = averagevalue - imageWave[row][column]
        endif
endfor

//These for loops apply the user selected line normalization row by row
 //to the whole image.
for (column =0 ;column<tot_rc ;column+=1)
    for (row =0 ;row<tot_rc ;row+=1)
        imageWave[row][ column ] = imageWave[row][column] +userSelRow[ row ]
    endfor
endfor

// Killing undesirable variables, strings and waves to release memory
// for other processes.
Killvariables /A/Z
KillStrings /A/Z
Killwaves userSelRow
end
// End of horizontal leveling

// The Function GenerateImageStatistics is responsible for generating
//image statistics for the image in Layerdata
```

```
// and for which particle analysis procedure is performed and particles
 //are identified. This function calculates and shows
// results in tabular format for image objects' height, areas, standard
//deviation and average heights and areas. etc.
Function GenerateImageStatistics()

//Load the Object areas vector calculated by analyze particle into
//areaVector wave for further processing
    wave areaVector = root:Images:LayerData_WMUF:Particles:W_ImageObjArea
//Load the Object circularity vector calculated by  analyze particle
 //into areaVector wave for further processing
    wave circularity =
     root:Images:LayerData_WMUF:Particles:Circularity
    WaveStats  areaVector
// WaveStats returns the statistics in the automatically created
//variables
    Make /O/T/N=(12,4) Statistics
//Table for storing resulting summary of the image

//Declaration of variables for later use, naming of variable shows its
//functionality
    variable counter, maximum, minimum,average,
    stdeviation,skewness,noofobjects, maxhono,minhon
    variable kurtosis,i_length,i_area,p_area,o_totpix,obj_circularity,
    circularityloop

//putting statistics information from automatically set variables
 //with "WaveStates" into user define variables
    noofobjects =V_npnts
    average = V_avg
    maximum =V_max
    maxhono = V_maxloc
    minimum = V_min
    minhon = V_minloc
    stdeviation =V_sdev
    skewness = V_skew
    kurtosis = V_kurt

    Make/N =(noofobjects)/O heightVector
```

```
// vector to store the height of every object in the image and used for
//averaging
    String notes =  note (root:Images:LayerData)
//Save notes "information associated with image" of the image into
//string variable

//The below Unit of code calculates the pixel size which is further
//used to calculate pixel area in true scale
    i_length = str2num( Stringbykey ("ScanSize",notes,":","\r"))
    i_area = i_length*i_length
    o_totpix = str2num( Stringbykey ("ScanPoints",notes,":","\r"))
    p_area = i_area/(o_totpix*o_totpix)

    variable temp_counter
    temp_counter = 0
    obj_circularity =0

// calculate total circularity using data in Circularity vector and
//store it in variable obj_circularity "objects circularity"
    for(circularityloop=0; circularityloop<noofobjects;circularityloop+=1)
        obj_circularity +=circularity[circularityloop]
    endfor

// Calculate true area for every object by multiplying the pixel area
//to each area item
    if ( areaVector[temp_counter] > i_length)
        do
            areaVector[temp_counter] = areaVector[temp_counter]*p_area
            temp_counter+=1
        while (noofobjects>temp_counter)
    endif

    if( WaveExists(Statistics) )
// If the statistics file exists store the statistical information at
//mentioned indices

        Statistics [0][0] = "Total Below information shows the
        statistics about the Current Image. No of Objects
         "+num2str(noofobjects)
```

```
        Statistics [1][0] = "AREA INFORMATION "
        Statistics [2][0] = "Average Area of Objects  "
        Statistics [2][1] =num2str(average)
        Statistics [3][0] = "Max Area of an Object "
        Statistics [3][1] =num2str(maximum)
        Statistics [4][0] = "Max Area Object # "
        Statistics [4][1] = num2str(maxhono)
        Statistics [5][0] = "Min Area of an Object "
        Statistics [5][1] =num2str(minimum)
        Statistics [6][0] = "Min Area Object # "
        Statistics [6][1] = num2str(minhon)
        Statistics [7][0] = "Standard Deviation of Areas "
        Statistics [7][1] = num2str(stdeviation)
        Statistics [8][0] = "Skewness of Areas "
        Statistics [8][1] = num2str(skewness)
        Statistics [9][0] = "Kurtosis of Areas "
        Statistics [9][1] = num2str(kurtosis)
        Statistics [10][0] = "Total Circularity  "
        Statistics [10][1] = num2str(obj_circularity)
        Statistics [11][0] = "Average Circularity "
        Statistics [11][1] =num2str(obj_circularity/ noofobjects)
// Storage of statistical area related information ends here

        wave LayerData = root:Images:LayerData
// load wave into user-defined wave for processing
        variable objectno
        objectno =0

// Loading particle locations in the image into user define waves to
//identify the starting point of every object
// and then walk through each object to identify height of each object
        wave min__xData =  root:Images:LayerData_WMUF:Particles:W_xmin
        wave max__xData =
         root:Images:LayerData_WMUF:Particles:W_xmax
        wave min__yData = root:Images:LayerData_WMUF:Particles:W_ymin
        wave max__yData = root:Images:LayerData_WMUF:Particles:W_ymax
```

```
// This loop finds the height of an item with each of its iterations.
        do
            variable xmin,xmax,ymin,ymax,maxval,loopcounter,
row,column
// Declaring variables for location of particles
// Setting variables for starting point of each particle
            xmin =min__xData[objectno]
            ymin =min__yData[objectno]
            xmax =max__xData[objectno]
            ymax =max__yData[objectno]
            maxval = LayerData[xmin][ymin]
            row =xmin
            column = ymin

            loopcounter = max(row,column)
//row and column have same value so anything will be fine

            variable forcounter
//This loop searches for height of the current particle. It looks for
//previous and next item for every pixel item.
// If previous point and next are both lower in height as compared to
//current pixel point, then it's a candidate for
// height particle and high point of the current object.
            for (forcounter=0; forcounter <loopcounter;forcounter+=1)

                if(LayerData[row+1][column] >maxval   &&
                (LayerData[row+1][column] >LayerData[row][column+1] &&
                (LayerData[row+1][column]>LayerData[row+1][column+1])))
                maxval = LayerData[row+1][column]
                    row+=1

                elseif (LayerData[row][column+1] >maxval   &&
                (LayerData[row][column+1] >LayerData[row+1][column] &&
                (LayerData[row][column+1]>LayerData[row+1][column+1])))
                    maxval = LayerData[row][column+1]
                    column+=1

                elseif (LayerData[row+1][column+1] >maxval   &&
                (LayerData[row+1][column+1] >LayerData[row+1][column]
```

```
                && (LayerData[row+1][column+1]>LayerData[row][column+1]))
                    maxval = LayerData[row+1][column+1]
                    column+=1
                    row+=1
                else
                    break;
                endif
            endfor
// end of far loop and one height is identified
            heightVector[objectno] = maxval
// Height value is stored against object number for further processing
            objectno+=1
        while(objectno<noofobjects)
// This loop runs for every object in image to calculate height
        WaveStats heightVector
// Automatically setting information variables
// Storing Objects height information into summary table
        Statistics [1][2] = "HEIGHT INFORMATION"
        Statistics [2][2] ="Average Height "
        Statistics [2][3] =num2str(V_avg)
        Statistics [3][2] = "Max Height "
        Statistics [3][3] = num2str(V_max)
        Statistics [4][2] = "Max Height Object # "
        Statistics [4][3] =num2str(V_maxloc)
        Statistics [5][2] ="Min Height "
        Statistics [5][3] =num2str(V_min)
        Statistics [6][2] = "Min Height Object #"
        Statistics [6][3] = num2str(V_minloc)
        Statistics [7][2] ="Standard Deviation  "
        Statistics [7][3] =num2str(V_sdev)
        Statistics [8][2] = "Skewness  "
        Statistics [8][3] =num2str(V_skew)
        Statistics [9][2] ="Kurtosis  "
        Statistics [9][3] =num2str(V_kurt)
        Statistics [10][2] =""
// To make the balance entries we have to put empty strings
```

```
        Statistics [10][3] =""
//Where data is not required limitation of the system otherwise
        Statistics [11][2] =""
// the system will print junk values.
        Statistics [11][3] =""

    endif

// Display different comparison graphs with various modes
//[mode of view]
    display heightVector vs root:Images:LayerData_WMUF:Particles:W_SpotCounter
    display areaVector vs root:Images:LayerData_WMUF:Particles:W_SpotCounter
    display heightVector vs areaVector; modifygraph mode =2
    display heightVector vs areaVector; modifygraph mode =3
    make /o hist
    histogram /B ={V_min , (V_max-V_min)/V_npnts, V_npnts}
    heightVector, hist
    display hist

// Kill the variables that are no longer required to return memory to
//operating system
    killVariables /A/Z
    KillStrings /A/Z
end

macro CalculateTheAverageDist(degree )
     string degree
     Prompt Degree,"Select the angle from right/bottom.  Please keep
    it '0 - 90' Degrees or less "

    if (V_Flag)

        return -1  // User cancelled
    endif
    distanceofobjects(str2num( degree))
end

Function distanceofobjects( degree)
    variable degree
```

```
     Make /O/T/N=(5,2)  avgDistances
//Make it five rows and two columns
    if ( degree > 4 && degree< 86)
        avgDistances[0][0] = "The average distance for angle"+num2str(degree-2)
        avgDistances[0][1] =num2str(findAverageNeighbourDistance( degree-2))
        avgDistances[1][0] = "The average distance for angle"+num2str(degree-1)
        avgDistances[1][1] =num2str(findAverageNeighbourDistance( degree-1))
        avgDistances[2][0] = "The average distance for angle"+num2str( degree)
        avgDistances[2][1] =num2str(findAverageNeighbourDistance( degree))
        avgDistances[3][0] = "The average distance for angle"+num2str( degree+1)
        avgDistances[3][1] =num2str(findAverageNeighbourDistance( degree+1))
        avgDistances[4][0] = "The average distance for angle"+num2str( degree+2)
        avgDistances[4][1] =num2str(findAverageNeighbourDistance( degree+2))
    else
        avgDistances[2][0] = "The average distance for angle "+num2str( degree)
        avgDistances[2][1] =num2str(findAverageNeighbourDistance( degree))
    endif
end

// This function takes an angle as an input and uses the angle for
//drawing a line from the right bottom of the image.
// The system draws multiple normal (perpendicular) lines to the angled
//line and calculates distances of objects lies on tangent lines.
// The System then calculates the average distance of every object to
 //its neighbouring object.
Function findAverageNeighbourDistance(degree)
    Variable degree
    Make /O  distances

//area of declaring all the local variables used in the Averaging
//procedure
// variable dimension is representing number of rows and columns
    Variable dimension , sizeofpixel , selectedcol, selectedrow, startrow,
    startcol, endrow, endcol, tempstartrow, tempstartcol, tempendrow, tempendcol
```

```
    variable  checkend, noofpoints, firstheight, distance,
    noofpositivevalues, distancenumber
    variable distancecounter, totalofdistances, selectrow
    totalofdistances = 0
//is Used to calculate average distances total/no of distances

    wave layerdata = root: Images: layerdata
    sizeofpixel = deltax(root:Images:LayerData )
//Extract true size of pixel from the current image
    dimension = DimSize(root:Images:LayerData,0 )
// Dimension of current image rows and/or columns

// Following check sets start and end of angled line on the image
// Details about function : If angle is less than 45 we calculate the
 //starting point of line by drawing angle with base in case
// angle is more than 45 we draw the angled line by considering angle
//with the right side of the image

    IF (degree>45 && degree < 90)
        degree = (90 - degree)
        selectedrow = degree / 45 * dimension
        selectedrow = round(selectedrow)
        selectedrow = dimension - selectedrow
        endrow = selectedrow - 1
        endcol = dimension
        startrow = 0
        startcol = dimension - (selectedrow)
    Else
        startrow = 0
        selectedcol = degree / 45 * dimension
        selectedcol = round(selectedcol)
        endrow = dimension - (selectedcol)
        startcol = selectedcol - 1
        endcol = dimension
    EndIF
```

```
    distancecounter = 0
    checkend = 0
//Helps loops to adjust the start and end position of lines parallel to
 //the input angled line.

    Do

//Using temporary variables for loop functionality to avoid change in
 //original drawn line.
        tempstartrow = startrow
        tempstartcol = startcol
        tempendrow = endrow
        tempendcol = endcol

//This check controls the start of line for given angle by moving it
 //with reference to change end of line with every iteration of the
 //loop.
        IF (checkend == 1)
            tempendrow -=1
            tempendcol-=1
        EndIF

//Set all the variables that determine the objects and distance for all
//perpendicular lines on the line with current start and end values.
        firstheight = 0
        distancenumber = 0
        noofpoints = 0
        noofpositivevalues = 0

//This loop runs for every perpendicular on current angled line and
 //finds distances of objects in those perpendicular lines.
        Do

            tempstartrow +=1
            tempstartcol+=1

        IF(layerdata[tempstartrow ][tempstartcol] > 0)
//Check for the objects in virtual lines.  Any value greater than zero
```

```
 //represents object and values less than zero represent background.
                noofpositivevalues+=1
               IF(firstheight==0 )
//If we encounter the first positive value in the virtual line, it is
//an indication of an object in current line
                   firstheight = 1
               EndIF

 // Note: In our images objects' heights are continuously growing and
 //then falling down mostly making a parabolic like shape.
//In case of positive value the check below sees if the current point
 //in image is taller than its previous and next point in height so if
 //it is taller than previous and next it's considered as peak of
 //current object.
           IF(layerdata[tempstartrow -1][tempstartcol-1] <
           layerdata[tempstartrow ][tempstartcol] && layerdata[tempstartrow]
           [tempstartcol ] >layerdata[tempstartrow +1][tempstartcol +1] )
               IF(firstheight==1)
                    IF (noofpositivevalues > noofpoints  )
         //This check avoids the multiple ups and downs in same object
          //by checking for background points between two peak points.
                       noofpoints+=1
                  Else

//In case of else we are sure that second peak is reached so now to
 //record the distance between two peaks set the different variables and
  //move on.
                      distancenumber +=1
                      noofpoints+=1
                      noofpositivevalues = 0

//This will print the value of all heights-->fprintf fileobject, "The
 //distance between object #" + num2str(distancenumber) + " and object
  //#" +  num2str(distancenumber + 1) + "  is :" + num2str(noofpoints *
   //sizeofpixel) + " meter\r\r\n\n\r"
                 distances [distancecounter] = (noofpoints * sizeofpixel)
                 totalofdistances+=  (noofpoints*sizeofpixel )
                 distancecounter+=1
```

```
                    noofpoints = 0
                EndIF
            EndIF
//The else operation will be performed when the current point on image
//is not taller from its previous and next so we consider it as  point
//between peaks not a peak.
            Else
            IF(firstheight==1)
                noofpoints+=1
            EndIF
            EndIF
        Else
//Else if the current point on image is not positive so check if it's
//between two peaks and it will be a central point considered for
//distances between peaks.
        IF(firstheight==1)
            noofpoints+=1
        EndIF
        EndIF

        while (tempstartrow !=tempendrow && tempstartcol != tempendcol)
//End of code for every perpendicular line.

//check to go control the starting point for every next line parallel
//to the provided angle.
        IF (startcol == 0 && startrow == 0)
            checkend = 1
        EndIF

//The checks below provide us with starting and ending points of next
 //line parallel to the previous line based on given input degree of
 //angle.

        IF(startcol > 0)
             startcol -=1
        Else
            startrow+=1
        EndIF
```

```
        IF(endrow <dimension)
            endrow+=1
        Else
            endcol-=1
        EndIF
 //This check is applied because the check does not work for while loop
        IF((startrow == (dimension-1) ) ||  (endcol  == 1))
            break
//breaks the loop on successful matching of the check.
         EndIF
    While (1)

    return totalofdistances/distancecounter

End
//End of Function

//This function is used to change the name of any image by user defined
//services.
Function renameImage()
    setdatafolder root:Images
    string imagename
    Prompt imagename ,"Select an Image from List",popup,WaveList("*", ";", "")
    doPrompt "Select Name to remove Extension ",Imagename
    renamepict imagename, Imagename
end
//end of unused function


// This is a utility function that calculates the nearest square for the
// input size and we use this service by providing original dimensions.
Function calculateSquareSize(size)
    variable size
    variable power =0
        do
            if (size >=2)
                if (mod(size ,2) ==1)
                    size = size-1
```

```
                endif
            endif
            size = size/2
            power +=1
        while(size > 1)
        size =1
        do
            size = size*2
            power -=1
            while (power >0)
    return size
    killvariables /A/Z
end
// End of Utality service / Function

// This function asks the user to select an image from the image list
//and if the dimensions of the image are not in form of 2 to power n it
//converts that
// image into 2 to the power n dimension for easier and quick image
//processing.
Function makeImageSquare()
    setdatafolder root:Images
// set folder pointer to images folder inside root.
    variable row, column, size,rowindex,columnindex
    string imagename
    Prompt imagename ,"Select an Image from List",popup,WaveList("*", ";", "")
// load all Images in the current folder that is root:Images
    doprompt "select an Image from List below",imagename
// Return the image name into Imagename variable selected from the
//dropdown list
    if (V_Flag)
        return -1 // User canceled
    endif
    if ( waveExists($imagename) == 0 )
      // check if the File name exists
        DoAlert 1,"File does not exist. Please try again"
```

```
        return -1
    else
        row = DimSize(root:Images:$imagename,0 )
//Get the rows and column of the image selected by user
        column = DimSize(root:Images:$imagename, 1 )
        size = min(row,column)
// get minimum of rows and columns (dimension) for the current user
//selected image
        size = calculateSquareSize(size)
// call to the utility service function that calculates 2 to power n
//dimension for current image
        if ( waveExists(LayerData) == 1 )
// If there is already an image with name LayerData delete it
            killwaves LayerData
        endif
        Make /O/N = (size,size) LayerData
//  Create a new Image with name LayerData
// Copy the 2 to power n data from the selected image into layer data
//created inside the current  procedure
        wave tempLayerData =$imagename
        for (rowindex=0 ; rowindex <size;rowindex+=1)
            for (columnindex=0 ; columnindex <size;columnindex+=1)
                LayerData [rowindex][columnindex] =
                tempLayerData[rowindex][columnindex]
            endfor
        endfor
    endif

    Killvariables /A/Z
// Delete all unrequired variables and strings to release memory after
//running this function
    KillStrings /A/Z

End

// This function sets the scale for an image that is a non-Igor image
//according to user desired inputs
```

```
Function SetScaleForImage()
    string scansize
    string imagename =""
    scansize = "0.0296e-9"
// Default pixel size which will be over write by user defined pixel
//size information
    variable row,column
    setdatafolder root:Images
    Prompt imagename ,"Select an Image from List",popup,WaveList("*",";", "")
    prompt scansize ,"Enter the size of actual Pixel"
    DoPrompt "Enter the file name and size of actual Pixel",imagename ,scansize
    if (V_Flag)
        return -1 // User canceled
    endif
    if ( waveExists(root:Images:$imagename) == 0 )
        DoAlert 1,"Try BY providing correct file that exists in imagefolders"
    else
// If user provides correct information so this section of code
//calculate and apply the scale for the user selected image
        row = DimSize(root:Images:$imagename,0 )
        column = DimSize(root:Images:$imagename, 1 )
        SetScale X,0,str2num(scansize)*row,root:Images:$imagename
        SetScale Y,0,str2num(scansize)*column,root:Images:  $imagename
        Note root:Images:$imagename, "Scane size  ="+scansize
    endif
    Killvariables /A/Z
    KillStrings /A/Z
End
//End of set scale service /Function

// This function is used to invert the data of SEM images to make it
//compatable and graphically identical to AFM Images. The effect of
//this service is to invert
// The image data i.e. in SEM data the background is higher than
 //objects in terms the MFP-3D reads the data. So the service Bring the
  //objects on top of background.
```

```
Function InvertLayer()
    variable row,column,rowindex,columnindex
    wave myLayerData = root:Images:LayerData
    row = DimSize(root:Images:LayerData,0 )
//Layer data should be square so row and column are dimenion of the
//image and must be same
    column = DimSize(root:Images:LayerData, 1 )

    for (rowindex=0 ; rowindex <row;rowindex+=1)
        for (columnindex=0 ; columnindex <column;columnindex+=1)
            myLayerData [rowindex][columnindex] = (myLayerData[rowindex]
            [columnindex]  -256) *(-1)
 //Putting the data back into layer data after changing pixels
//Values with the mentioned formula and thenique
        endfor
    endfor

end

// This service take care of images that are not Igor native images.
 //For Example SEM images, it allows the user to select SEM image
// and puts it into layer data and and then inverts the negative for
//further easy processing and Igor image like view.
Function ManageExternalImage()
    string imagename , conversionmechanism
    variable row,column,rowIndex,columnindex
    setdatafolder root:Images
    Prompt imagename ,"Select an Image from List to Import into Igor
    Format",popup,WaveList("*", ";", "")
// load list of images in image folder
    prompt conversionmechanism ,"Enter the Type of Conversion" popup,
    "Invert/Flip Image;Copy Image;"
// give user option for operation
    DoPrompt "Enter the file name and Conversion Mechanism",imagename ,
    conversionmechanism
//Ask user for image format mechanism
    if (V_Flag)
```

```
            return -1 // User canceled
        endif
        if ( waveExists(root:Images:$imagename) == 0 )
            DoAlert 1,"Try by providing correct file name exists in image folders"
        else
            row = DimSize(root:Images:$imagename,0 )
            column = DimSize(root:Images:$imagename, 1 )
            SetDataFolder root:Images
            Make /B/U/O/N = (row,column) myLayerData
            wave tempLayerData =$imagename
            if (Stringmatch(conversionmechanism,"Invert/Flip Image")==1)
                for (rowindex=0 ; rowindex <row;rowindex+=1)
                    for (columnindex=0 ; columnindex <column;columnindex+=1)
//Putting the data back into layer data after changing pixels  Values
//with the mentioned formula and technique
                        myLayerData [rowindex][columnindex] =
                        (tempLayerData[rowindex][columnindex] -256) *(-1)
                    endfor
                endfor
            else
                myLayerData =tempLayerData
// put the image directly into layer data without inverting it
//according to user options
            endif
        endif

// delete the undesired variables strings and waves to return memory to
 //the processor for further usage.
        Killvariables /A/Z
        KillStrings /A/Z
        KillWaves tempLayerData
End
// End of user define procedure that takes care of external images.
//Function multiply an image over an other image.
macro MultiplyLayer()
    MultiplyImages()
end
```

```
Function MultiplyImages()
    string imagenamefirst, imagenamesecond , conversionmechanism
    variable row,column,rowIndex,columnindex
    setdatafolder root:Images
    Prompt imagenamefirst ,"Select an Image
    multiplication",popup,WaveList("*", ";", "")
// load list of images in image folder
    DoPrompt "select it man", imagenamefirst
//Ask user for image format mechanism
    if (V_Flag)
    return -1 // User canceled
    endif
    if ( waveExists(root:Images:$imagenamefirst) == 0 )
    DoAlert 1,"Try by providing correct file that exists in image folders"
    else
        row = DimSize(root:Images:$imagenamefirst,0 )
        column = row
        SetDataFolder root:Images
        Make /B/U/O/N = (row,column) myLayerData
        wave tempLayerData =$imagenamefirst
        wave tempSecLayerData = $ imagenamefirst
//wave tempSecLayerData = $ imagenamesecond
        for (rowindex=0 ; rowindex <row;rowindex+=1)
            for (columnindex=0 ; columnindex  <column;columnindex+=1)
                tempLayerData[rowindex] [columnindex] = tempLayerData[rowindex]
                [columnindex] * tempSecLayerData[rowindex][columnindex]
            endfor
        endfor
    endif
End //End of Procedure File
```