1-1-2010

A Novel Image Processing Platform Development For Preprocessing Images With Two-Phase Microstructures Containing Isolated Particles

Yuxiu Zhang
Wayne State University,

A NOVEL IMAGE PROCESSING PLATFORM DEVELOPMENT FOR PRE-PROCESSING IMAGES WITH TWO-PHASE MICROSTRUCTURES CONTAINING ISOLATED PARTICLES

by

YUXIU ZHANG

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2010

MAJOR: MECHANICAL ENGINEERING

Approved by:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

I would like to dedicate my thesis work to my family whose love, moral support, motivation and encouragement has been helping me overcome all the difficulties that I encountered during this work.

ACKNOWLEDGMENTS

I am very much indebted to my advisor-Professor Xin Wu for offering me such a wonderful opportunity to carry on my master study and for his enlightenment and encouragement throughout my dissertation work.

I would like to express my deep appreciation to my committee members and Dr. Yi Liu , and fellow students who have provided support during my master study. It is their mentoring, training, discussion and collaboration that made my student days a fun and rewarded journey.

Last but not least, I am especially grateful to my husband Xinyu Du for all he did for me throughout these years.

.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 The Research Background

In automotive industry, the automotive underbody, structural, and body panels requires higher strength and lower mass steels. These steels have been developed from traditional low strength steels, conventional high-strength steels (HSS) to advanced high-strength steel (AHSS) which are compared and shown in Fig. 1.1.[1] From Fig. 1-1, the AHSS can be observed to possess higher tensile strength and fair formability [2]. For instance, the dual phase (DP) steels, one of AHSS, has higher formablity than Carbon-manganese (CMn) steels, one of HSS, at the same level of tensile strength. Therefore, the AHSS including DP steels, transformation induced plasticity (TRIP) steels, complex phase (CP) steels and others, as light weight materials, are becoming more and more important to vehicle structures for the sake of the great combination of strength and ductility.

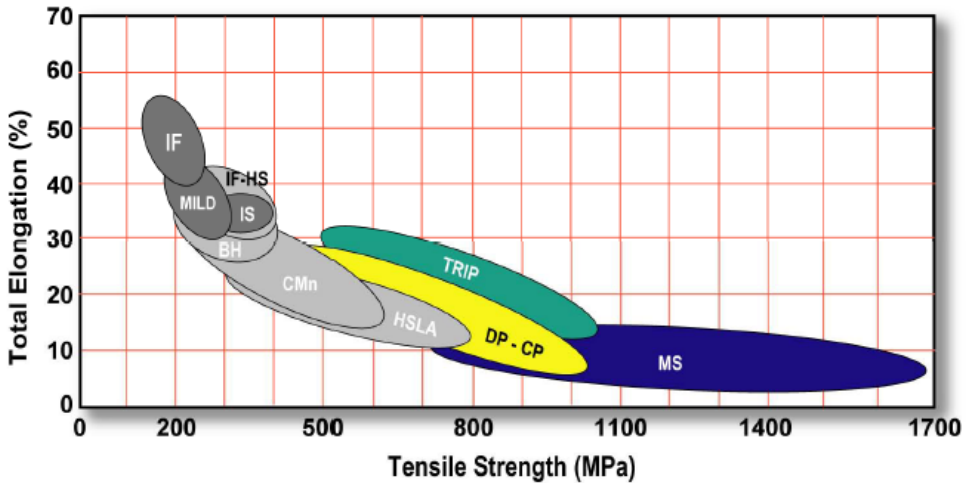


Figure 1-1 The schematic comparison of various steels among low strength steels (dark grey), conventional HSS (light grey) and AHSS (colorful)[1]

The principal difference between conventional HSS and AHSS is their microstructure. Conventional HSS are single phase ferritic steels (e.g. bake hardenable steels) while AHSS are primarily multi-phase steels which contain ferrite, martensite, bainite and/or retained austenite in quantities sufficient to produce unique mechanical properties.

In order to describe the AHSS products, we use the classification system in ULSAB-AVC (UltraLight Steel Auto Body – Advanced Vehicle Concepts) program for the whole thesis from several kinds of terminology methods used to classify steel products. For this classification system, each steel grade is identified by metallurgical type, yield strength (in MPa), and tensile strength (in MPa). For example, DP 500/800 means a dual phase steel type with 500 MPa minimum yield strength and 800 MPa minimum ultimate tensile strength.

Among variable types of AHSS, we will focus on DP steels. DP steels are produced by controlled cooling from the austenite phase (in hot-rolled products) or from the two-phase ferrite plus austenite phase (for continuously annealed cold-rolled and hot-dip coated products) to transform some austenite to ferrite before a rapid cooling transforms the remaining austenite to martensite. DP steels consists of a ferritic matrix containing a hard martensitic second phase in the form of islands. The soft ferrite phase is generally continuous, giving these steels excellent ductility. When DP steels deform, strain is concentrated in the lower-strength ferrite phase surrounding the islands of martensite, creating the unique high work-hardening rate exhibited by DP steels. In DP steels, carbon enables the formation of martensite (the hard second phase) at practical cooling rates to increase the hardenability of the steel. Therefore, the volume fraction of

martensite is a key parameter to evaluate the steel strength. Obviously, the higher the volume fraction is, the more strength the DP steels are. To calculate the volume fraction of DP steels, an effective way is to analyze the microstructure image. Hence, microstructure characterization is an important method to investigate the property of AHSS.

There are two main approaches of image acquisition which are scanning electron microscope (SEM) and optical systems using visible light. When the magnification requirement is not very high, the latter approach is generally employed. For instance, a compound optical microscope can produce a magnified image of a specimen up to 1000×. For this application, the former approach is always desired.

A Hitachi S-2400 SEM (Shown in Fig. 1-2) is used by us in Laboratory of Analytical Electron Microscopy (LAEM) to capture the sample surface of DP steels. SEM is a type of electron microscope that images the sample surface by scanning it with a high-energy beam of electrons in a raster scan pattern. The electrons interact with the atoms that make up the sample producing signals that contain information about the sample's surface topography, composition and other properties such as electrical conductivity [3]. The SEM is equipped with Dispersive X-Ray Spectrometry (EDS), Robinson Backscattered Electron (BSE) Detector, Detector of Electron Backscattered Diffraction (EBSD)/Orientation Imaging Microscope (OIM). Magnification in the SEM can be controlled over a range of up to 300,000 X with the resolution up to 4 nm. Unlike optical and transmission electron microscopes, image magnification in the SEM is not a function of the power of the objective lens. In our application, we took the SEM image

with the magnification of 500 X, 1500 X and 5000 X, respectively for the well-prepared samples from DP steels. .

Figure 1-2 Hitachi S-2400 SEM in Laboratory of Analytical Electron Microscopy (LAEM)

1.2 The Image Processing Methods for Microstructure Analysis

With the raw SEM image the image processing methods should be used to extract the statistic information for martensite (e.g. the volume fraction, the distribution of martensite and so on). Generally, four steps need to be executed which are image filtering, image segmentation, binary image treatment and measurement or modeling[4].

The aim of image filtering is to improve the quality of images or to enhance some desired features. In most image processing applications including ours, it is inevitable to expose some noise points on raw SEM images. These points are apparently not belonging to the martensite but carried in by environment factors. These points should be eliminated from images before further processing in order to reduce the analysis error. Generally, noise appears in frequency much larger than that of the pertinent

information. So we can use low-pass filter to eliminate the noise. On the other hand, the contour information of the martensite which is used to recognize the outline is necessary but sometimes is very hard with the similar grey tone levels as the background (ferrite). Then enhancing the outline of martensite is helpful to extract martensite information. We can use high-pass filter to realize it. Currently, there are several image filtering methods available, e.g. linear filter, rank filter [5] or morphological filters[6-7].

The raw SEM image in our application is grey tone image, while the expected resulting image contains two phase, ferrite and martensite that is a binary image. Therefore we can use threshold method to segment the image. This step is extremely important as it corresponds to the maximum loss of information. So we have to pay a lot of attention to the choice of the method to be used. There exist a lot of thresholding methods from the literatures [5-6, 8-11]. In [11], the author presented 11 threshold methods based on the histogram. Here, we will address the following five representative methods. The first one is threshold by pre-defined levels. In this case, an upper and lower value of grey tone level is pre-defined separately for segmentation. All image pixels with grey tones belonging to this domain will enable to create a binary image representing the extracted information. For a raw image, several domains can be defined which is called multiple threshold. Generally the value of the pre-defined threshold is determined manually. Another method is by maximization of the interclass variance. With this method, the histogram is divided into a certain number of classes. When the position of the class limit varies, the value of the variance changes and the position which maximize that value is determined [9]. The third method is by entropy

maximization. In this case, the histogram is divided in several classes. This threshold by entropy maximization gives good results when there are few objects [12]. The fourth approach is by moment-preserving. This method is proposed by Tsai [13] consists of assigning a well-determined grey tone level to each class, so that the first statistical moments will be identical to that of the initial image. This method provides good results with low contrast images. The last method is by contour information. The threshold by information on the contour is also possible from an algorithm computing the image contrast [10]. With this method the threshold retained gives the maximum mean contrast. This type of method is efficient if the contours are related to only one transition, so to a two-phased system.

With the binary image from segmentation, it's often necessary to eliminate artifacts or to fill holes that are generated from segmentation process[4]. For this step, the mathematical morphology is a popular tool. For instance, we can use geodesic operators to fill the holes which have no physical meanings[14]. Another useful operator is skeleton (SK operator) for thinning purpose[15].

The last step is measuring and modeling[16-21]. After all image processing we mentioned above, the interesting information contained in the image can be extracted in parametric or function form. The interesting parameters include volume fraction, surface fraction, linear fraction etc and the functions include covariance function, morphological functions, granulometric functions etc. These parameters or functions can be obtained from the calculation of the binary images using variant mathematical methods. Moreover, we can model the object from these parametric characterizations to explore the stereological nature.

1.3 The Existing Software on Image Processing

There exist numerous software used in image processing, which fall into three categories-research code, image processing toolkits and mathematical toolkits. Some relative software will be listed below.

1) Research code. Venkatesh and Rosin presented Automatic edge thresholding code to extract edge field described in their 1995's paper[22]. Edge list approximation code was also proposed by Rosin which utilized the nonparametric segmentation of curves into various representations[23]. Color Machine Vision Library is a fast color-segmentation software library provided by Jim Bruce which is able to threshold and calculate connected components of 640x480 60fps video streams entirely in software. It is a free low level color vision processing library available in [24]. Image Segmentation is a C++ implementation to efficiently graph-based segment image[25]. Segmentation of Skin-Cancer Images is an algorithm implementation to segment skin cancer images or other pigmented lesions. It converts a color image into an intensity image first and segments the image by intensity thresholding. Then it refines the segmentation using image edges. Double thresholding is used to focus on an image area. Image edges are then used to localize the boundary in the area and fitted by a closed elastic curve[8].

2) Image processing toolkits. Facor image filter is a commercial software that detects image structures and distinguishes them from noise in digital images [26].It is able to suppress noise or to emphasize image structures. CImg is a free C++ image processing library containing most classical image processing algorithms[27]. Edge Detection and Image Segmentation (EDISON) is an open source software that performs

mean shift based discontinuity preserving filtering, image segmentation and edge detection with embedded confidence[28]. EVision is a commercial image processing and analysis toolkit on Windows operation system. It provides all familiar tools in image processing such as image linear and morphological filtering, projections, profiles, geometric transforms, color conversions, blob analysis, pattern matching as well as application libraries such as OCR and matrix code reading[29]. NeatVision is a Java-based free image analysis and software development environment. It contains over 200 image and general data processing algorithms. Furthermore, users can extend the core library using the plug-in like developers interface[30]. Scion Image is a free image processing software based on NIH image[31]. It was used in our project and will be introduced in the next section. THBComponentware is a Windows-based commercial software[32]. It is able to align, stretch, scroll and zoom images. Its image processing functions include resize, rotate, crop, mirror, invert, filter, sharpen, blur, color conversion etc. UTHSCSA image tool is a free image processing and analysis program for Windows platform[33]. It can acquire, display, edit analyze, process, compress gray or color images. It supports standard image processing functions such as contrast manipulation, sharpening, smoothing, edge detection, median filtering and spatial convolutions. Visiopharm Integrator System[34] is a commercial, database-driven image analysis system mainly for medical image analysis. It is equipped with a wide range of modules for viewing, montage, and segmentation of images. Measurement results are automatically stored in the built-in database making the data management an integrated part of the work flow. XMegaWave is a free image processing environment. It includes

edge detection, segmentation, morphological filters. The users can implement his own processing algorithm in C and integrate it into XMegaWave[35].

3) Mathematical Toolkits. The most famous one is Matlab. Since it plays an important role in this project, we will introduce it in detail later. Scilab is a free scientific software package for numerical computations[36]. It includes image processing design toolbox and image and video processing toolbox for image processing. Similarly to Matlab, it contains most common image processing algorithms including filter, matching, thresholding, histogram, blob analysis etc. Computational Geometry Algorithms Library (CGAL) is a C++ library of geometric primitive and standard data structures and algorithms used in computational geometry. CGA library contains basic geometric primitives such as points, vectors, lines and operations such as intersections and distance calculation. The data structures and geometric algorithms include convex hull, planar map, multidimensional query structures and so on[37]. StatLib is a statistical software to calculate the statistical features in image processing[38].

More image processing software and detailed information can be found in [39] and [35]. Since these software or code can not provide the same functions provided in this work, we will not describe them one by one.

1.4 The Drawbacks of Image Processing Methods Using in the Project

Although the above software or algorithms can be used in microstructure analysis [40-44], the results of the volume fraction of martensite are still imprecise. For example, the first step of image processing is filtering. If we employ the median filter

(3X3), a kind of rank filter, to remove noise for the raw SEM image, the resulting martensite area will be changed apparently.

For image segmentation step, the current threshold technique also has limitations. In the raw SEM image, the grey tone levels for different martensite area, even in the same image, are always different. If we select a threshold value according to one area, other martensite area after threshold will be larger or smaller.

Therefore, currently we employ such a hybrid method that drawing the contour of martensite area by hand for the first three steps and then analyze the volume fraction etc. parameters using software scion. To draw the contour image, we imported the raw SEM image into a tablet PC. Then start the software of Windows journal. Windows Journal is a notetaking application, created by Microsoft and integrated in the operation system, Windows XP. It allows the user to create and organize handwritten drawings. With the help of Windows journal, the resulting contour image excludes all noises and stands out the outline of martensites. Opening the contour image in the software paint, we can fill the martensite area by black, and convert the color for the whole image. Then a binary image is obtained for information extraction. This method is much more accurate than the mathematical methods. But there are two short-comings. First, drawing a contour image is time-consuming process. Generally for a 5000X image, it takes about 40 minutes to draw all martensite outlines. However for a 1500X image it takes more than 3 hours to draw all martensite outlines. Second, the outlines of many martensite areas in the raw SEM image are difficult to be distinguished. Hence, it's impossible to keep the same routine for all martensites in the same image while drawing. That makes the contour image also inaccurate.

To extract the necessary martensite information, we use the software Scion Image. Scion Image is free image processing and analysis software developed by Scion corporation. It is based on the popular NIH Image on the Macintosh platform. It can acquire, display, edit, enhance, analyze and animate images. It reads and writes TIFF and BMP files, providing compatibility with many other applications, including programs for scanning, processing, editing, publishing and analyzing images. It supports many standard image processing functions, including contrast enhancement, density profiling, smoothing, sharpening, edge detection, median filtering, and spatial convolution with user defined kernels. Scion Image for Windows can be used to measure area, mean, centroid, perimeter, etc. of user defined regions of interest. It also performs automated particle analysis and provides tools for measuring path lengths and angles. The Scion image performs well in our project; however, it is somewhat unstable for particle analysis (martensite area extraction). Furthermore, the function of the software is fixed and limited; it's difficult to achieve more features according to metallurgist's desire.

1.5 The Outline for the Novel Image Processing Platform

Since the extraction of martensite information from raw SEM image is very important and the current methods are unsatisfied, we will develop a novel image processing platform to pre-process the raw images in this thesis.

The first requirement of the new image processing platform is easy to use and develop. Because most metallurgists are not expert on programming and the field of image processing, this feature will be valuable for them to apply this platform in different

applications. The second requirement is the integration and extensibility. In current process, we need to use three software-windows journal, windows paint, and scion image and the analysis results are limited. Therefore, it's more convenient for users if we can include all useful functions in one package and exclude the functions unrelated to microstructure analysis. The third requirement is time-saving and accurate. These two requirements are always contradictive. With the current image processing methods, if we draw the contour of martensite areas by hand, it's more accurate but time-consuming. On the other hand, if we use the mathematical algorithm, the result is inaccurate but the process is fast. Therefore, we need to design a new approach overcome the drawbacks for these two methods.

To satisfy these requirements, we choose Matlab® as our development environment among several development tools. MATLAB® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB® is an interactive system whose basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems with matrix and vector formulations, e.g. image processing problems, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or FORTRAN. Matlab integrates a lot of image processing functions in Image Processing Toolbox which are a great benefit for us to build our own image processing program fast. Matlab also provides provides a set of tools for creating graphical user interfaces (GUIs) which is called GUIDE, the MATLAB® Graphical User Interface development environment. These tools greatly

simplify the process of designing and building GUIs and facilitate the users to operate the software.

To develop a program using GUIDE, there are two steps. The first one is using the GUIDE Layout Editor to lay out all necessary GUI components -- such as panels, buttons, text fields, sliders, menus, and so on -- into the layout area. The second step is program the GUI in Matlab® M-file. GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for all the GUI callbacks -- the commands that are executed when a user clicks a GUI component. Using the M-file editor, it's easy to add code to the callbacks to perform the functions what we want.

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment. The toolbox supports a wide range of image processing operations, including spatial image transformations, morphological operations, neighborhood and block operations, linear filtering and filter design, transforms, image analysis and enhancement, image registration, deblurring and region of interest operations etc. Many of the toolbox functions are MATLAB® M-files, a series of MATLAB® statements that implement specialized image processing algorithms. We can extend the capabilities of the Image Processing Toolbox by writing our own M-files, or by using the toolbox in combination with other toolboxes, such as the Signal Processing Toolbox etc.. With the help of Image Processing Toolbox, a lot of basic image processing functions such as image open, display or save etc. can be easily integrated into our platform.

Although Matlab® provides a good environment for us to develop new platform, the image processing functions built-in Matlab® are still belonging to the mathematical operations that can not help us to utilize experts' experiences to achieve better image processing results as well as save time. To apply these experiences on microstructure analysis, the fuzzy logic [45] [46] can help. Fuzzy logic is such a tool which is a form of multi-value logic derived from fuzzy set theory to deal with reasoning. In contrast with binary sets having binary logic, also known as crisp logic, the fuzzy logic variables may have a membership value between 0 and 1. For instance, we can employ fuzzy logic variables to describe a linguistic variable such as age may have a value such as young or its antonym old. With fuzzy variables, the experts' experience can be described as a fuzzy expert system. Nowadays, there are a lot of papers applying fuzzy logic on image processing [47-55]. However no paper has appeared on microstructure analysis using fuzzy logic approach as much as we know.

In our project, we can describe the experts' experience for contour information with respect to fuzzy logic. Then the experts' judgments used to distinguish the martensite area can be converted to a linguistic inference system instead of the mathematical operations. In this way, we can guarantee the accuracy for the results and save a lot of time for the whole image process.

In the chapters 2 and 3, we will introduce the development for the novel image processing platform in detail and compare our results with the previous results from other methods. Finally, the chapter 4 will conclude the whole thesis.

CHAPTER 2 THE IMAGE PROCESSING PLATFORM DEVELOPMENT

In this chapter, the image processing platform development is described in detail. The Matlab 7 including image processing toolbox is used to develop the whole platform and for users' convenience all programs are able to work in the Matlab Graphical User Interface Development Environment (GUIDE).

2.1 The Frame of the Whole Platform and the Main Program Development

The platform is named Microstructure Image Processing V1.0 and developed in Matlab GUIDE. Therefore, all programs in the platform are associated with a file for interface window (.fig) and another file for program code (.m). The main program is named ImageProV1 (.fig and .m) (shown in Fig. 2-1) which provides an entry to the whole software package. All functions in the software package can be called from the menu bar of the main program. The menu bar is created via menu editor (shown in Fig. 2-2) in GUIDE. Each function has a call back function existing in the main program. If the function is not very complex e.g. file open, the whole function is coded in the main program. Otherwise, in the call back function, we call another program to complete the function, e.g. fuzzy rule1.

In this platform we have six classes of functions listed in the menu bar. File class includes the file I/O functions and program termination functions i.e. SEM image file open, contour image file open, image save, program reset, and program exit. View class includes the functions of original image information, zoom image, and the histogram of image for users' further check. Threshold/filter class provides the methods to convert the SEM images to BW image and filter the resulted BW image which are

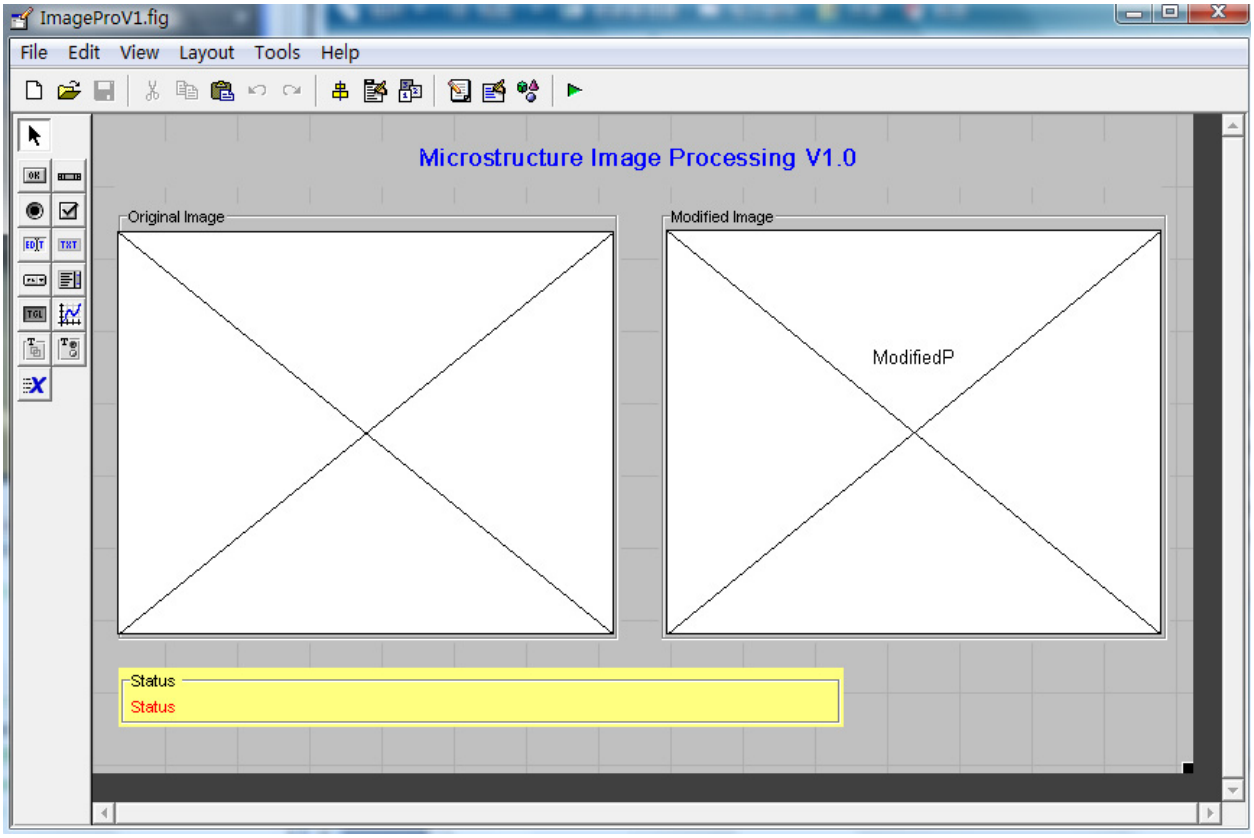


Figure 2-1 Main Program in Matlab GUIDE program. Seven GUI Components, two axes (for original image and modified image) and two strings (title and status) and three panels are placed from the side tool bar.

automatic global threshold, automatic local threshold, manual global threshold, manual local threshold, median filter and adaptive filter. AI optimization class includes two fuzzy methods to generate BW image and refine the image respectively. Measurement class includes the functions to calculate the volume fraction and other information about the final image. Help class provides the software version and user manual. We will explain all functions in detail in the following sections.

Besides the menu bar, the main program includes two axes to display the original image and the modified image respectively and one text box to show the current operation status. To display the original image and the modified image, we firstly place

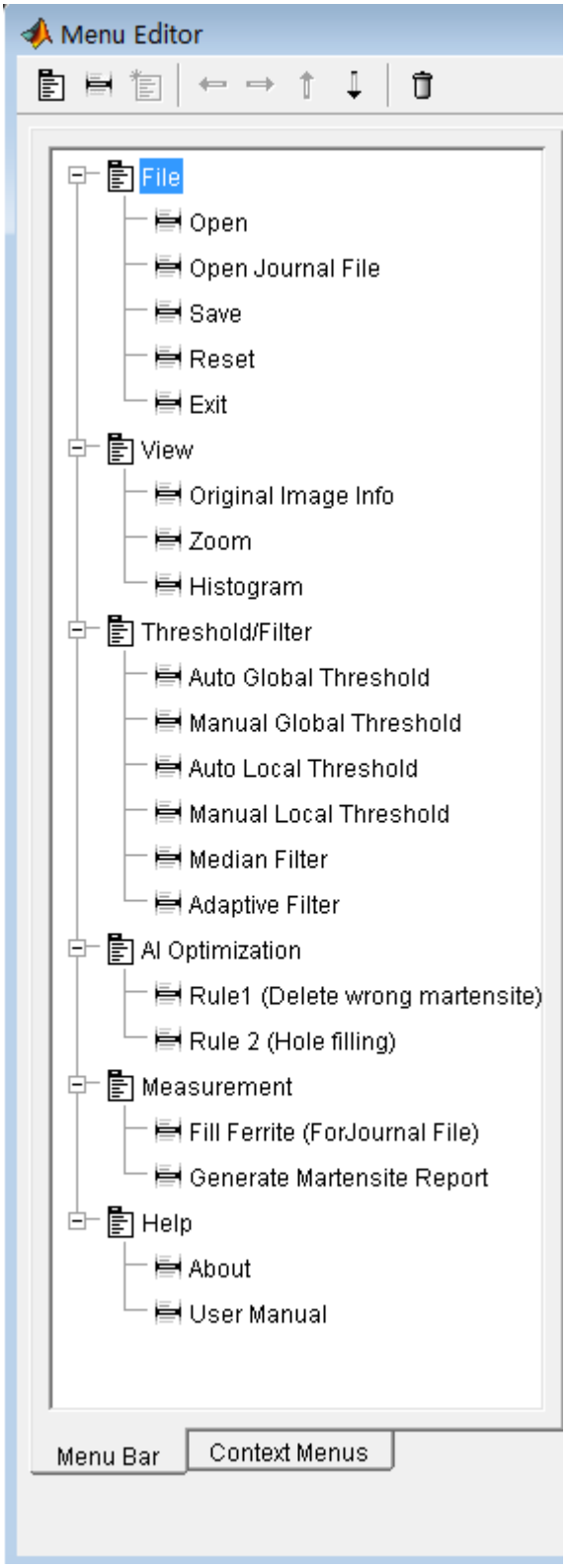


Figure 2-2 All functions in the software package shown in the menu editor

two axes in the main program in GUIDE. The “tag” property is used to distinguish each axe and the variable handles.tag can be used in the .m file. For example, the axe for original image (shown in Fig. 2-1) has the tag named handles.OriginalP. Then we employ the Matlab function axes() to locate the object axe and function image() to display the desired image on the object axe. For instance, the codes,

```
axes(handles.OriginalP);
image(handles.X0);
```

are used to display the image “handles.X0” on the axe “Original Image”.

For text box in main program, we use set() command to update the current operation status all the time.

2.2 The File Menu

In the File menu, there are five programs, Open, Open Journal File, Save, Reset and Exit (Fig. 2-3). We will introduce these programs one by one in this section. The “Open” program employs “uigetfile()” function to get the file name and path name for objective image. “imread()” function is used to read the original SEM image information from the hard disc. Here the original SEM image formats which are supported by the platform are listed in Table 2-1.

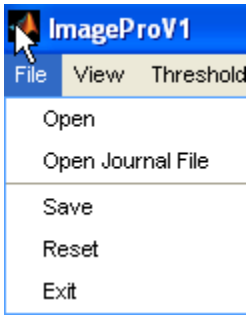


Figure 2-3 File menu

Table 2-1 Support Formats list for Microstructure Image Processing Platform V1.0

Format	Full Name	Variants
'bmp'	Windows Bitmap (BMP)	1-bit, 4-bit, 8-bit, 16-bit, 24-bit, and 32-bit uncompressed images and 4-bit and 8-bit run-length encoded (RLE) images
'cur'	Windows Cursor Resources (CUR)	1-bit, 4-bit, and 8-bit uncompressed image
'gif'	Graphics Interchange Format (GIF)	1-bit to 8-bit images
'hdf'	Hierarchical Data Format (HDF)	8-bit raster image data sets, with or without an associated colormap, and 24-bit raster image data sets
'ico'	Windows Icon Resources	1-bit, 4-bit, and 8-bit uncompressed images

	(ICO)	
‘jpg’ or ‘jpeg’	Joint Photographic Experts Group (JPEG)	Any baseline JPEG image or JPEG image with some commonly used extensions, including: Image Type Bitdepth Compression Grayscale 8- or 12-bit lossy grayscale 8-, 12-, or 16-bit lossless RGB 24- and 36-bit lossy or lossless
‘pbm’	Portable Bitmap (PBM)	1-bit images using either raw (binary) or ASCII (plain) encoding
‘pcx’	Windows Paintbrush (PCX)	1-bit, 8-bit, and 24-bit images
‘pgm’	Portable Graymap (PGM)	ASCII (plain) encoding with arbitrary color depth, or raw (binary) encoding with up to 16 bits per gray value
‘png’	Portable Network Graphics (PNG)	1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit indexed images; and 24-bit and 48-bit RGB images
‘pnm’	Portable Anymap (PNM)	PNM is not a file format itself. It is a common name for any of the other three members of

		the Portable Bitmap family of image formats: Portable Bitmap (PBM), Portable Graymap (PGM) and Portable Pixel Map (PPM).
'ppm'	Portable Pixmap (PPM)	ASCII (plain) encoding with arbitrary color depth or raw (binary) encoding with up to 16 bits per color component
'ras'	Sun Raster (RAS)	1-bit bitmap, 8-bit indexed, 24-bit true color and 32-bit true color with alpha data
'tif or 'tiff'	Tagged Image File Format (TIFF)	Any baseline image, including 1-bit, 8-bit, and 24-bit uncompressed images; 1-bit, 8-bit, and 24-bit images with packbits compression; 1-bit images with CCITT compression; and 16-bit grayscale, 16-bit indexed, and 48-bit RGB image
'xwd'	X Windows Dump (XWD)	1-bit and 8-bit ZPixmaps, XYBitmaps, and 1-bit XYPixmaps

There are four types of images used in the field of image processing, which are binary image, intensity image, indexed mage and true color image. A binary image only contains pixels with the value of 0 or 1 interpreted as black and white, respectively. It's

also called bi-level image. Since our aim is to get the information for two phases, martensite and ferrite, the binary image should be the resulting image format. An intensity image contains pixels of class uint8, uint16 or double which specify the intensity values. Hence, the intensity image is also called grayscale image. An indexed image is also called pseudo color image because its entire pixel values are not double color value like RGB image, but an index. The index can be logical, unit 8, uint16 or double directing into a double color map array called palette. In this way the indexed color image requires less space for storage comparing with RGB image. A true color image is represented by 3 m-by-n array of class uint8, uint16 or double. Each array defines the intensity value for basic colors. Therefore, it's known as an RGB image.

In our project, the original SEM images are generally RGB images. In the end of the program "Open", we use Matlab function "image()" to display it.

Program "Open Journal File" is similar to the program "Open" except for the resource file. Here, we open the contour images from windows journal. This feature is useful for some complex images that can not be recognized by this platform.

Program "Save" is to save the current result to the hard disc. "imwrite()" function is used to save image result. We can use "Save" command for the intermediate result or final result.

Program "Reset" set all systems variable to be 0 and clear the both figures in the main program. Here the "cla" function is used for clear an image.

Program "exit" is to close the platform. Here three Matlab functions are related which are uiwait(), uiresume() and delete(). The uiwait() function is in the opening

function to make the main program wait for user's response. The uiresume() function is in the call back function "exit". When we call this command, the program will jump out of the uiwait() status and continue to output function. In the output program, the delete() function will be called to close all windows for the whole platform.

2.3 The View Menu

The VIEW menu shown in Figure 2-4 is to provide some tools for image details. "Original Image Info" provides the information for original image, e.g. image type, directory, size etc. using Matlab function imageinfo(). "Zoom" is to scale the original image or the modified image using Matlab function imtool. One can open two windows to check the local detail for the original image and the modified image respectively to compare with them. "Histogram" provides the histogram information for original image or modified image by Matlab function imhist().

Figure 2-4. View menu

From the development of the File menu and the View menu, we are strongly impressed by the power of Matlab. It's really easy to develop these basic image processing functions using Matlab.

2.4 The Threshold technique

In Scion Image or others software, we have to adjust threshold value manually. In this platform, we provide both automatic threshold method and manual threshold method. The automatic threshold method, we use Matlab function graythresh() to determine the threshold value automatically. Here the graythresh() function implement Otsu's method [9] which chooses the threshold value to minimize the introclass variance of the black and white pixels. With the threshold value, we can use im2bw() function to convert the original RGB image to a binary image. When we use function image() to display the resulting binary image, we need to convert the binary image to its corresponding RGB image. Here two Matlab functions gray2ind() and ind2rgb() are used for the conversion.

Although the automatic threshold method facilitates the selection of threshold value and unifies the threshold criteria for different images, the resulting binary image is still not as good as it from manually contouring image. One important reason is the image intensity for original SEM image is not uniform. Therefore, a possible improvement is to divide the original image into different areas. Then we can select the threshold value for different areas. In this platform, we implement two methods according to this idea. One is determining threshold value for local area automatically and another one is determining threshold value for local area manually. To realize these two methods, we provide a new program window to allow users to decide the local area number. The array for the original image is accordingly divided. From the results in the next chapter, we can prove this method is effective.

2.5 The Filtering Technique

Two filtering methods are integrated into this image processing platform—median filter and adaptive filter. We create two GUI programs along with a new window to adopt the parameters for median filter and adaptive filter respectively. Fig. 2-5 shows the median filter window. The filter function is realized in the call back function of accept button (Fig. 2-5).

The median filter [56] performs median filtering of image array in two dimensions. Each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image. In Matlab, we employ medfilt2() function to realize the median filter. The drawback of this filter is that it pads the image with 0's on the edges, so the median values for the points within [m n]/2 of the edges might appear distorted.

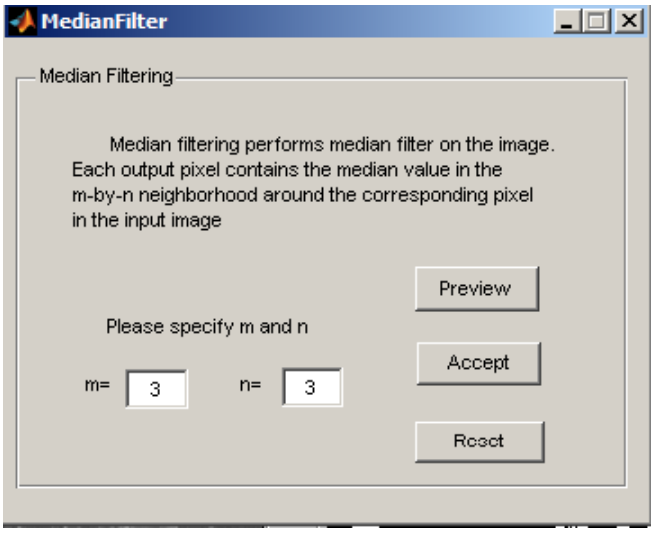


Figure 2-5. Median filter window

The adaptive filter in Matlab applies Wiener filter (Matlab function wiener2()) to an image adaptively, tailoring itself to the local image variance[56]. Where the variance is large, wiener2() performs little smoothing. Where the variance is small, wiener2() performs more smoothing. This approach often produces better results than other linear filtering methods. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image. In addition, there are no design tasks; the wiener2() function handles all preliminary computations and implements the filter for an input image. wiener2(), however, does require more computation time than linear filtering. wiener2 works best when the noise is constant-power ("white") additive noise, such as Gaussian noise. The wiener2() function try to estimate the local mean and variance around each pixel first according to the following equations,

$$\mu = \frac{1}{NM} \sum_{n_1, n_2 \in \eta} \alpha(n_1, n_2)$$

$$\sigma^2 = \frac{1}{NM} \sum_{n_1, n_2 \in \eta} \alpha^2(n_1, n_2) - \mu^2$$

Where η is the *N*-by-*N* local neighborhood of each pixel in the image. Then a pixlewise filter is created using these estimations,

$$b(n_1, n_2) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}(\alpha(n_1, n_2) - \mu)$$

Where v^2 is the noise variance. If the noise variance is not given, the filter uses the average of all the local estimated variances.

Matlab also provides other filtering methods which can be found in the software Scion Image. They can be added to our platform in a similar way. However since our aim is to develop a linguistic inference system to deal with the noise, it's not necessary to integrate or introduce the filtering methods too much.

2.6 Fuzzy Linguistic Inference System

The filtering technique is used to remove noise which would be recognized as martensite possibly. However, the correct martensite areas will be changed a lot when a filter is applied to the original image. For manually drawing the contour for martensite area, the expert can distinguish where the right martensite areas are and which the noises are. Therefore, manually drawing is more accurate than the filtering method. In this section, we will utilize the fuzzy technique to simulate the expert's inference process. In such a way, the new platform can get the good result much faster than manually drawing.

From the local threshold method in section 2.4, we have the binary image for martensite areas as well as a lot of noise points involved (shown in Fig. 2-6). Here we use the algorithm in reference [57] to distinguish all martensite areas. This algorithm is to calculate all connected areas. First run encodes the input image. Secondly, scan the runs, assigning preliminary labels and recording label equivalences in a local equivalence table. Thirdly, resolve the equivalence classes. Last, relabel the runs based on the resolved equivalence classes. The Matlab function BWLabel() is able to realize this algorithm very fast.

With all potential martensite area information, we need to judge each potential area. Generally when the expert draws the contour of martensite area, he/she will make the judgment according to the martensite size. If a potential martensite area is very large relative to other martensite area, it can be judged as a martensite area. In our platform, we use a program with a GUI window (Fig. 2-7) to simulate such a linguistic inference process. Therefore, in our platform, we use these sixteen fuzzy rules to make the decision.

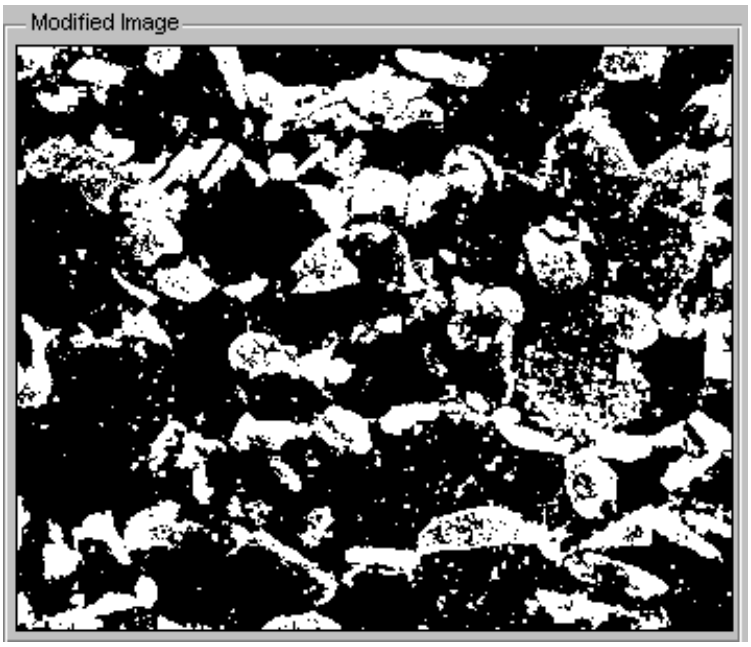


Figure 2-6. The binary image after threshold without filtering process.

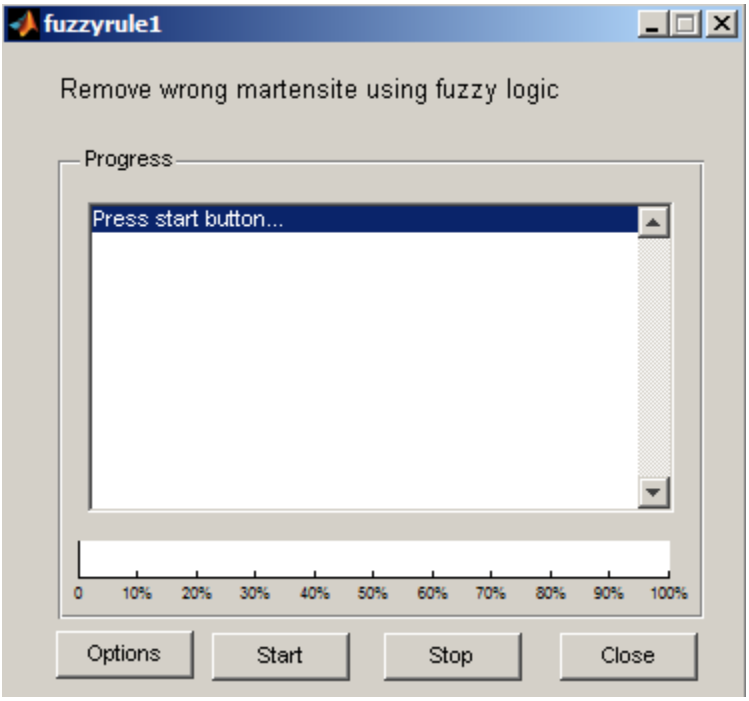


Figure 2-7. The window for removing wrong martensites.

Here, **the estimated martensite number**, **estimated wrong martensite size** and **the size fraction** are fuzzy variables. Their values are linguistic label like some or small representing by trapezoidal fuzzy sets. The **estimated martensite number** has four fuzzy values *a few*, *some*, *many* and *numerous* shown in Fig. 2-8. The estimated wrong martensite size and the size fraction have four fuzzy values shown in Fig 2-9.

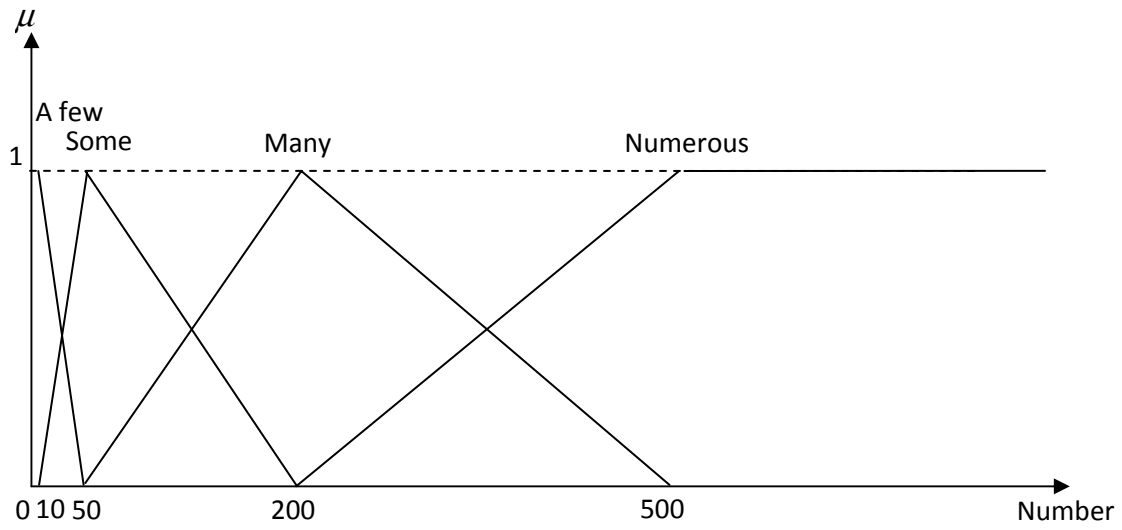


Figure 2-8. The definition of estimated martensite number.

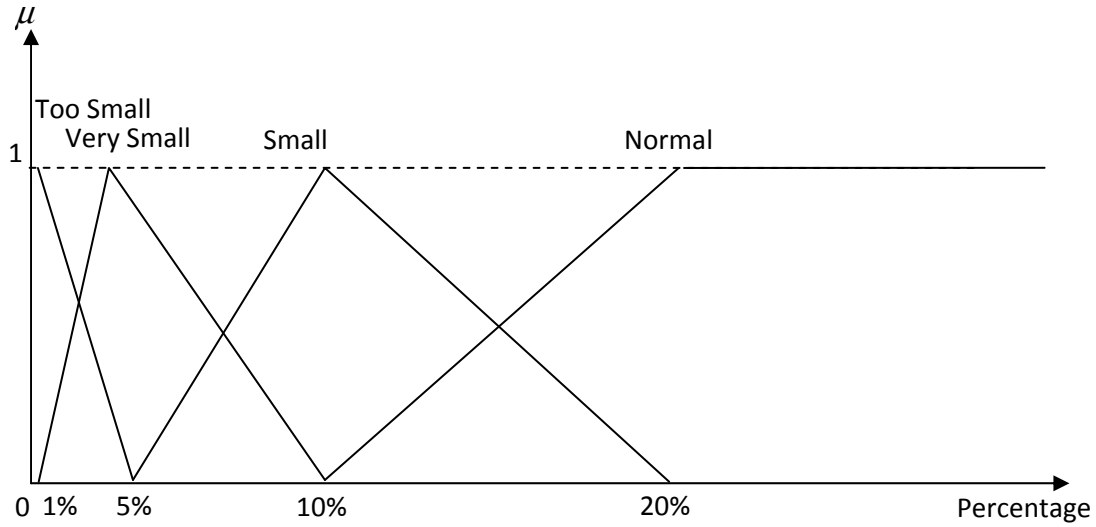


Figure 2-9. The definitions of estimated wrong martensite size and the size fraction.

In the current settings, we use trapezoidal fuzzy sets for these fuzzy variables. For example, the membership value for fuzzy set "*a few*" is 0 when the estimated

martensite number is 0 and it is 1 when the estimated martensite number is between 0 and 10. The size fraction is calculated according to the following equation,

$$size\ fraction(j) = \frac{(N_1 - 10)PotMartSize(j)}{\sum_{i=N-N_1}^{N-10} PotMartSize(i)} \times 100\%$$

Where the PotMartSize is the size of the current martensite. Its unit is the pixel number. *N* is the possible martensite number from the BWLabel function. N_1 is the defuzzification value of **estimated martensite number**. Here height defuzzification approach is employed. In this equation, the first larger ten martensites are not counted into the calculation since it's much larger than the left martensites. 10 is the adjustable parameters which can be changed according the expert's experience. Using such an inference system, almost all noise points will be deleted. We will show a exemplar resulting image in the next chapter.

IF the **estimated martensite number** is *a few* and the **estimated wrong martensite size** is *too small*, THEN the martensite whose **size fraction** is less than too *small* will be deleted.

IF the **estimated martensite number** is *some* and the **estimated wrong martensite size** is *too small*, THEN the martensite whose **size fraction** is less than too *small* will be deleted.

IF the **estimated martensite number** is *many* and the **estimated wrong martensite size** is *too small*, THEN the martensite whose **size fraction** is less than too *small* will be deleted.

IF the **estimated martensite number** is *numerous* and the **estimated wrong martensite size** is *too small*, THEN the martensite whose **size fraction** is less than too *small* will be deleted.

IF the **estimated martensite number** is *a few* and the **estimated wrong martensite size** is *very small*, THEN the martensite whose **size fraction** is less than *very small* will be deleted.

IF the **estimated martensite number** is *some* and the **estimated wrong martensite size** is *very small*, THEN the martensite whose **size fraction** is less than *very small* will be deleted.

IF the **estimated martensite number** is *many* and the **estimated wrong martensite size** is *very small*, THEN the martensite whose **size fraction** is less than *very small* will be deleted.

IF the **estimated martensite number** is *numerous* and the **estimated wrong martensite size** is *very small*, THEN the martensite whose **size fraction** is less than *very small* will be deleted.

IF the **estimated martensite number** is *a few* and the **estimated wrong martensite size** is *small*, THEN the martensite whose **size fraction** is less than *small* will be deleted.

IF the **estimated martensite number** is *some* and the **estimated wrong martensite size** is *small*, THEN the martensite whose **size fraction** is less than *small* will be deleted.

IF the **estimated martensite number** is *many* and the **estimated wrong martensite size** is *small*, THEN the martensite whose **size fraction** is less than *small* will be deleted.

IF the **estimated martensite number** is *numerous* and the **estimated wrong martensite size** is *small*, THEN the martensite whose **size fraction** is less than *small* will be deleted.

IF the **estimated martensite number** is *a few* and the **estimated wrong martensite size** is *normal*, THEN the martensite whose **size fraction** is less than *normal* will be deleted.

IF the **estimated martensite number** is *some* and the **estimated wrong martensite size** is *normal*, THEN the martensite whose **size fraction** is less than *normal* will be deleted.

IF the **estimated martensite number** is *many* and the **estimated wrong martensite size** is *normal*, THEN the martensite whose **size fraction** is less than *normal* will be deleted.

IF the **estimated martensite number** is *numerous* and the **estimated wrong martensite size** is *normal*, THEN the martensite whose **size fraction** is less than *normal* will be deleted.

For the holes in the martensite areas caused by threshold process, we also can use a similar method to deal with it. In this platform, we provide another program to fill the wrong holes. In this program all holes will be identified using the same method

BWLabel(). The estimated incorrect hole size is the only fuzzy parameter in this program which is in terms of the percentage of minimum martensite size. The trapezoidal fuzzy sets are employed for it whose shapes are the same as estimated wrong martensite size and height locations are at 10%, 50%,100% and 300% respectively. The example for a resulting image will be also shown in the next chapter.

2.7 The Windows Journal File Operations

For some complex images, it's difficult to obtain the accurate results even using the fuzzy inference system. In such a case, the novel platform provides a program to process the contour image from windows journal. The GUI window is shown in Fig. 2-10. In contour image, the black color represents the boundary of martensite, while the white color represents both the martensite and ferrite. Therefore, we need to distinguish the ferrite first. The ferrite should be the largest connected region. The user was asked to specify any point within the ferrite. Then we use the function get(gcf,'CurrentPoint') to catch that point location. All connected area to this point will be rendered as a grey color. Applying BWLable() function, we can find all martensite areas and fill the martensite areas with black color and the ferrite area with white color, respectively. Since the color for the martensite area is required to be white in the result generation part, we use complement operation to convert the color for the whole image. With this program, we do not need to launch other programs to convert the contour image to final binary image like windows paint and do not need to fill martensite area color one by one.

Figure 2-10. The program for windows journal file.

2.8 Report Generation

To finalize the image processing, we develop the program GenerateMartReport to generate martensite information. The GUI window for this program is shown in Fig. 2-11.

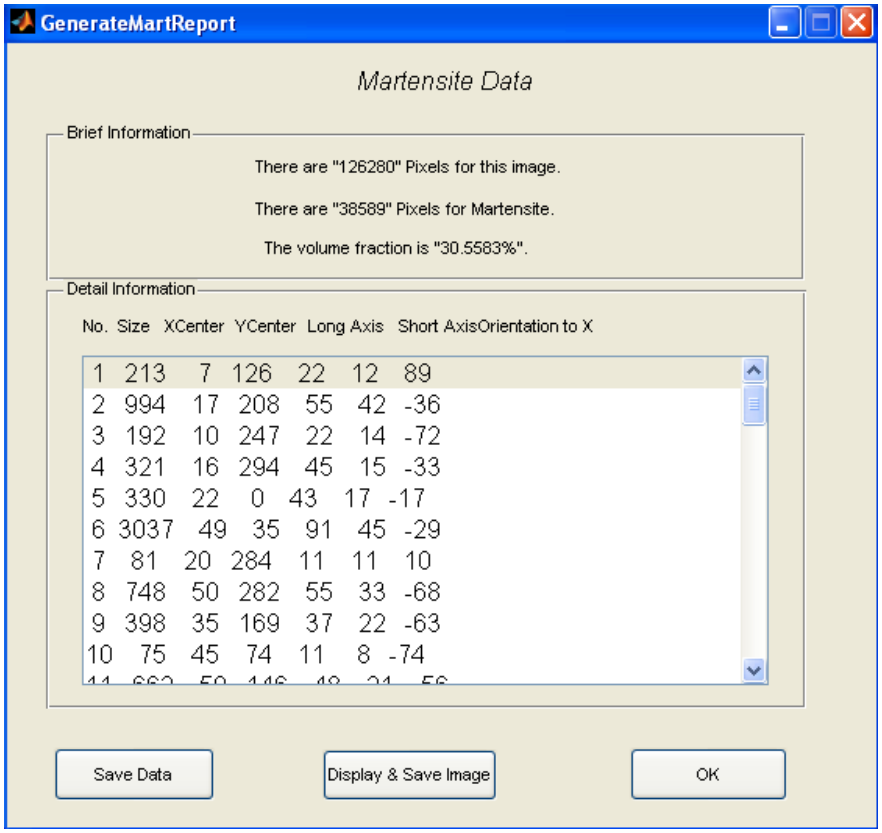


Figure 2-11. The program to generate martensite information report

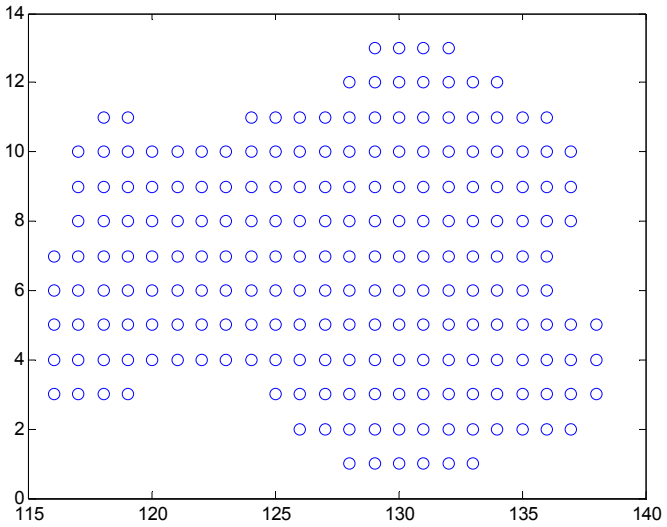
In this program, we have two tasks. The first one is to calculate the martensite information such as volume fraction, martensite position etc. We use Matlab function BWLabel to identify each martensite area. Then the size, center for each martensite area is easy to be obtained. The volume fraction is then calculated according to the following equations:

$$VolumeFraction = \frac{\sum_{i=1}^{m} MartSize(i)}{ImageSize}$$

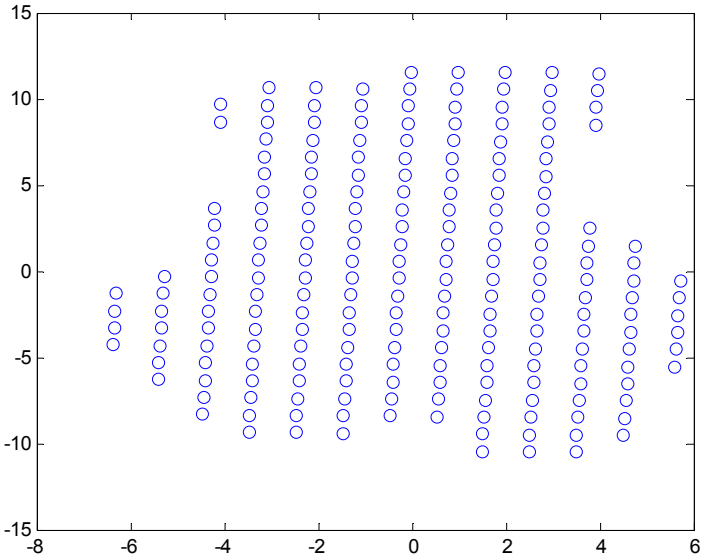
where the MartSize(i) is the size of each martensite area, ImageSize is the size of the whole Image, and m is the total number of martensite areas.

The center of each martensite is calculated by taking mean value on x coordinates and y coordinates, respectively.

The long axis, short axis and orientation are important for analyzing sample crack. We use principal component analysis (PCA) method to calculate these values. PCA was proposed by Karl Pearson in 1901[58]. It is used to find the direction which can best represent the data, i.e. the variance is largest. In our application, the long axis direction is just the largest variance direction. Therefore we centralize each martensite by subtracting by mean first. Then the covariance matrix for each martensite is calculated. Through Karhunen-Loeve transform (KLT), we can obtain the eigenvectors and eigenvalues. The eigenvector which corresponds to the largest eigenvalue is the direction of the largest variance i.e. long axis. The derivation of this process can be transformed to an optimal problem which can be found in [59]. With the eigenvectors, we can multiply eigenvector matrix by the centralized martensite data to rotate the martensite. The long axis and short axis range can be calculated by Matlab function range(). The orientation can be calculated by carrying out atan() function on the eigenvector.



(a)



(b)

Figure 2-12. The PCA process. (a) Original martensite point cloud. (b) the martensite point cloud after centralizing and rotating by PCA.

All these information can be saved to hard disk using Matlab function fopen(), fprintf() and fclose() by clicking Save Data button shown in Fig. 2-9.

The second task is to show all martensite areas with the number and orientation in a figure. Once the button display & save image is clicked, a new window will be pop up shown in Fig. 2-11. Here the Matlab function edge() is used to find the martensite contour if the source file is raw SEM image. The number for each martensite area is assigned around the corresponding martensite. Each martensite's orientation is also marked one the right side in degree on the figure.

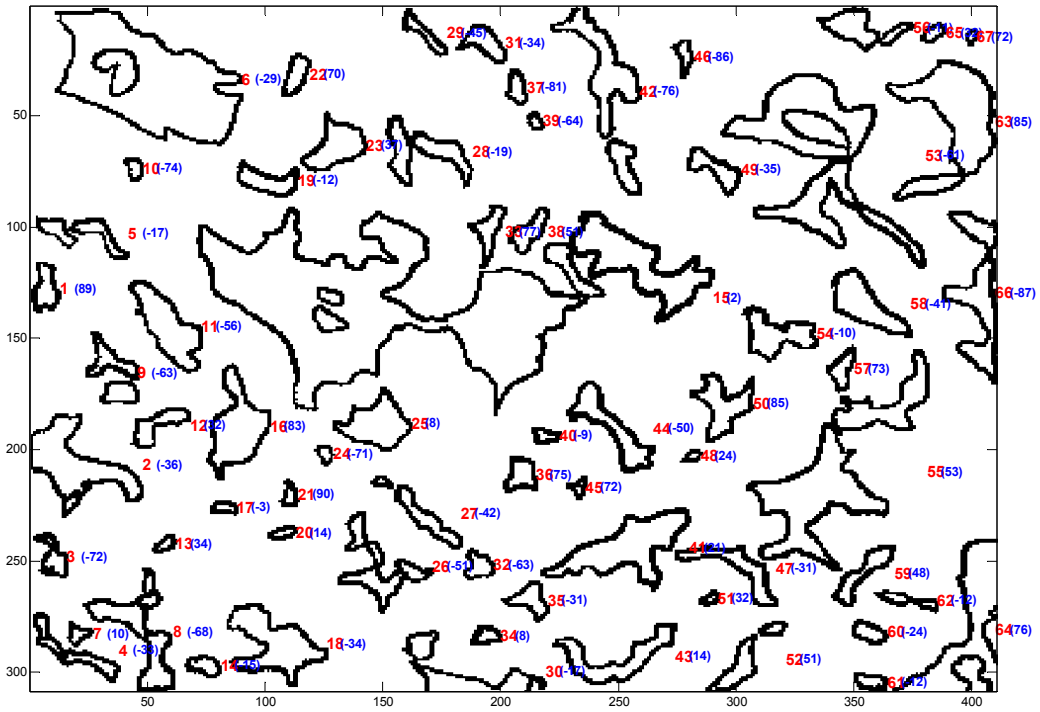


Figure 2-13. The martensite distribution image with number (red) and orientation (blue) generated by our platform.

CHAPTER 3 RESULTS AND COMPARISONS

In this chapter, the novel image processing platform is used to analyze DP steels SEM images. We will compare the results to show the advantage of our platform.

3.1 The Analysis for Raw SEM Image.

Example 1. B1YX2 for DP780 steels (5000X)

The original image is shown in Fig. 3-1. In Scion, if we choose the threshold value according to the bottom martensite area. Then after threshold, some top martensite areas are obviously enlarged (Fig. 3-2). If a filter is applied to remove noises, the image becomes blurring. In Fig. 3-3, the effect on image can be observed for different iterations of filter. One can easily conclude that the number of iteration is larger, the effect of noise remove is better, but the distortion of martensite area is more distinct. If our platform is used, all these drawbacks can be circumvented.

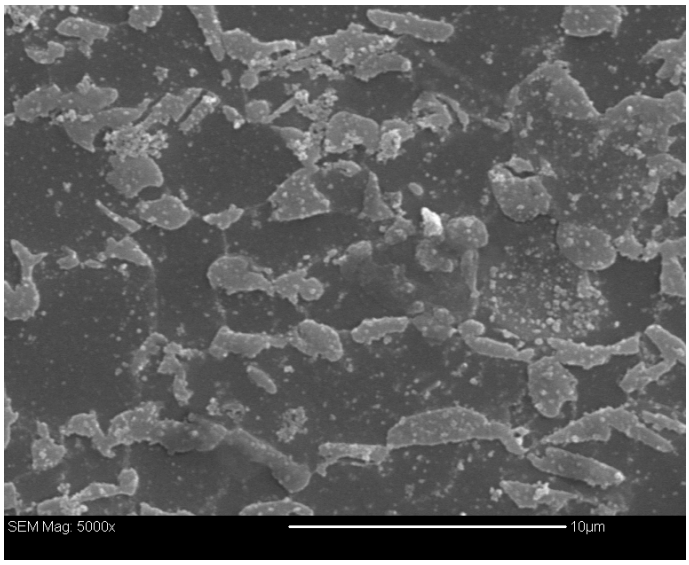
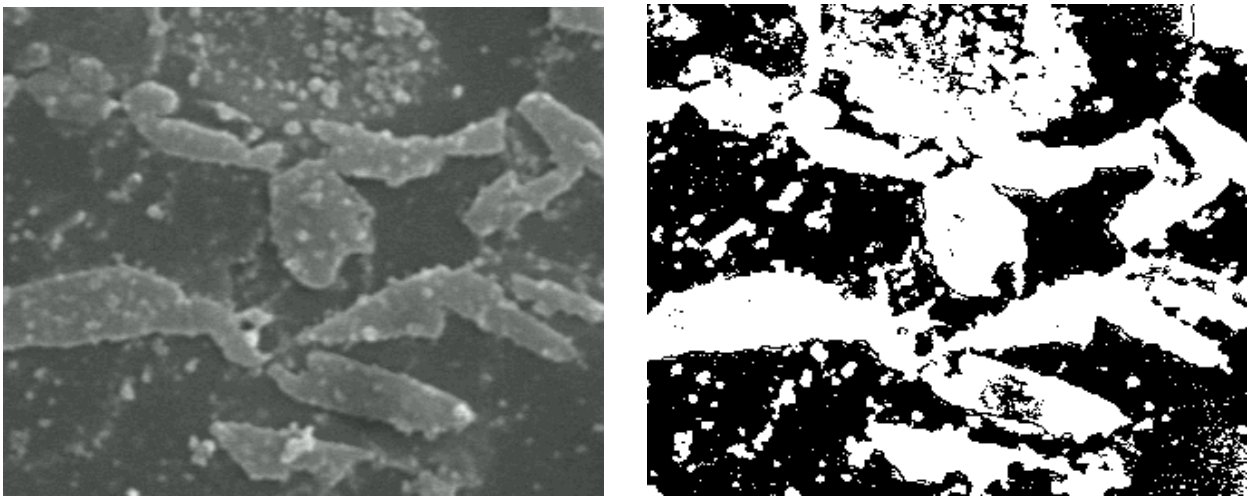
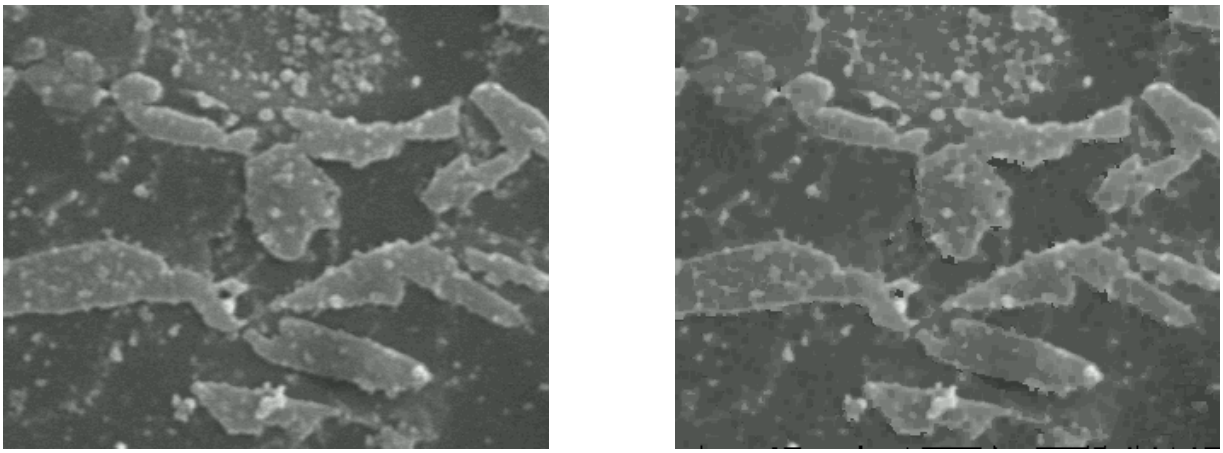


Figure 3-1 The Microstructure of B1YX2 for DP780 steels

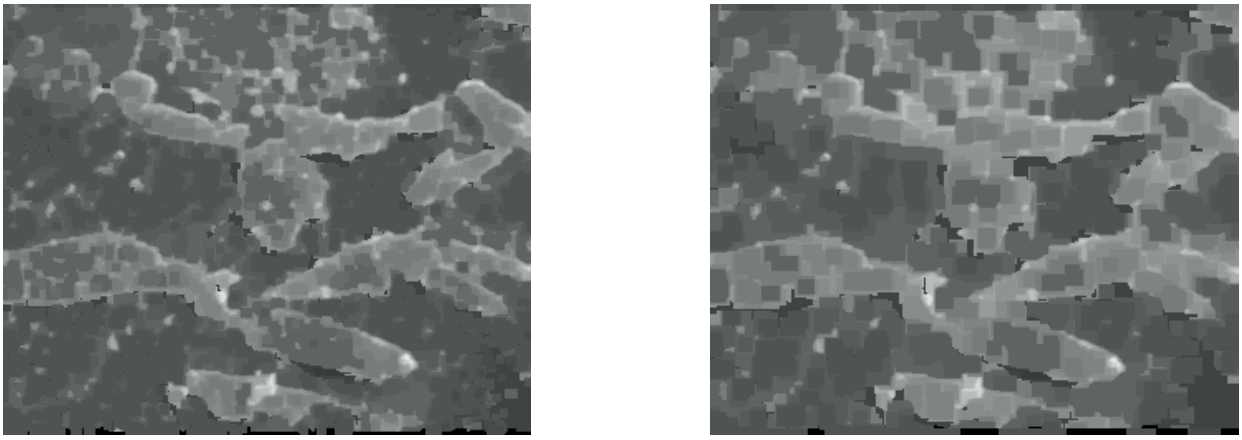


(a) (b)

Figure 3-2. The threshold result for part of Fig. 3-1. (a) Original image. (b) Manual threshold result.



(a) (b)

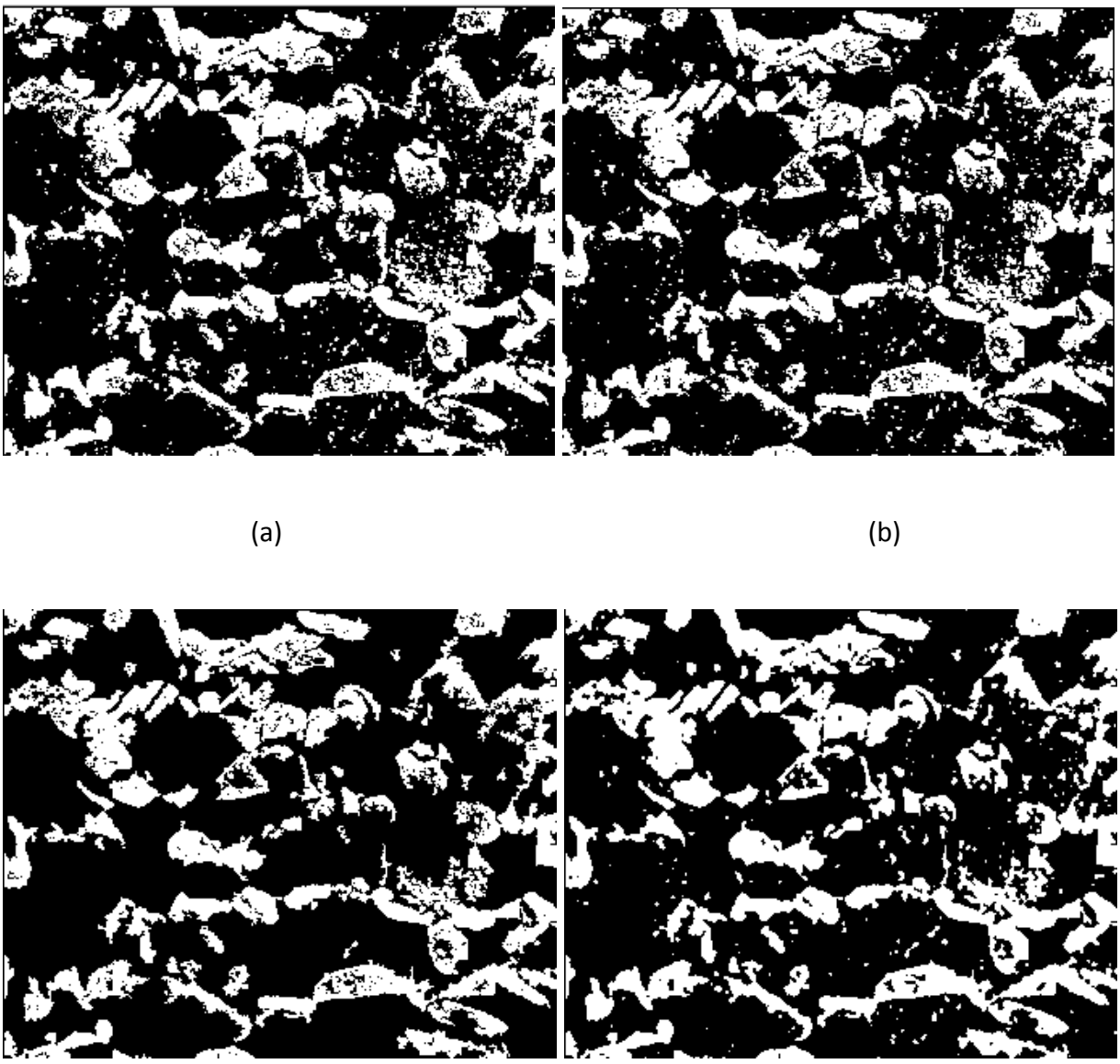


(c) (d)

Figure 3-3. The rank filter result for part of Fig. 3-1. (a) Original image. (b) One iteration of rank filter. (c) Two iterations of rank filter. (d) Five iterations of rank filter.

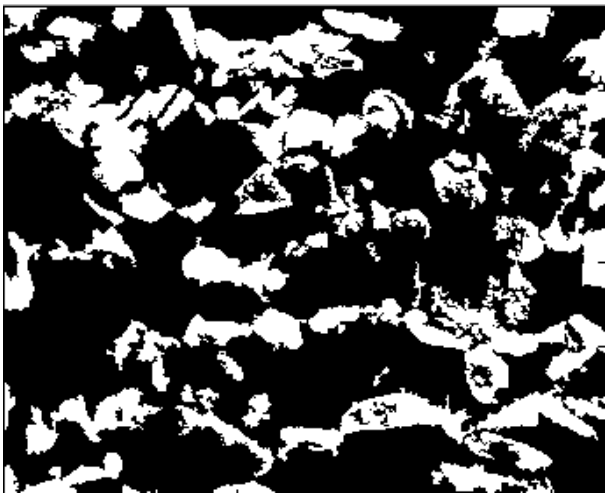
We can open the source file in our platform. The threshold process will be done first. Here Fig. 3-4 shows the global automatic threshold result and the local (5X5) automatic threshold result. From the comparison of (a) and (b), we can conclude the local threshold technique achieves more accurate result, e.g. the black points in martensite area are less, the outline of martensite area is more accurate etc. The image (b) still contains a lot of wrong martensite areas (noise), the number of real martensite is about 20-100. Therefore, we choose the fuzzy variable **estimated martensite number** to be *some* and the **estimated wrong martensite size** to be *very small* to do the fuzzy linguistic inference operation to remove the noises. The result is shown in Fig. 3-4 (c). If we use mathematical method like median filter (5X5), the result (shown in Fig. 3-4 (d)) is not satisfied: there are still some noises left and some martensite areas are distorted comparing Fig. 3-4(c). From Fig. 3-4 (c), we can use the fuzzy method to fill the hole. The fuzzy variable **estimated incorrect hole size** is chosen as *small* since we want to

keep some big holes which are more like ferrite than martensite. The final binary image is shown in Fig. 3-4 (e).



(a) (b)

(c) (d)



(e)

Figure 3-4. Output results from a sequential image processing steps: (a) The global automatic threshold result . (b) The local (5X5) automatic threshold result which is better than (a). (c) Using the fuzzy method to remove noises. (d) Using median filter (5X5) to remove noises which is worse than (c). (e) Using the fuzzy method to fill the hole.

Using Generate Martensite Report function, we can get the volume fraction which is 29.28% and the martensite area distribution plot is shown in Fig. 3-5. The individual martensite information is given in Table 3-1. The whole imaging process takes less than 5 minutes and the result is more accurate than other methods.

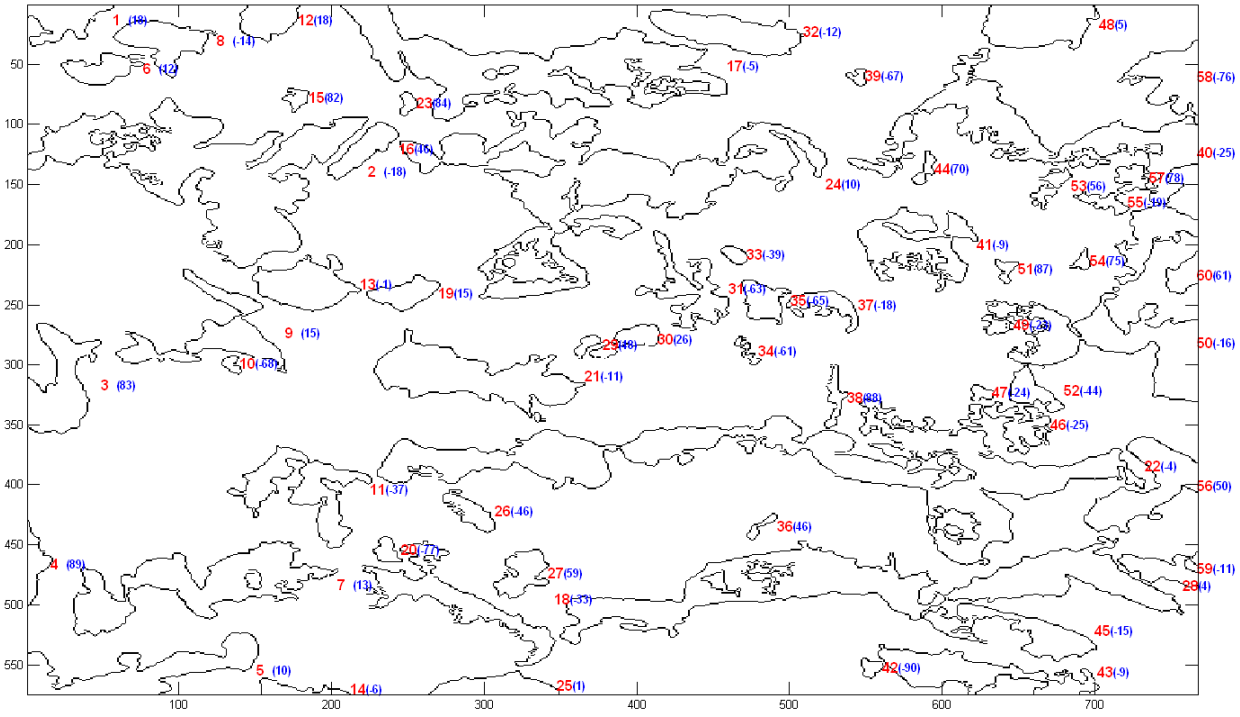


Figure 3-5. The output figure of martensite distribution, number and orientations for B1YX2 DP780 from our image processing platform.

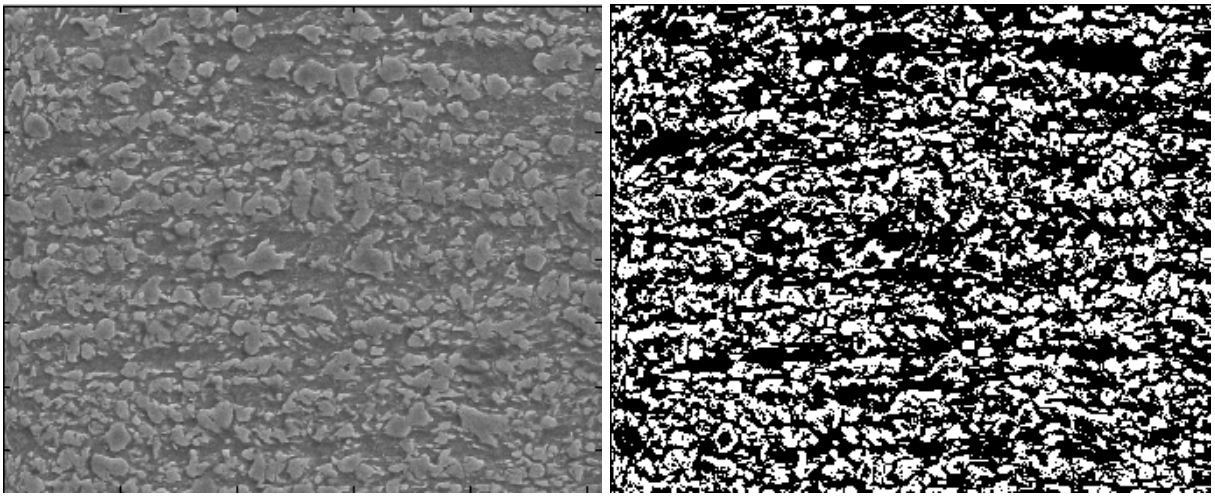
Table 3-1 Individual martensite information for B1YX2 DP780 obtained from our platform

No.	Size	X Center	Y Center	Long Axis (pixels)	Short Axis (pixels)	Orientation to X Axis (Degree)
1	1094	23	15	61	38	18
2	11159	129	163	203	144	-18
3	2381	22	315	91	44	83
4	515	5	469	71	16	89
5	3132	76	565	155	49	10
6	801	50	54	54	25	12
7	5056	114	499	180	60	13
8	1431	90	36	65	43	-14
9	2383	127	272	104	100	15
10	123	135	299	16	12	-68
11	2504	186	400	87	72	-37
12	888	158	16	42	33	18
13	1901	184	231	71	46	-1
14	370	170	571	60	13	-6
15	187	177	79	20	16	82

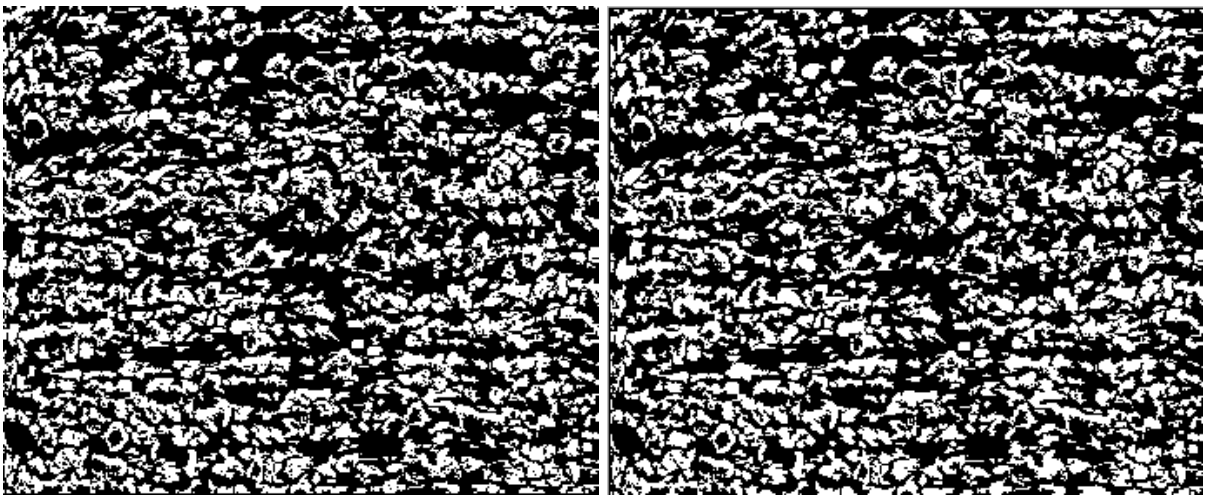
16	846	222	119	58	21	46
17	9159	332	58	258	92	-5
18	2617	274	488	142	67	-33
19	733	249	243	48	27	15
20	237	236	454	23	17	-77
21	4261	288	309	138	60	-11
22	14896	551	356	503	133	-4
23	145	249	83	19	12	84
24	12259	365	148	272	153	10
25	1057	309	567	84	19	1
26	525	291	423	43	18	-46
27	839	325	474	46	34	59
28	10333	539	476	408	77	4
29	158	366	283	21	13	48
30	561	397	276	49	21	26
31	1531	435	253	87	56	-63
32	2166	461	23	102	31	-12
33	174	464	209	18	11	-39
34	126	472	287	25	9	-61
35	687	482	247	43	34	-65
36	125	479	438	26	8	46
37	669	524	248	50	30	-18
38	303	530	323	53	14	88
39	111	545	61	14	14	-67
40	7596	695	120	234	146	-25
41	2501	581	196	78	67	-9
42	164	553	554	16	14	-90
43	2483	611	551	147	34	-9
44	180	589	133	29	12	70
45	1883	659	525	115	32	-15
46	1004	649	353	61	41	-25
47	139	625	323	16	13	-24
48	2260	662	18	78	41	5
49	190	635	268	22	16	-23
50	4753	717	279	156	82	-16
51	150	642	224	20	15	87
52	618	657	318	44	27	-44
53	335	671	148	37	17	56
54	116	692	213	18	13	75
55	298	707	166	38	17	-19
56	2481	735	383	86	60	50
57	367	724	144	26	23	78
58	1666	752	61	60	59	-76
59	845	744	470	55	30	-11
60	456	758	227	38	23	61

Example 2. Side for DP780 steels (1500X)

Example 2 is a complex image shown in Fig. 3-6 (a). The magnification is only 1500X. Therefore, there are a lot of martensite particles. Generally, it takes about two hours to draw the contour manually and finish the image processing. In our platform, it would be very easy. Fig. 3-6 (b-d) show the result for local threshold (7X7), fuzzy noise removing (two fuzzy variables are chosen as *numerous* and *small*) and fuzzy hole filling (the fuzzy variable is set as *normal*), respectively.



(a) (b)



(c) (d)

Figure 3-6. The sequential image processing results for side direction of DP780 (1500X). (a) The raw SEM image. (b) The local (7X7) automatic threshold result. (c) Using the fuzzy method to remove noises. (d) Using the fuzzy method to fill the hole.

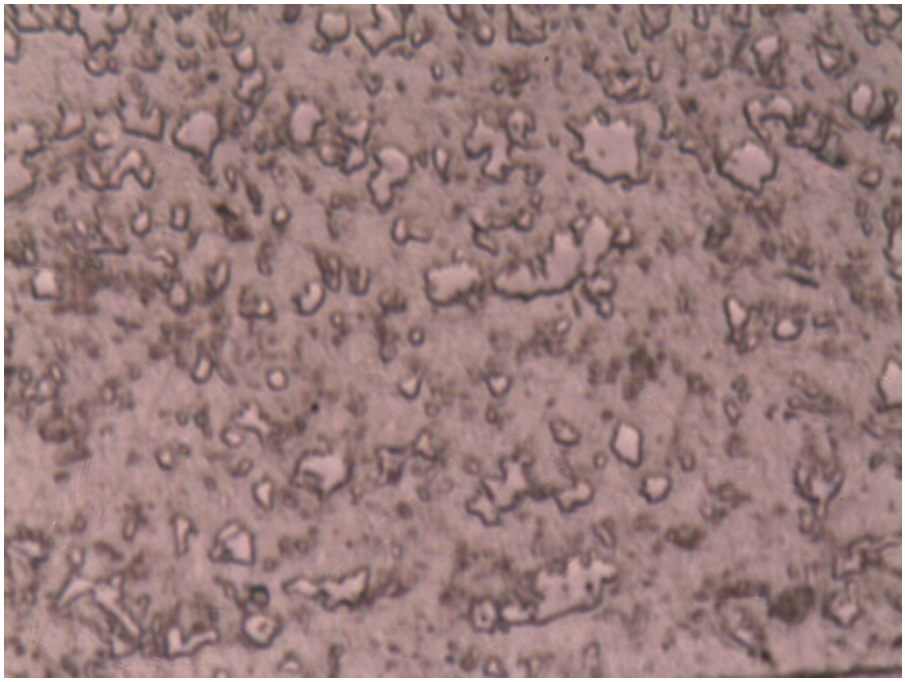
After calculating the martensite information, we can obtain that the martensite number is 621 and the volume fraction is 36.46%. Using our novel imaging processing platform, the whole process only spends about 5 minutes.

3.2 Windows Journal File Results

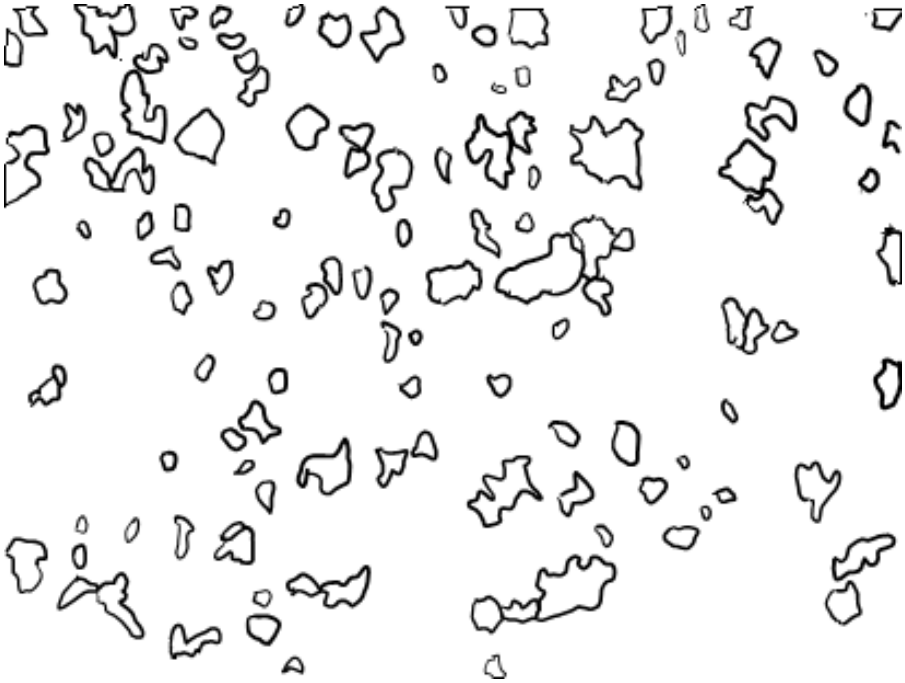
For some images that can not be currently recognized by our platform, we can use windows journal to draw its contour figure. Then in our platform, we can employ journal file operations to make the binary image and extract the martensite information.

Example 3. DP600 1000 steels (5000X)

The original raw SEM image and contour image for DP600A are shown in Fig. 3-7. The binary image is shown in Fig. 3-8 and the martensite distribution result is shown in Fig. 3-9. There are totally 107 martensite areas, and the volume fraction is 19.1%.



(a)



(b)

Figure 3-7. (a) The original SEM image for DP 600a. (b) The corresponding contour image drawing by hand in windows journal.

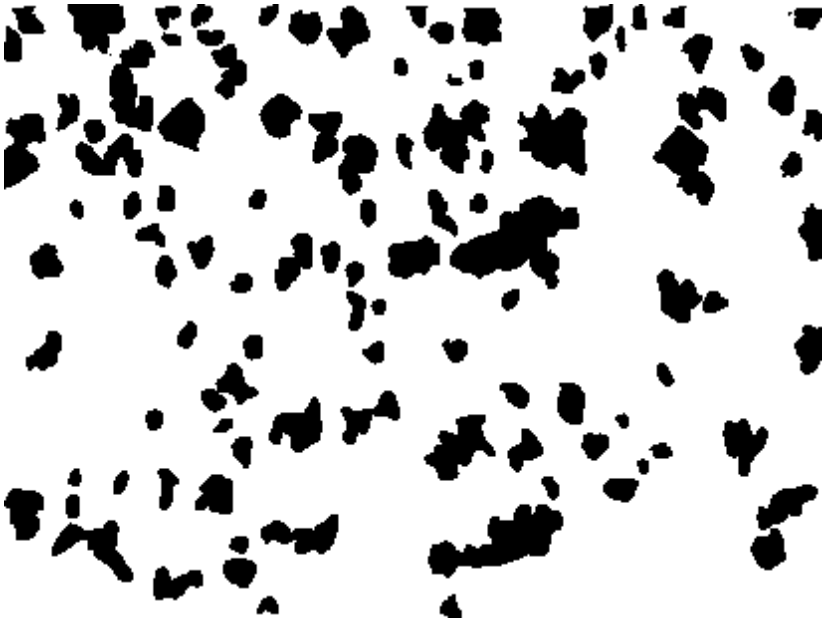


Figure 3-8. The binary image result for DP 600A.

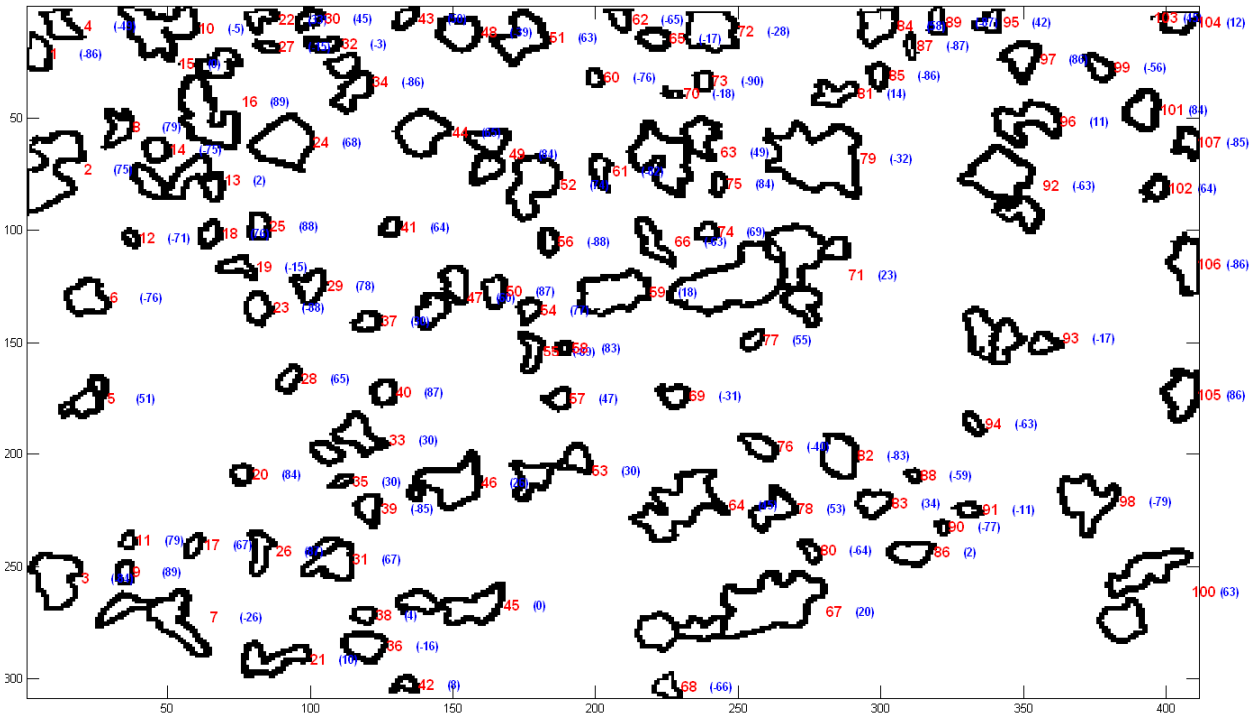
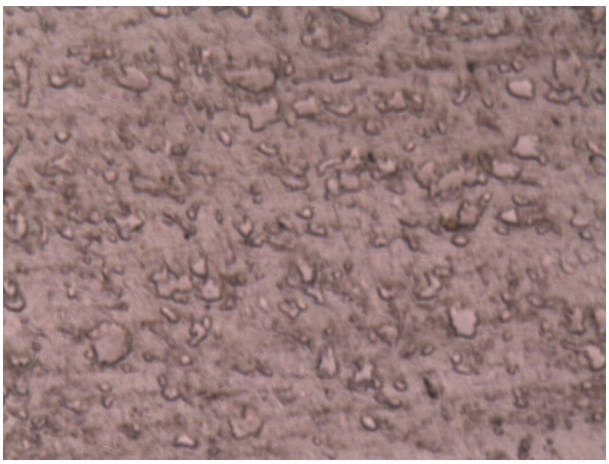


Figure 3-9. The martensite distribution result for DP 600A from our platform..

Similarly, the DP600B and DP600C are processed in the same way. The original raw SEM image and contour image for DP600B are shown in Fig. 3-10 (a) and (b), respectively. The binary image is shown in Fig. 3-10 (c). The volume fraction of martensite is 15.43%. The original raw SEM image and contour image for DP600C are shown in Fig. 3-11 (a) and (b), respectively. The binary image is shown in Fig. 3-11 (c). Volume fraction of martensite: 6.56%

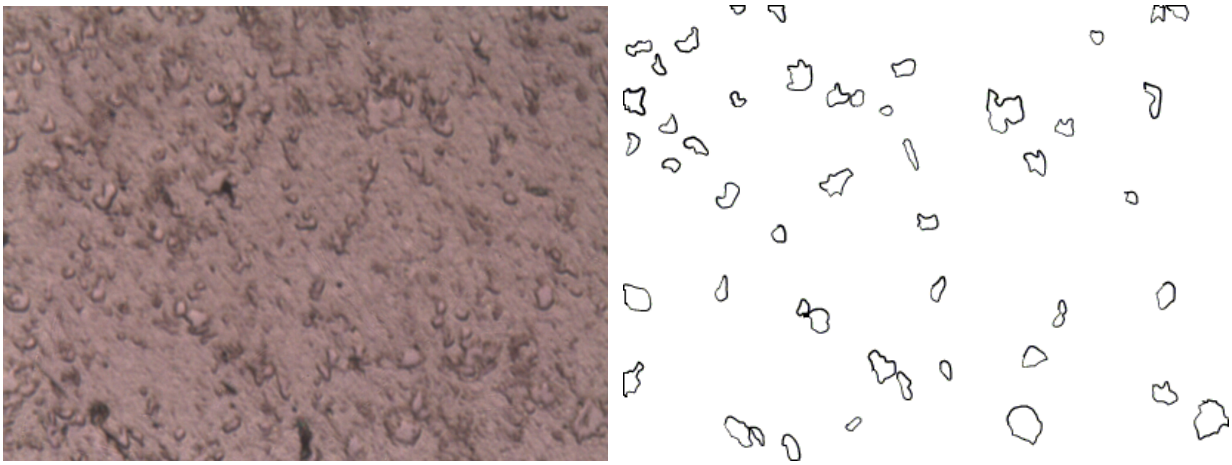


(a)

(b)

(c)

Figure 3-10. The image processing results for side direction of DP600B (5000X). (a) The raw SEM image. (b) Contour image drawing by hand in windows journal (c) The binary image result automatically generated from our platform using Fill Ferrite function.



(a) (b)

(c)

Figure 3-11. The image processing results for side direction of DP600C (5000X). (a) The raw SEM image. (b) Contour image drawing by hand in windows journal (c) The binary image result automatically generated from our platform using Fill Ferrite function.

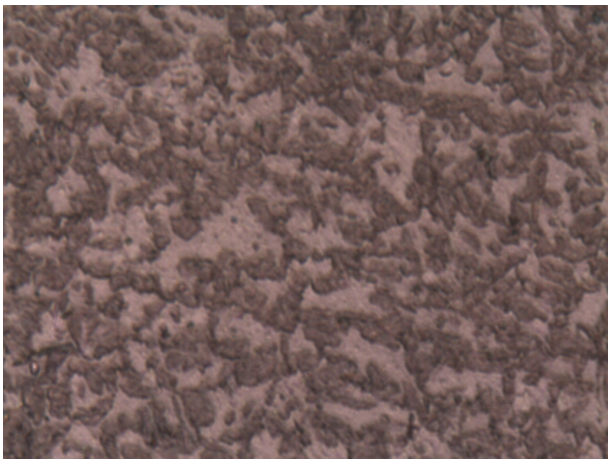
Example 4. DP980 steels (5000X)

In this example, DP980 steels in three different directions are processed.

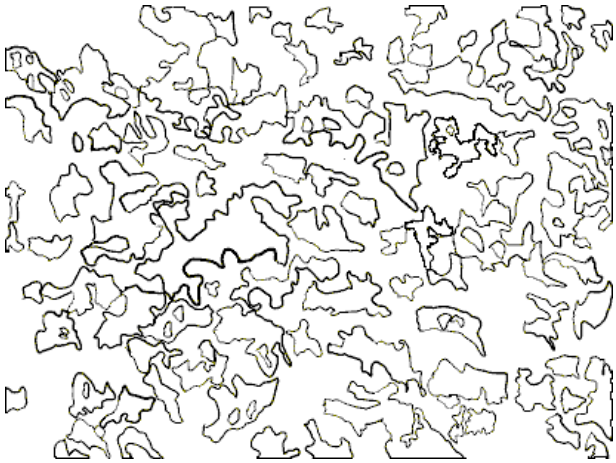
The original raw SEM image and contour image for DP980A are shown in Fig. 3-12 (a) and (b), respectively. The binary image is shown in Fig. 3-12 (c). The volume fraction of martensite is 51.87%.

The original raw SEM image and contour image for DP980B are shown in Fig. 3-13 (a) and (b), respectively. The binary image is shown in Fig. 3-13 (c). Volume fraction of martensite is 50.30%.

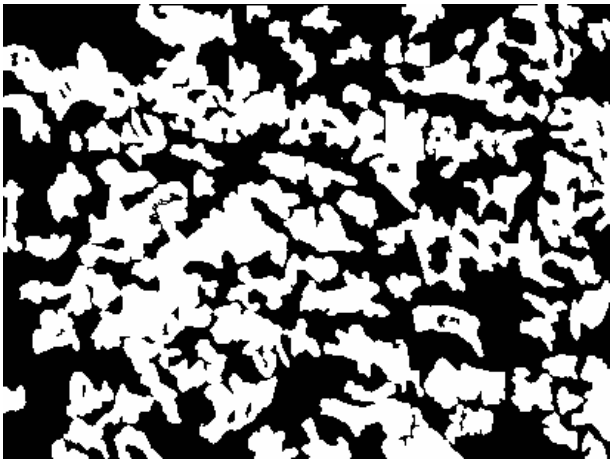
The original raw SEM image and contour image for DP980C are shown in Fig. 3-14 (a) and (b), respectively. The binary image is shown in Fig. 3-14 (c). Volume fraction of martensite: 39.77%



(a)

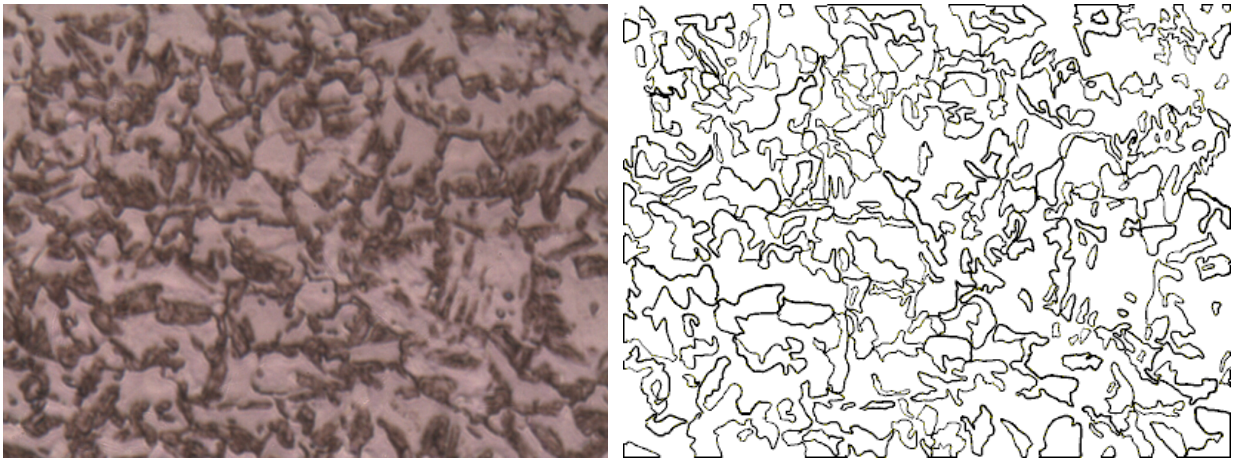


(b)



(c)

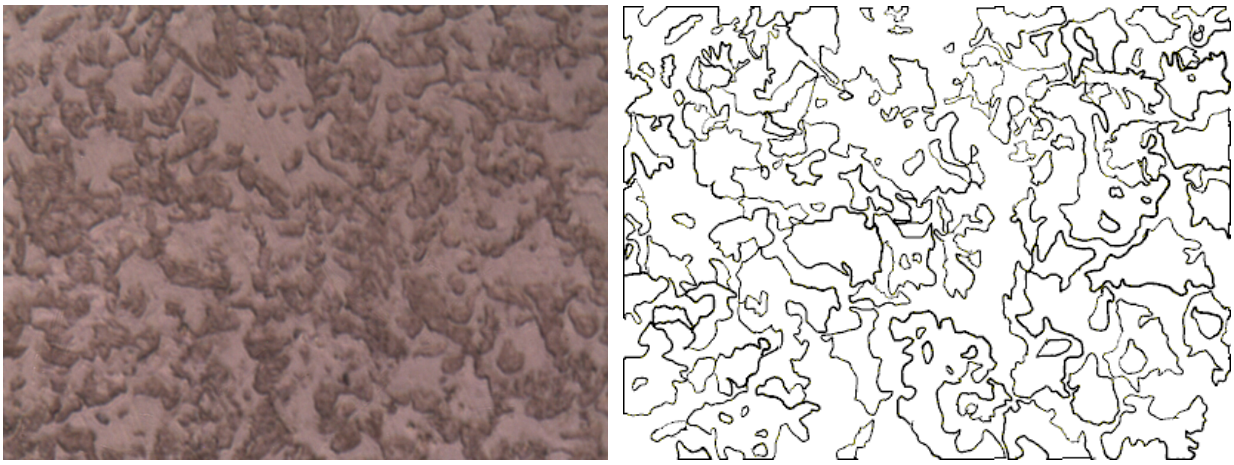
Figure 3-12. The image processing results for side direction of DP980A (5000X). (a) The raw SEM image. (b) Contour image drawing by hand in windows journal (c) The binary image result automatically generated from our platform using Fill Ferrite function.



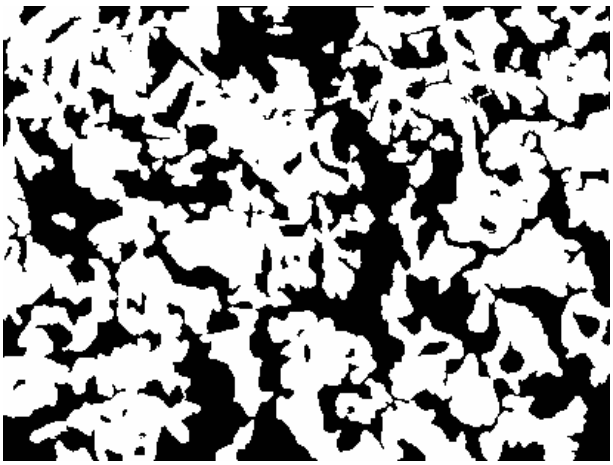
(a) (b)

(c)

Figure 3-13. The image processing results for side direction of DP980B (5000X). (a) The raw SEM image. (b) Contour image drawing by hand in windows journal (c) The binary image result automatically generated from our platform using Fill Ferrite function.



(a) (b)



(c)

Figure 3-14. The image processing results for side direction of DP980C (5000X). (a) The raw SEM image. (b)Contour image drawing by hand in windows journal (c) The binary image result automatically generated from our platform using Fill Ferrite function.

CHAPTER 4 CONCLUSION AND FUTURE WORK

In this thesis, we developed a novel image processing platform for microstructure analysis for DP steels in the Matlab environment. Based on the existing powerful functions in Matlab Image Processing Toolbox (e.g. threshold, filter, edge find, image read and display etc.), more functions and algorithms are added in this platform that makes this platform exclusive from other common image processing software: We improve the current threshold method so that the new platform can automatically divide the whole image into several regions and apply the automatic threshold method on each region. We also design and apply the fuzzy linguistic inference system to remove martensite noises and fill holes in martensite areas. The linguistic approach utilizes the experts' experience and exhibits better performance than traditional filters. Some geometrical information e.g. long axis, short axis and orientation etc. can be extracted and marked on the final output figure. Four examples show the validation and efficiency of our new platform on microstructure analysis comparing with the previous processing method (i.e. using Scion Image) It worth noting that the platform is validated by DP steel images but not limited to them. Similar results can be expected with our platform for other types of AHSS.

Since the new platform is developed in the Matlab environment, it leaves us more facility and flexibility to extend the platform for further desires. There are several directions can be done in the future:

1) As we know, there are a lot of different raw images and some of them can not be processed directly by our platform (e.g. Fig. 3-7 (a)). It's significant to develop some new algorithms to process these images directly.

2) In the current version, the division of the image for auto thresholding is specified by users. We can utilize the fuzzy inference method to automatically select the image sub-area and find the appropriate threshold.

3) The martensite is identified by pixel intensity currently. For some color raw images, texture recognition is a more accurate way.

4) Statistical analysis is a promising direction in microstructure analysis field. With the help of Matlab Statistic Toolbox, it is not difficult to apply statistical algorithms, functions and knowledge in platform and obtain more meaningful features.

APPENDIX A: IMAGE PROCESSING PROCESSDURES

To process a dual-phase image, you can do the following steps:

Step1. Start Matlab.

Step2. Change the Current Directory to the locations where you store the Microstructure Image Processing V1.0

Step3. In Matlab Command Windows type ImageProV1

Step4. Click File->Open to open a saved dual-phase image. For example there is sample image B1_YX2_03X5000.PNG in the installation folder.

Step5. Click Threshold/Filter to operate threshold process. You can choose any one from possible four threshold method.

Step6. Click AI Optimization->Rule1 to delete wrong martensites. Click start button to start the process and click the close button to return.

Step7. Click AI Optimization->Rule 2 to fill the holes within the martensites. Click start button to start the process and click the close button to return.

Step8. Click Measurement->Generate Martensite Report to generate and save the martensite information.

Step9. Click View->Zoom to check and compare the detail of original image and modified image. If you need to do some changes please restart from step 4.

Step10. Click File->Save to save the modified image.

To process a contour image, you can do the following steps:

Step1. Draw the contour image using windows journal. Change it to an RGB image format. For example the sample image DP780-a-1000-5(J).png in the installation folder.

Step2. Start Matlab.

Step3. Change the Current Directory to the locations where you store the Microstructure Image Processing V1.0

Step4. In Matlab Command Windows type ImageProV1

Step5. Click File->Open to open the contour image.

Step6. Click Measurement->Fill Ferrite to change it to a dual-phase image. When the window FillFerrite is open, left click on the ferrite of the image. The color of ferrite will be changed from white to gray. Then click fill martensite button. Click convert color button to make sure the color of martensite be white. Click accept button to return to the main program.

Step 7. Click Measurement->Generate Martensite Report to generate and save the martensite information.

REFERENCES

[1] A. A. G. Committee, *Advanced High-Strength Steel Applications Guidelines: A Special Edition of In-Depth AHSS Case Studies*. Southfield, Michigan: Auto/Steel Partnership, 2008.

[2] *Advanced High Strength Steel (AHSS) Application Guidelines (Version 3)*: International iron & steel institute committee on automotive applications, 2006.

[3] E. Suzuki, "High-resolution scanning electron microscopy of immunogold-labelled cells by the use of thin plasma coating of osmium," *Journal of Microscopy,* vol. 208, pp. 153-157, 2002.

[4] M. Coster and J.-L. Chermant, "Image analysis and mathematical morphology for civil engineering materials," *Cement & Concrete Composites,* vol. 23, pp. 133-151, 2001.

[5] W. Pratt, *Digital image processing*. New York: Wiley, 1991.

[6] J. Serra, *Image analysis and mathematical morphology*. London: Academic Press, 1982.

[7] J.-L. Chermant and M. Coster, "Role of mathematical morphology in filtering, segmentation and analysis," *Acta Stereol,* vol. 13, pp. 125-136, 1994.

[8] L. Xu*, et al.*, "Segmentation of skin cancer images," *Image and Vision Computing,* vol. 17, pp. 65-74, 1999.

[9] N. Ostu, "A threshold selection method from gray-level histogram," *IEEE Trans. Syst. Man Cybernet,* vol. 9, pp. 62-66, 1979.

[10] R. Kohler, "A segmentation system based on thresholding," *Comp Vision Graph Image Process,* vol. 15, pp. 319-338, 1981.

[11] C. A. Glasbey, "An analysis of histogram-based thresholding algorithms " in *Graphical Models and Image Processing*, 1993, pp. 532-537.

[12] J. Kapur*, et al.*, "A new method for gray-level picture thresholding using the entropy of the histogram," *Comp Vision Graph Image Process,* vol. 29, pp. 273-289, 1985.

[13] W.-H. Tsai, "Moment-preserving thresholding: a new approach," *Comp Vision Graph Image Process,* vol. 29, pp. 377-393, 1985.

[14] J. I. Pastore*, et al.*, "Segmentation of brain magnetic resonance images through morphological operators and geodesic distance," *Digital Signal Processing,* vol. 15, pp. 153-160, 2005.

[15] A. X. Falcao*, et al.*, "Multiscale skeletons by image foresting transform and its application to neuromorphometry," *Pattern Recognition* vol. 35, pp. 1571-1582, 2002.

[16] M. S. Horritt, "A statistical active contour model for SAR image segmentation," *Image and Vision Computing,* vol. 17, pp. 213-224, 1999.

[17] T. Heimann and H.-P. Meinzer, "Statistical shape models for 3D medical image segmentation: A review," *Medical Image Analysis,* vol. 13, pp. 543-563, 2009.

[18] R. Larsen and K. B. Hilger, "Statistical shape analysis using non-euclidean metrics," *Medical Image Analysis,* vol. 7, pp. 417-423, 2003.

[19] J. Lotjonen*, et al.*, "Statistical shape model of ventricles and epicardium from short and long axis MR images," *Medical Image Analysis,* vol. 8, pp. 371-386, 2004.

[20] D. Shen*, et al.*, "Segmentation of prostate boundaries from ultrasound images using statistical shape model," *IEEE Trans. Med. Imaging,* vol. 22, pp. 539-551, 2003.

[21] B. Van Ginneken, *et al.*, "Active shape model segmentation with optimal features," *IEEE Trans. Med. Imaging,* vol. 21, pp. 924-933, 2002.

[22] S. Venkatesh and P. L. Rosin, "Dynamic threshold determination by local and global edge evaluation," *CVGIP: Graphical Models and Image Processing,* vol. 57, pp. 146-160, 1995.

[23] P. L. Rosin and G. A. W. West, "Non-parametric segmentation of curves into various representations," *IEEE Trans. PAMI,* vol. 17, pp. 1140-1153, 1995.

[24] J. Bruce. *Color Machine Vision*. Available: http://www-2.cs.cmu.edu/~jbruce/cmvision

[25] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision,* vol. 59, pp. 109-131, 2004.

[26] G. C. Ltd. *Facor*. Available: http://www.facor.info/aboutus.htm

[27] D. Tschumperlé. *CImg Library*. Available: http://cimg.sourceforge.net/

[28] (2005, *Edge detection and iamge segmentation system*. Available: http://coewww.rutgers.edu/riul/research/code/edison/index.html

[29] Euresys. *Open eVision 1.1*. Available: http://www.euresys.com/Products/MachineVisionSoftware/MachineVision.asp

[30] V. S. Group. *NeatVision*. Available: http://www.neatvision.com/

[31] S. Corporation. *Scion Image*. Available: http://www.scioncorp.com/

[32] *THB Componentware*. Available: http://www.thbcomponents.com/

[33] S. B. Dove. (2002, *UTHSCSA ImageTool*. Available: http://ddsdx.uthscsa.edu/dig/itdesc.html

[34] *Visiopharm* Available: http://www.visiopharm.com/

[35] D. Huber. (2005, *The Computer Vision Homepage*. Available: http://www.cs.cmu.edu/~cil/v-source.html

[36] Ditigteo. *Scilab*. Available: http://www.scilab.org/

[37] CGAL. *Computational geometry algorithms library*. Available: http://www.cs.uu.nl/CGAL/

[38] CMU. *StatLib*. Available: http://lib.stat.cmu.edu/

[39] A. F. Clark. (2004, *Computer Vision Software*. Available: http://peipa.essex.ac.uk/info/software.html

[40] C. T. W. Lahaye, "Daily routine quantitative assessment of microstructure of steel by image analysis," *Materials Characterization,* vol. 36, pp. 191-202, 1996.

[41] M. E. Hoque*, et al.*, "Automated Image Analysis of Microstructure Changes in Metal Alloys," presented at the IS&T/SPIE Symposium on Electronic Imaging 2005, Machine Vision Applications in Industrial Inspection XVIII, San Jose, CA, 2005.

[42] X. Yang*, et al.*, "Evaluation of biofilm image thresholding methods," *Water Res,* vol. 35, pp. 1149-1158, 2001.

[43] Z. Shan and A. M. Gokhale, "Digital image analysis and microstructure modeling tools for microstructure sensitive design of materials," *International Journal of Plasticity,* vol. 20, pp. 1347-1370, 2003.

[44] J. Komenda, "Automatic recognition of complex microstructures using the image classifier," *Materials Characterization,* vol. 46, pp. 87-92, 2001.

[45] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic theory and applications*. Upper Saddle River, NJ: Prentice Hall PTR, 1995.

[46] Zadeh, "Fuzzy sets," *Information and Control,* vol. 8, pp. 338-353, 1965.

[47] Y. Nie and K. E. Barner, "The fuzzy transformation and its applications in image processing," *IEEE Trans. on Image Processing,* vol. 15, pp. 910-927, 2006.

[48] C.-S. Lee*, et al.*, "Genetic-based fuzzy image filter and its application to image processing," *IEEE Trans. on System, Man and Cybernetics(B),* vol. 35, pp. 694-711, 2005.

[49] D. Sinha*, et al.*, "Design and analysis of fuzzy morphological algorithms for image processing," *IEEE Trans. on Fuzzy Systems,* vol. 5, pp. 570-584, 1997.

[50] J. K. Udupa and P. K. Saha, "Fuzzy connectedness and image segmentation," *Proceedings of the IEEE,* vol. 91, pp. 1649-1669, 2003.

[51] M.-E. Algorri and F. Flores-Mangas, "Classification of anatomical structures in mr brain images using fuzzy parameters," *IEEE Trans. on Biomedical Engineering,* vol. 51, pp. 1599-1608, 2004.

[52] L. Hildebrand and M. Fathi, "Knowledge-based fuzzy color processing," *IEEE Trans. on System, Man and Cybernetics(C),* vol. 34, pp. 499-505, 2004.

[53] N. V. Lopes*, et al.*, "Automatic histogram threshold using fuzzy measures," *IEEE Trans. on Image Processing,* vol. 19, pp. 199-204, 2010.

[54] L. Amini*, et al.*, "Automatic segmentation of thalamus from brain MRI integrating fuzzy clustering and dynamic contours," *IEEE Trans. on Biomedical Engineering,* vol. 51, pp. 800-811, 2004.

[55] D. T. Vo*, et al.*, "Adaptive fuzzy filtering for artifact reduction in compressed images and videos," *IEEE Trans. on Image Processing,* vol. 18, pp. 1166-1178, 2009.

[56] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[57] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision* Boston, MA: Addison-Wesley, 1992.

[58] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine,* vol. 2, pp. 559-572, 1901.

[59] R. O. Duda*, et al.*, *Pattern Classification (2 edition)*. New York: John Wiley & Sons, Inc., 2000.

ABSTRACT

A NOVEL IMAGE PROCESSING PLATFORM DEVELOPMENT FOR PRE-PROCESSING IMAGES WITH TWO-PHASE MICROSTRUCTURES CONTAINING ISOLATED PARTICLES

by

YUXIU ZHANG

December 2010

Advisor: Dr. Xin Wu

Major: Mechanical Engineering

Degree: Master of Science

A novel image processing platform is developed in this thesis for pre-processing the images of two-phase microstructures containing isolated particles based on Matlab®. The two phases in this work represent martensite particles and ferrite matrix in Dual Phase steels that are developed for lightweight automotive body structures. An automated image processing tool is useful for obtaining statistical microstructure characteristics, important for better understanding material's deformation and fracture behavior. However, a generalized commercial software can not automatically realize the microstructural features of special interest, and a user-specified pre-processing tool is needed.

This platform utilizes the Matlab® GUI technique, Matlab image processing functions, fuzzy technology to distinguish martensites from ferrites, optimize the intermediate martensite results, analyze and generate the martensite distribution information and volume fraction. The multiple region thresholding technique is proposed to enhance the image segmentation. A fuzzy linguistic system is presented to utilize

experts' experience to remove noise and fill the unexpected holes on the image. The examples show that with this platform the pre-processing on images of two phase microstructures containing isolated particles is more convenient, fast and accurate.

AUTOBIOGRAPHICAL STATEMENT

Yuxiu Zhang

Education

07/2005, B.S. in Mechanical Engineering, Department of Mechanical Engineering, Tsinghua University, Beijing, China

Professional Experience

04/2008—08/2008 Student Assistant ME Dept.of Wayne State University, Detroit, MI, US

04/2007—08/2007 Internship: Test Engineer, Coldheading Company, Detroit, MI, US

01/2007—05/2007 Part Time Metallurgic Technician, ME Dept. of Wayne State University, Detroit, MI US

09/2006—12/2006 Student Administrative Assistant, Graduate Admission Office of Wayne State University Detroit MI US

06/2006—09/2006 Student Assistant, ECE Department of Wayne State University Detroit MI US