

TRANSACTIONS ON IMAGE PROCESSING

By

N.SARAVANAKUMAR

Register Number: 71205621038

Of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

Of

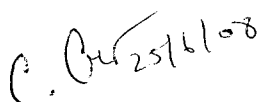
MASTER OF COMPUTER APPLICATION

**ANNA UNIVERSITY
CHENNAI 600 025**

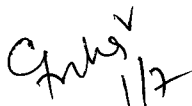
June 2008

BONAFIDE CERTIFICATE

Certified that this project report titled “**TRANSACTIONS ON IMAGE PROCESSING**” is the bonafide work of **Mr. N.SARAVANAKUMAR** (Register Number: 71205621038) who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate

**SUPERVISOR****HEAD OF THE DEPARTMENT**

Submitted to Project and Viva Examination held on 1.7.2008

**INTERNAL EXAMINER****EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost I thank God for his good will and blessings showered on me throughout the project. The success of this project needs cooperation and encouragement from different quarters. Words are inadequate to express my profound and deep sense of gratitude to those who helped me in bringing out this project successfully.

I wish to express my deep unfathomable feeling of gratitude and indebtedness to **Dr. Joseph V. Thanikal, B.E., M.E., Ph.D., PDF., CEPIT.**, Principal, Kumaraguru College of Technology, Coimbatore for the successful completion of the project work.

I am very gladly taking this opportunity to express a special word of thanks to **Dr. M. Gururajan M.Sc., Ph.D**, Head of the Department, Department of Computer Applications, Kumaraguru College of Technology, Coimbatore for encouraging me to do this work.

I am very much indebted to **Mrs. V.Geetha, M.C.A., M.Phil**, Assistant Professor, Department of Computer Applications, Kumaraguru College of Technology, Coimbatore for her complete assistance, guidance and support given to me throughout my project.

I would express heartfelt thanks to **Ms. C.Geethamani, M.C.A., M.Phil**, Lecturer, Department of Computer Applications, Kumaraguru College of Technology, Coimbatore as with out his best guidance it would not have been possible for me to successfully complete this project who also gave his innovative ideas at crucial times and tremendous encouragement.

I also dedicate equal and grateful acknowledgements to all the respectable members of the faculty and lab in-charges of the **Department of Computer Applications, Kumaraguru College of Technology, Coimbatore** and student friends for their motivation, encouragement and continuous support.

TABLE OF CONTENTS

NO	TITLE	PAGE NO
	Abstract	ii
	List of Figures	vi
1	Introduction	
	1.1 Organization profile	1
	1.2 Problem definition	3
2	System Analysis	
	2.1 Existing System Architecture	4
	2.2 Problem in Existing System	4
	2.3 Proposed System Architecture	4
	2.4 Advantages of Proposed System	5
3	Development Environment	
	3.1 Hardware Environment	6
	3.2 Software Environment	6
	3.3 Software Description	7
4	Analysis of Requirements	
	4.1 Assumptions	15
	4.2 System architecture	15
	4.3 Architectural goals and constraints	15
	4.4 Layering architecture	17
5	System Design	
	5.1 Use case Diagram	20
	5.2 Activity Diagram	22

5.3 Data flow Diagram	24
6 System Testing	
6.1 Testing methodology	27
6.1.1 Unit testing	27
6.1.2 Integration testing	28
6.1.3 Verification & Validation Testing	28
6.1.4 System testing	29
6.2 Test plan	30
6.2.1 Test environment	30
6.2.2 Features to be tested	30
7 Conclusion	31
8 Appendices	
8.1 Screen shots	33
9 References	47



YOUTH SOFT

30th May 2008

PROJECT COMPLETION CERTIFICATE

This is certify to that Mr.N.Saravanakumar (Reg.No: 71205621038) doing final year M.C.A in Kumaraguru College of Technology has completed his project entitled “TRANSACTIONS ON IMAGE PROCESSING” Under the guidance of Mr. B. Pradeep Kumar, Senior Software Engineer, in our concern during the period of December 2007 to May 2008.

He has successfully completed the project as per the requirements.

R.SANTHOSHKUMAR
Project Manager

ABSTRACT

The primary objective of the project, '**TRANSACTIONS ON IMAGE PROCESSING**' is restoration of cracks on digitized paintings. Crack detection and restoration can provide clues to art historians, museum curators and the general public on how the painting would look like in its initial state, i.e., without the cracks.

Digital Image Processing is a rapidly evolving field with growing applications in science and engineering. Image processing holds the possibility of developing the ultimate machine that could perform the visual functions of all living beings. Many theoretical and technological breakthroughs are required before we could build such a machine that is there is an abundance of image processing applications that can serve mankind with the available and anticipated technology in the near future. Imaging began in the 19th century with photography and continued with x-rays, television and electronic scanning in the 20th century. Image processing as a field of study began in the 1950s with pictures of the earth from high flying spy airplanes and then with pictures of the earth's surface taken from orbiting satellites.

An integrated methodology for the detection and removal of cracks on digitized paintings is presented in this project. The cracks are detected by thresholding the output of the morphological top-hat transform. Afterward, the thin dark brush strokes which have been misidentified as cracks are removed using either a median radial basis function neural network on hue and saturation data or a semi-automatic procedure based on region growing. Finally, crack filling using order statistics filters or controlled anisotropic diffusion is performed. The methodology has been shown to perform very well on digitized paintings suffering from cracks.

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	Image co-ordinates	12
3.2	Display of a Pixel Value	14
5.1	Use Case Diagram	21
5.2	Activity Diagram	23
5.3	Level-0 DFD	24
5.3	Level-1 DFD	25
5.3	Level-2 DFD	26

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

Established in the year 2001 Youth Soft having its operation in Chennai - India, has experienced to serve global and local industrial outfits that belong to many functional Domains. Youth Soft acts as an independent application development division (Youth Soft - **Software Development Group**), which contributes a few list of application software to the Domestic and Foreign Business community.

Apart from domestic software development and support, Youth Soft has a principle tie up with M/s Valiant Ship Management Limited Hong Kong a multinational business entity involved in the Ship Management products, for IT support and specializing in Marine Asset Management, As Youth Soft - Marine Asset Management Information System Division).

With the wide spread knowledge of the principle Youth Soft has acclaimed global recognition in implementing systems for various ship management companies across South East Asia

With the continuous effort in understanding various functional and technical titles, Youth Soft is rigorously working towards improvising knowledge in the functional and technical areas. Youth Soft's Functional expertise extends to managing and developing Management Information Systems belonging to

- Materials Management
- Production and Maintenance Planning
- Educational Institution Management
- Portfolio Management
- Safety Management procedures
- Protocols and middle level
- POS (Point of Sale) sector Management etc.

Excluding the above, any functional area could be addressed with proper system and application through proper individual or group outsourcing depending upon the requirement of the market. The technology areas related to development of systems

Microsoft Development Environment

Visual Studio .NET

VB

ASP

MS SQL Server 2000

Crystal reports 9 + V , Crystal Decisions XI

Data Mining, *Web Intelligence and Business Intelligence using MS Data warehousing Tools (Informatica - ETL Tool). Testing Tools which includes Win Runner, Load Runner, Rational Rose and Silk. Apart from the same development of Custom made API's, Independent Components and Supporting Patches and plugin's are also done periodically to attain user satisfaction and requirement filling process.

Youth Soft also does frequent and periodical review of technical and functional expertise, and improvises to any latest trends and technology available in the market through perennial knowledge management programs. By which Youth Soft confers to a standard development promise that it can keep up to the requirement of the client even if falling out of boundaries than the existing expertise

1.2 PROBLEM DEFINITION

Many paintings, especially old ones, suffer from breaks in the substrate, the paint, or the varnish, called Cracks. Appearance of cracks on paints deteriorates the perceived image quality. Digital image processing (DIP) remains a challenging domain of programming for several reasons. Also, several image processing techniques require the most careful optimizations and especially for real time applications. Digital paintings are of great historical importance. Study and implement the proposed methodology for the restoration of cracks on digitized paintings. Crack detection and restoration can provide clues to art historians, museum curators and the general public on how the painting would look like in its initial state, i.e., without the cracks.

A technique that decomposes the image to textured and structured areas and uses appropriate interpolation techniques depending on the area where the missing information lies has also been proposed. The results obtained by these techniques are very good. A methodology for the restoration of cracks on digitized paintings, which adapts and integrates a number of image processing and analysis tools is proposed in this paper. The methodology is an extension of the crack removal framework presented.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM ARCHITECTURE

- Multi-oriented Gabor filters
- Motion Detection Methods
- Image Inpainting
- Image Decomposition-based Missing Information Detection

2.2 PROBLEMS IN EXISTING SYSTEM

- Gabor filters require manual interaction.
- Motion-detection methods rely on information obtained over several adjacent frames for filling.
- Inpainting assumes that the regions where information has to be filled in are known.
- Decomposition-based techniques require more computational resources and hence slow.

2.3 PROPOSED SYSTEM ARCHITECTURE

Proposed method combines several DIP techniques for:

Crack detection

Cracks usually have low luminance and, thus, can be considered as local intensity minima with rather elongated structural characteristics. Therefore, a crack detector can be applied on the luminance component of an image and should be able to identify such minima.

Separation of brush strokes from the cracks

In some paintings, certain areas exist where brush strokes have almost the same thickness and luminance features as cracks. The hair of a person in a portrait could be such an area. Therefore, the top-hat transform might misclassify these dark brush strokes as cracks. Thus, in order to avoid any undesirable alterations to the original image, it is very important to separate these brush strokes from the actual cracks, before the implementation of the crack filling procedure.

Crack filling (interpolation)

After identifying cracks and separating misclassified brush strokes, the final task is to restore the image using local image information (i.e., information from neighboring pixels) to fill (interpolate) the cracks. Two classes of techniques, utilizing order statistics filtering and anisotropic diffusion are proposed for this purpose. Both are implemented on each RGB channel independently and affect only those pixels which belong to cracks. Therefore, provided that the identified crack pixels are indeed crack pixels, the filling procedure does not affect the “useful” content of the image.

2.4 Advantages of Proposed Method

- All processing steps can be executed in real-time, and thus the user can instantly observe the effect of parameter tuning on the image under study and select in an intuitive way the values that achieve the optimal value result.
- Only little manual interaction is required but can achieve high optimal results customized for domain experts.

CHAPTER 3

DEVELOPMENT ENVIRONMENT

3.1 H/W ENVIRONMENT

Main Processor	–INTEL PENTIUM IV
RAM	– 512 MB
Hard Disk Capacity	–40 GB
Monitor	– HP17”
Keyboard	– HP 106 keys
Mouse	– HP Optical Scroll Mouse

3.2 S/W ENVIRONMENT

I. Technologies

- a. Java 2 Platform, Standard Edition 1.4.2 (J2SE)
- b. Java Creator 350

3.3 SOFTWARE DESCRIPTION

WINDOWS XP PROFESSIONAL

Windows XP Professional integrates the strengths of Windows 2000, Windows 98 and Windows Millennium Edition such as standards-based security, manageability, and reliability.

The Microsoft® Windows® XP Professional operating system includes a variety of technologies that communicate with the Internet to provide increased ease of use and functionality. Browser and e-mail technologies are obvious examples, but there are also technologies such as Automatic Updates that help users obtain the latest software and product information, including bug fixes and security patches. These technologies provide many benefits, but they also involve communication with Internet sites, which administrators might want to control.

Control of this communication can be achieved through a variety of options built into individual components, into the operating system as a whole, and into server components designed for managing configurations across your organization. For example, as an administrator, you can use Group Policy to control the way some components communicate. For some components, you can direct all communication to the organization's own internal Web site instead of to an external site on the Internet.

This white paper provides information about the communication that flows between components in Windows XP Professional with either Service Pack 1 or Service Pack 1a (both referred to as "SP1" in this white paper) and sites on the Internet, and describes steps to take to limit, control, or prevent that communication in an organization with many users. The white paper is designed to assist you, the administrator, in planning strategies for deploying and maintaining Windows XP Professional with SP1 in a way that helps to provide an appropriate level of security and privacy for your organization's networked assets.

JAVA

The Java programming language is a state-of-the-art, object-oriented language that has syntax similar to that of C. The language designers strove to make the Java language powerful, but, at the same time, they tried to avoid the overly complex features that have bogged down other object-oriented languages like C++. By keeping the language simple, the designers also made it easier for programmers to write robust, bug-free code. As a result of its elegant design and next-generation features, the Java language has proved popular with programmers, who typically find it a pleasure to work with Java after struggling with more difficult, less powerful languages.

THE JAVA VIRTUAL MACHINE

The Java Virtual Machine, or Java interpreter, is the crucial piece of every Java installation. By design, Java programs are portable, but they are only portable to platforms to which a Java interpreter has been ported. Sun ships VM implementations for its own Solaris operating system and for Microsoft Windows and Linux platforms. Many other vendors, including Apple and various commercial UNIX vendors, provide Java interpreters for their platforms. The Java VM is not only for desktop systems, however. It has been ported to set-top boxes and handheld devices that run Windows CE and PalmOS. 1.1.3. The Java platform is just as important as the Java programming language and the Java Virtual Machine. All programs written in the Java language rely on the set of predefined classes that comprise the Java platform. Java classes are organized into related groups known as packages. The Java platform defines packages for functionality such as input/output, network

JAVA IS OBJECT-ORIENTED

To some, object-oriented programming (OOP) technique is merely a way of organizing. Programs and it can be accomplished using any language. Working with a

real object-oriented language and programming environment, however, enables you to take full advantage of object-oriented methodology and its capabilities of creating flexible, modular programs and reusing code. Many of Java's object-oriented concepts are inherited from C++, the language on which it is based, but it borrows many concepts from other object-oriented languages as well. Like most object-oriented programming languages, Java includes a set of class libraries that provide basic data types, system input and output capabilities, and other utility functions. These basic classes are part of the Java development kit, which also has classes to support networking, common Internet protocols, and user interface toolkit functions.

JAVA IS PLATFORM-INDEPENDENT

Platform independence is one of the most significant advantages that Java has over other programming languages, particularly for systems that need to work on many different platforms. Java is platform-independent at both the source and the binary level. Normally, when you compile a program written in C or in most other languages, the compiler translates your program into machine codes or processor instructions. Those instructions are specific to the processor your computer is running—so, for example, if you compile your code on a Pentium system, the resulting program will run only on other Pentium systems. If you want to use the same program on another system, you have to go back to your original source, get a compiler for that system, and recompile your code. Things are different when you write code in Java. The Java development environment has two parts: a Java compiler and a Java interpreter. The Java compiler takes your Java program and instead of generating machine codes from your source files, it generates byte codes.

WRITE ONCE, RUN ANYWHERE

Sun identifies "Write once, run anywhere" as the core value proposition of the Java platform. Translated from business jargon, this means that the most important promise of Java technology is that you have to write your application only once for the

Java platform and then you'll be able to run it anywhere. King, graphics, user-interface creation, security, and much more.

SECURITY

Another key benefit of Java is its security features. Both the language and the platform were designed from the ground up with security in mind. The Java platform allows users to download untrusted code over a network and run it in a secure environment in which it cannot do any harm: untrusted code cannot infect the host system with a virus, cannot read or write files from the hard drive, and so forth. This capability alone makes the Java platform unique

NETWORK-CENTRIC PROGRAMMING

Sun's corporate motto has always been "The network is the computer." The designers of the Java platform believed in the importance of networking and designed the Java platform to be network-centric. From a programmer's point of view, Java makes it easy to work with resources across a network and to create network-based applications using client/server or multitier architectures.

DYNAMIC, EXTENSIBLE PROGRAMS

Java is both dynamic and extensible. Java code is organized in modular object-oriented units called classes. Classes are stored in separate files and are loaded into the Java interpreter only when needed. This means that an application can decide as it is running what classes it needs and can load them when it needs them. It also means that a program can dynamically extend itself by loading the classes it needs to expand its functionality. The network-centric design of the Java platform means that a Java application can dynamically extend itself by loading new classes over a network. An application that takes advantage of these features ceases to be a monolithic block of code.

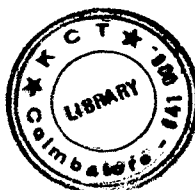
Instead, it becomes an interacting collection of independent software components. Thus, Java enables a powerful new metaphor of application design and development.

INTERNATIONALIZATION

The Java language and the Java platform were designed from the start with the rest of the world in mind. When it was created, Java was the only commonly used programming language that had internationalization features at its core rather than tacked on as an afterthought. While most programming languages use 8-bit characters that represent only the alphabets of English and Western European languages, Java uses 16-bit Unicode characters that represent the phonetic alphabets and ideographic character sets of the entire world. Java's internationalization features are not restricted to just low-level character representation, however. The features permeate the Java platform, making it easier to write internationalized programs with Java than it is with any other environment.

PERFORMANCE

As described earlier, Java programs are compiled to a portable intermediate form known as byte codes, rather than to native machine-language instructions. The Java Virtual Machine runs a Java program by interpreting these portable byte-code instructions. This architecture means that Java programs are faster than programs or scripts written in purely interpreted languages, but Java programs are typically slower than C and C++ programs compiled to native machine language. Keep in mind, however, that although Java programs are compiled to byte code, not all of the Java platform is implemented with interpreted byte codes. For efficiency, computationally intensive portions of the Java platform such as the string-manipulation methods are implemented using native machine code.



P-2286

IMAGE DETAILS

PIXELS

An image is a rectangular array of dots called pixels (picture elements) where the number of rows M and number of columns N of dots in an image are specified. At each row-column intersection (m, n) there is a pixel, or picture element. The point (m, n) is the location of the pixel, while the pixel value at that location is designated by $p(m,n)$, or sometimes by $f(m,n)$ or $f(x,y)$. The following figure shows the location (m, n) , where $0 < m < M-1$ and $0 < n < N-1$. Note that in the figure the downward vertical direction is x and y is the horizontal rightward direction. The origin is in upper left corner.

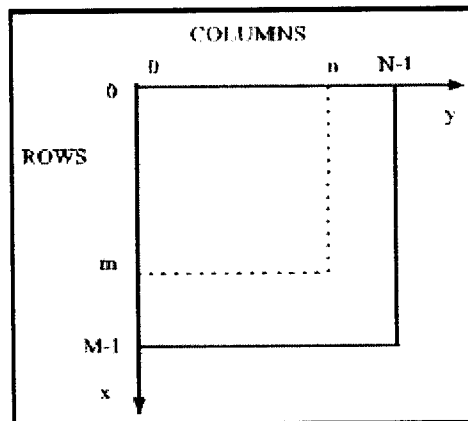


Fig.3.1 Image co-ordinates

PIXEL VALUES

The pixel values in an image may be:

- Grayscale (or)
- Color

GRAYSCALE PIXEL VALUES

The grayscale images are simpler and each pixel value corresponds to one byte. Grayscale usually has a range from 0 (no light intensity, or full black) to 255 (full light intensity, or full white), in integer steps. Thus the 256 grayscale values for pixels are 0, 1, 2... 255. Each of these takes one byte of storage in a computer or in a file, so if an image has $M \times N = 256 \times 256 = 65,536$ pixels then that image takes 65,536 bytes of pixel storage in computer memory or on a storage device. An $M \times N = 1024 \times 1280$ image has 1,310,720 pixels. A pixel value $p(x,y)$ or $p(m,n)$, where $0 \leq m \leq M$ and $0 \leq n \leq N$, is a byte (0 to 255 in binary) in grayscale or 3 bytes in color, where the respective bytes are the red (R), green (G) and blue (B) values.

COLOR PIXEL VALUES

Color images most often take one of two different forms. The most common method is called **true-color** and uses one byte for each of red, green and blue. Thus a single pixel value requires 3 bytes of memory or disk storage. From these values we can form $(256) \times (256) \times (256) = 16,777,216$ discrete colors, which is about the maximum number of different colors that humans can distinguish. An $M \times N = 256 \times 256$ color image of three bytes per pixel would then require $3(65,536) = 196,608$ bytes. For an $M \times N = 1024 \times 1280$ color image the requirement is $3(1,310,720) = 3,932,160$. It is clear that more colors and more pixels are more costly in computer storage and time to send on the Internet.

An older format for color is to allow only 256 colors at any one time on the screen. A byte indicates a color but it is actually the address from 0 to 255 of one of 256 color registers, each of which contains 18 bits for 6 bits each of R, G and B. The 256 color set, called a palette, must be loaded into the registers before the image can be displayed, or else the palette can be read from the image file.

GRAYSCALE PROCESSING

Grayscale images are suitable and enough for any image processing application/project to apply and analyze all types of image-processing techniques such as interpolation, filtering, enhancing, etc. Even when we process color images we often process the intensity part, which is grayscale, and then put the color back into the processed image. In color-images, true-color presents all the colors that human eye can visualize. This is suitable for many types of images, but the current trend is toward even more colors than true-color, which may be a waste of resources due to the fact that such fine resolution of color is wasted on humans.

The following figure shows the grayscale pixel values as a function $f(m,n)$ of the pixel locations at rows m and columns n . Thus we can picture an image as a 3-D surface that has elevation (gray level) as range above the image plane that is the domain. The gray levels are discrete values, so the surface is made of discrete steps at the pixels.

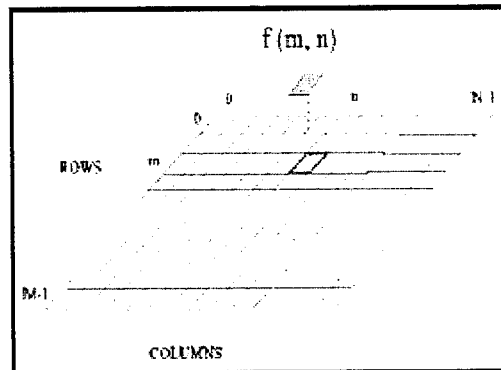


Fig.3.2 Display of a Pixel Value

CHAPTER 4

ANALYSIS OF REQUIREMENTS

4.1 ASSUMPTIONS

A certain degree of user interaction, most notably in the crack-detection stage, is required for optimal results. User interaction is rather unavoidable since the large variations observed in the typology of cracks would lead any fully automatic algorithm to failure. However, all processing steps can be executed in real time, and, thus, the user can instantly observe the effect of parameter tuning on the image under study and select in an intuitive way the values that achieve the optimal visual result. Needless to say, only subjective optimality criteria can be used in this case since no ground truth data are available. The opinion of restoration experts that inspected the virtually restored images was very positive

4.2 SYSTEM ARCHITECTURE

An integrated methodology for the detection and removal of cracks on digitized paintings is presented in this paper. The cracks are detected by thresholding the output of the morphological top-hat transform. Afterward, the thin dark brush strokes which have been misidentified as cracks are removed using either a median radial basis function neural network on hue and saturation data or a semi-automatic procedure based on region growing. Finally, crack filling using order statistics filters or controlled anisotropic diffusion is performed. The methodology has been shown to perform very well on digitized paintings suffering from cracks

4.3 ARCHITECTURAL GOALS AND CONSTRAINTS

The appearance of cracks on paintings deteriorates the perceived image quality. However, one can use digital image processing techniques to detect and eliminate the cracks on digitized paintings. Such a “virtual” restoration can provide clues to art historians, museum curators and the general public on how the painting would look like

in its initial state, i.e., without the cracks. Furthermore, it can be used as a nondestructive tool for the planning of the actual restoration. A system that is capable of tracking and interpolating cracks is presented. The user should manually select a point on each crack to be restored.

A method for the detection of cracks using multi oriented Gabor filters is presented. Crack detection and removal bears certain similarities with methods proposed for the detection and removal of scratches and other artifacts from motion films. However, such methods rely on information obtained over several adjacent frames for both artifact detection and filling and, thus, are not directly applicable in the case of painting cracks. Other research areas that are closely related to crack removal include image inpainting which deals with the reconstruction of missing or damaged image areas by filling in information from the neighboring areas, and disocclusion, i.e., recovery of object parts that are hidden behind other objects within an image.

Methods developed in these areas assume that the regions where information has to be filled in are known. Different approaches for interpolating information in structured and textured image areas have been developed. The former are usually based on partial differential equations (PDEs) and on the calculus of variations whereas the latter rely on texture synthesis principles. A technique that decomposes the image to textured and structured areas and uses appropriate interpolation techniques depending on the area where the missing information lies has also been proposed. The results obtained by these techniques are very good. A methodology for the restoration of cracks on digitized paintings, which adapts and integrates a number of image processing and analysis tools is proposed in this paper. The methodology is an extension of the crack removal framework presented.

4.4 LAYERING ARCHITECTURE

IMAGE FILE PROCESSING

To understand how image files are stored and their file formats, and perform basic operations like reading, writing and drawing an image on screen.

IMAGE FILES

An image is stored in a particular file format. The most popular formats nowadays are GIF (Graphics Interchange Format), JPEG (Joint Photographic Experts Group), PNG (Portable Network Graphics), TIFF (Tagged Image File Format), PGM (Portable Gray Map) and PPM (Portable Pixel Map).

PNM FILE FORMAT

PNM is the acronym for Portable Any Map. The PNM images are the simplest image file formats for processing. These files contain the actual pixel values without any compression or decomposition of the data into file sections; and hence these images are more suitable for any image processing application. PNM images are of three types, namely portable bit-maps (PBM), portable gray-maps (PGM), and portable pix-maps (PPM). These formats are a convenient (simple) method of saving image data as they are equally easy to read in ones own applications.

PBM FILE FORMAT

PBM images are for storing black and white images. PBM stores single bit pixel image as a series of ASCII 0s or 1s. Traditionally 0 refers to white while 1 refers to black. The header is identical to PPM and PGM format except there is no maximum pixel value in the third header line, as it doesn't have any meaning here. The magic identifier for PBM is P1.

PGM FILE FORMAT

PGM images are grayscale image file formats. It is of two types, containing magic identifiers namely P2 and P5. P2 stores pixel values as the ASCII characters for the digits, delimited by spaces between consecutive pixel values. P5 writes the bytes without delimiters binary numbers from 0 to 255. It is often used by image processing researchers around the world for comparison of their methods and algorithms with other ones and appears in most journals on image processing.

We see that the P5 type of PGM is packed (without delimiters) so there is a single byte for each pixel. This is also called the raw data format. The size of this file is 65,576 bytes. The P2 type of PGM uses a byte for each numerical symbol (digit) and therefore requires three bytes for each number greater than 99 and also uses a space character after each pixel value. Thus the file is nearly four times as large. This P2 file is 245,724 bytes. However, humans can read the P2 type of PGM file, whereas they can not read the ASCII characters of the packed bytes, which can appear different in different editors (characters 128 to 255).

PPM FILE FORMAT

PPM images are RGB color images, which magic identifiers P3 for ASCII data and P6 for binary data. P6 image files are obviously smaller than P3 and much faster to read. Note that P6 PPM files can only be used for single byte colors. The components are stored in the usual order, red - green - blue.

CRACK DETECTION

The basic characteristics of cracks are that they have elongated structural characteristics. Crack-detection is carried out using top-hat transform. Top-hat

transform is basically a grayscale morphological filter that transforms the input grayscale image into another grayscale output image where pixels with a large gray value are potential crack or crack-like elements.

REMOVAL OF MIS-IDENTIFIED CRACKS

Apply thresholding to separate cracks from the rest of the image. Manual interaction is required in this module. A binary image will be created showing only the crack regions is created.

CRACK REMOVAL

Use trimmed mean filter to remove the cracks. The thresholded binary image and top-hat transformed image are stacked over to identify cracks and then the cracks from the original image are removed using the pixels from the regions around the cracks.

CHAPTER 5

SYSTEM DESIGN

5.1 USECASE DIAGRAM

An actor initiates a use case, and an actor (possibly the initiator, but not necessarily) receives something of value from the use case. The graphic representation is straightforward: An ellipse represents a use case, and a stick figure represents an actor. The initiating actor is on the left of the use case, and the receiving actor is on the right. (Many modelers omit the receiving actor, and the UML 2.0 specification doesn't mention it.) The actor's name appears just below the actor. The name of the use case appears either inside the ellipse or just below it. An association line connects an actor to the use case, and represents communication between the actor and the use case. The association line is solid, like the line that connects associated classes.

One of the benefits of use case analysis is that it shows the boundary between the system and the outside world. Actors are typically outside the system, whereas use cases are inside. You use a rectangle (with the name of the system somewhere inside) to represent the system boundary. The rectangle encloses the system's use cases.

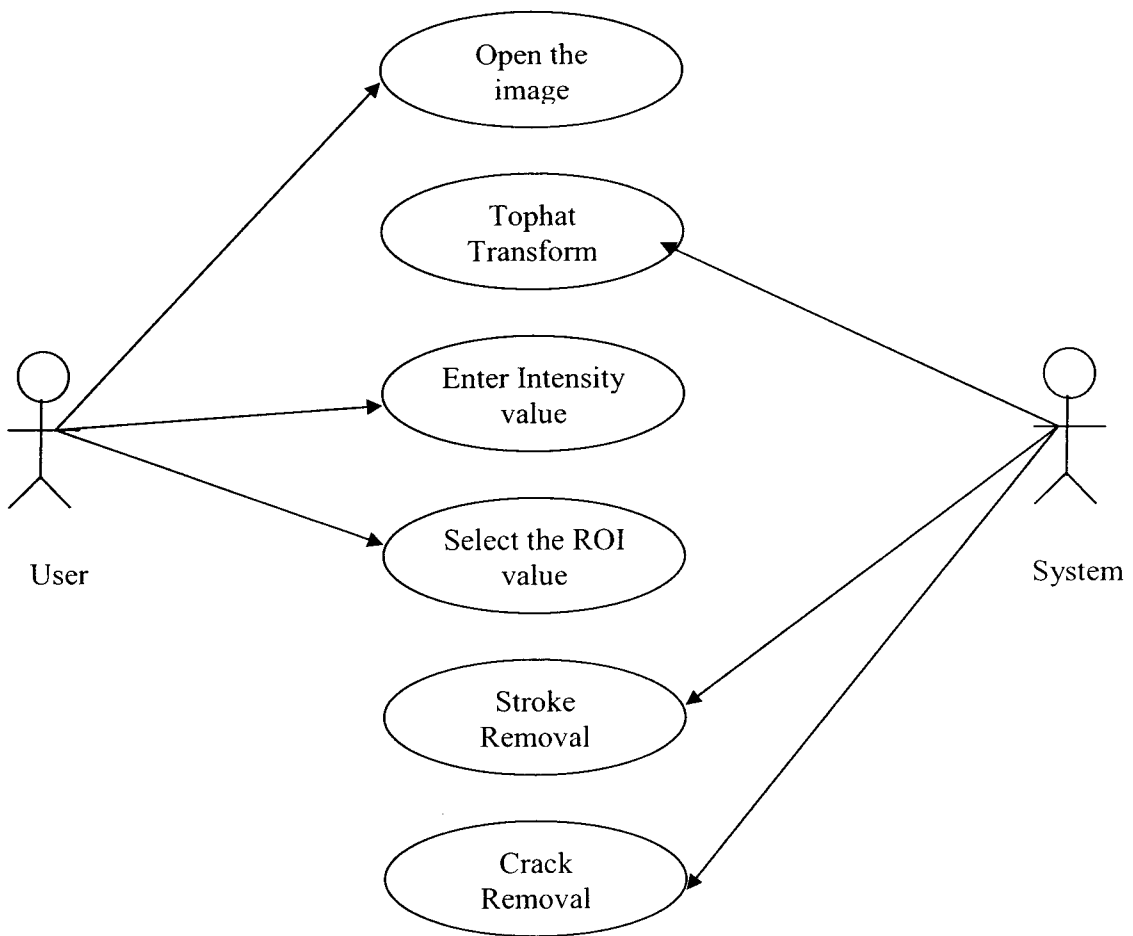


Figure 5.1 Use Case Diagram

5.2 ACTIVITY DIAGRAM

The processing within an activity goes to completion and then an automatic transmission to the next activity occurs. An arrow represents the transition from one activity to the next. Like the state diagram, the activity diagram has a starting point represented by a filled-in circle and an endpoint represented by a bull's-eye. A sequence of activities almost always comes to a point where a decision has to take place. One set of conditions leads to one path, another set of conditions to another path, and the two paths are mutually exclusive. You can represent a decision point in either of two ways. (Sounds like a decision.) One way is to show the possible paths coming directly out of an activity. The other is to have the activity transition to a small diamond—reminiscent of the decision symbol in a flowchart—and have the possible paths flow out of the diamond. (As an old flow charters, I prefer the second way.) Either way, you indicate the condition with a bracketed condition statement near the appropriate path.

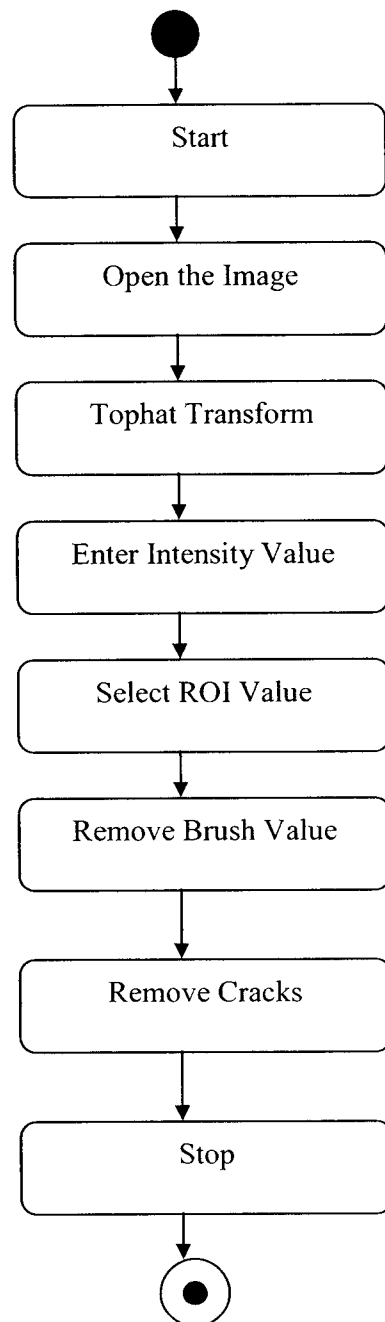


Figure 5.2 Activity Diagram

5.3 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) is a graphical representation of the flow of process through an information system.

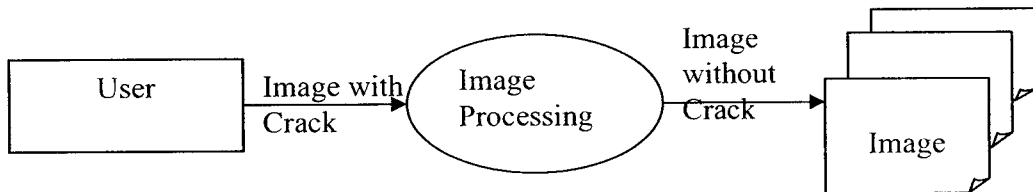


Figure 5.3 Level-0 DFD Diagram

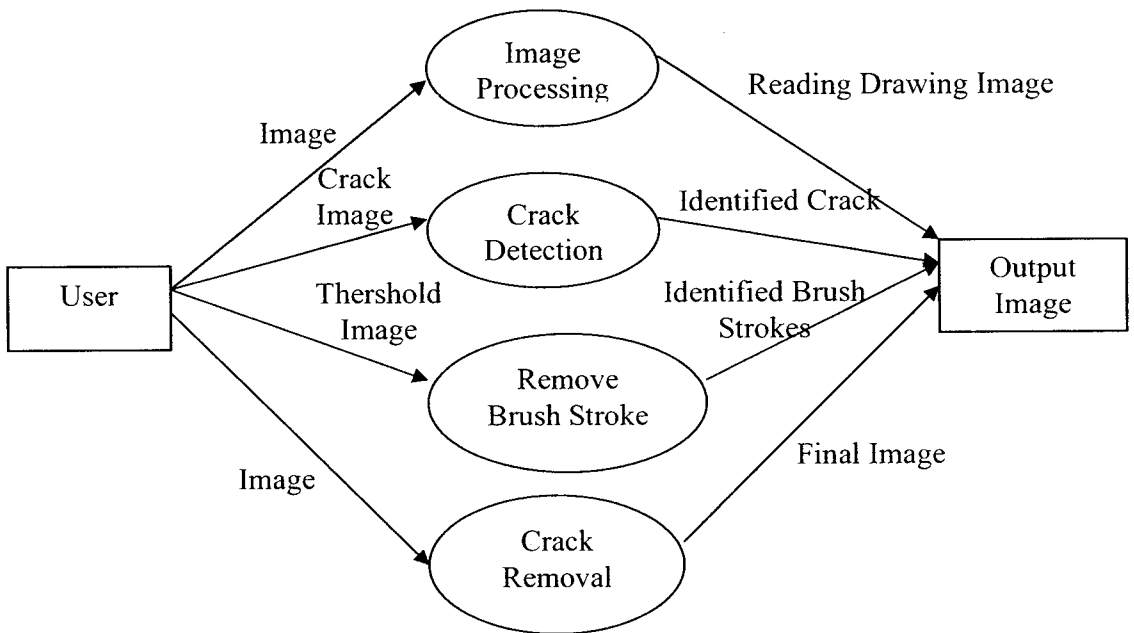


Figure 5.4 Level-1 DFD Diagram

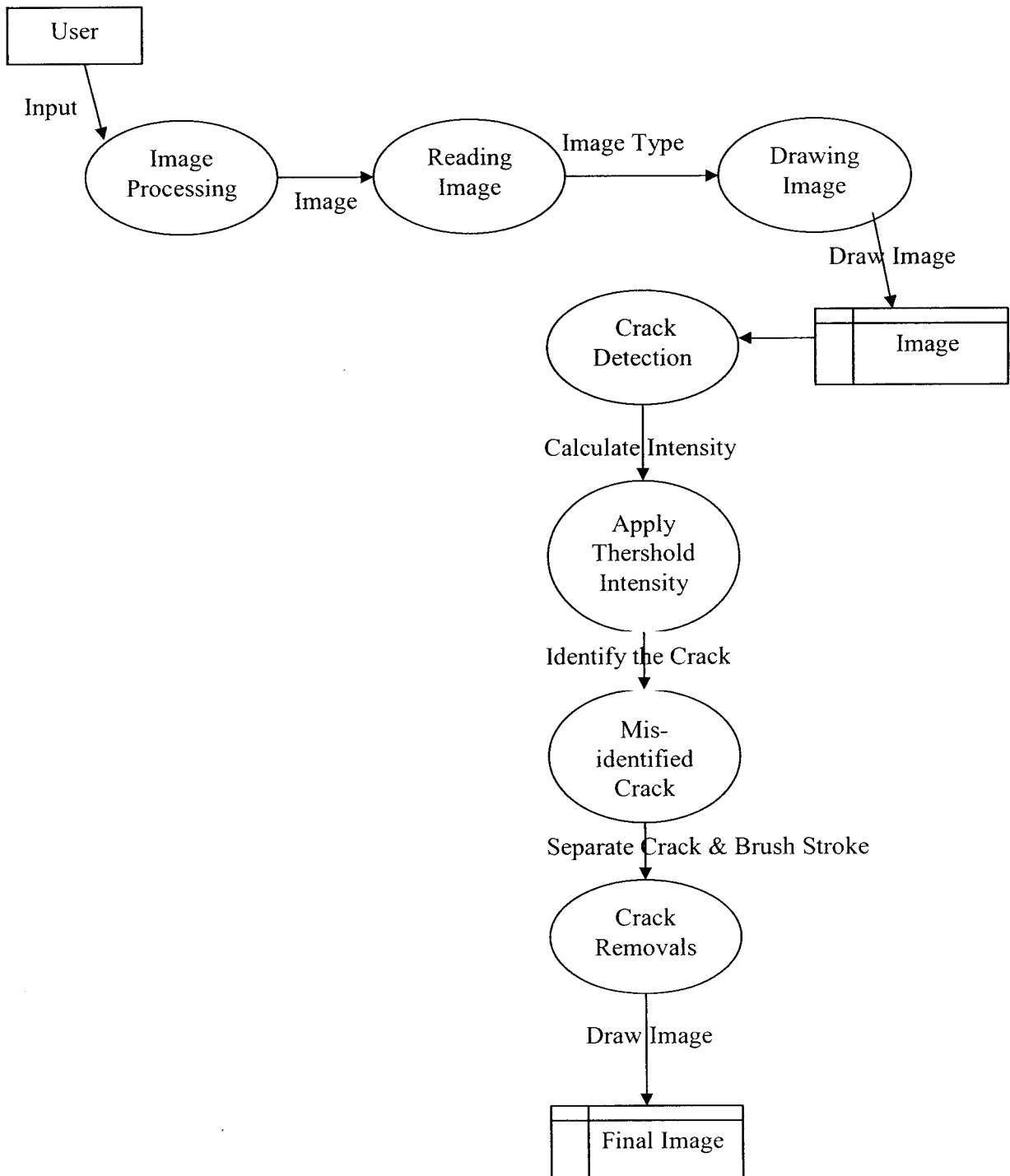


Figure 5.4 Level-2 DFD Diagram

CHAPTER 6

SYSTEM TESTING

6.1 TESTING METHODOLOGY

6.1.1 UNIT TESTING

A unit test is a procedure used to validate that a particular module of source code is working properly. Thus, we can say that this is a module-level testing where each of the modules are tested individually. This type of testing is mostly done by the developers and not by end-users. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing provides a strict, written contract that the piece of code must satisfy.

Here in this project, we have tested each pair of modules before it could be integrated and packaged. Each module pair has been tested for their functionality. They were originally developed to run as CLI (Command Line Interface) and were forced to undergo Black Box testing, White Box Testing and Domain testing. At later stage in the system development, a GUI was developed to view the monitored status.

Test 1:

Procedure

The mandatory fields have to be filled before proceeding to next process.

Solution

The alert message has to be displayed to fill the mandatory details.

Test 2:

Procedure

Enter intensity value to identify the cracks and it displays the identified cracks.

Solution

This problem is solved by re-enter the intensity values and repeats the process again and again to identify the cracks.

Test 3:**Procedure**

When an image is in progress and if a user tries to perform another operation.

Solution

The prompt is made that the image is in progress and want to continue with the refresh button.

6.1.2 INTEGRATION TESTING

Integration testing is testing the software after all its modules are put together. The problem, of course, is “putting them together” that is interfacing. Data can be lost across when sub-functions are combined and it may not produce the desired major function individually may be magnified to unacceptable levels and global data structures can present problems. Integration Testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

There is often a tendency to attempt Non-increment Integration; that is to construct the program using a “big Band” approach. All modules are combined in advance. The entire program is tested as a whole and chooses usually results. Sets of errors are encountered. Corrections are difficult because isolating the causes is complicated by the vast expanse of the entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

6.1.3 VERIFICATION AND VALIDATION TESTING:

Verification refers to the set of activities that ensure that system correctly implements a specific function. Validation refers to a different set of activities that ensure that the system that has been built is traceable to customer requirements. Verification and validation encompass a wide array of software quality assurance (SQA) activities that

include formal reviews, quality and configuration audits, performance monitoring, simulation, feasibility study, documentation review, database review, algorithm analysis, development testing, qualification testing and installation testing.

Open an Image:

All the pixels are read by the system to draw the image with all pixels and to read dimension of the image.

Intensity Value

The numeric fields must not contain any alphabets or special characters. This validation is checked.

ROI Value

The character field must not contain any special characters. This validation is checked

6.1.4 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

6.2 TEST PLAN

6.2.1 TEST ENVIRONMENT

- Test cases are prepared according to design / specifications.
- Each Test case and the items under test are documented in detail in a set format.
- The input and the initial values are then filled in the Unit Test Plan (UTP) as per each test case & theoretical values evaluated.
- The UTP is then reviewed and executed.

6.2.2 FEATURES TO BE TESTED

The following functional features will be tested:

- Opening the image
- Tophat Transform
- Enter the different intensity values
- Select the ROI Value
- Algorithm for stroke removal
- Algorithm for Crack removal

Additionally testing will be done simultaneously from multiple clients to ensure that all data changes are done online.

CHAPTER 7

CONCLUSION

Cracks are detected by using top-hat transform, whereas the thin dark brush strokes, which are misidentified as cracks, are separated either by an automatic technique by a semi-automatic approach. Crack interpolation is performed by appropriately modified order statistics filters or controlled anisotropic diffusion. The methodology has been applied for the virtual restoration of images and was found very effective by restoration experts.

However, there are certain aspects of the proposed methodology that can be further improved. For example, the crack-detection stage is not very efficient in detecting cracks located on very dark image areas, since in these areas the intensity of crack pixels is very close to the intensity of the surrounding region. A possible solution to this shortcoming would be to apply the crack-detection algorithm locally on this area and select a low threshold value.

The application is formulated by analyzing the requirements of the users in the company. Each and every module has undergone various test conditions. With a full stretch testing, it has been ensured that the system can enhance ideally without any bugs or crashes, which will make the end user more compatible with the project. The application is designed as user friendly and all the options available are clear and self explanatory so that the user can understand the system easily.

FUTURE ENHANCEMENT:

Use of image Inpainting techniques could also improve results in that aspect. Another improvement of the crack filling stage could aim at using properly adapted versions of nonlinear multichannel filters (e.g., variants of the vector median filter) instead of processing each color channel independently. These improvements will be the topic of future work on this project.

Another situation where the system (more particularly, the crack filling stage) does not perform as efficiently as expected is in the case of cracks that cross the border between regions of different color. In such situations, it might be the case that part of the crack in one area is filled with color from the other area, resulting in small spurs of color in the border between the two regions.. However, this phenomenon is rather seldom and, furthermore, the extent of these erroneously filled areas is very small (2–3 pixels maximum). A possible solution would be to perform edge detection or segmentation on the image and confine the filling of cracks that cross edges or region borders to pixels from the corresponding region.

CHAPTER 8

APPENDICES

8.1 Screen shots

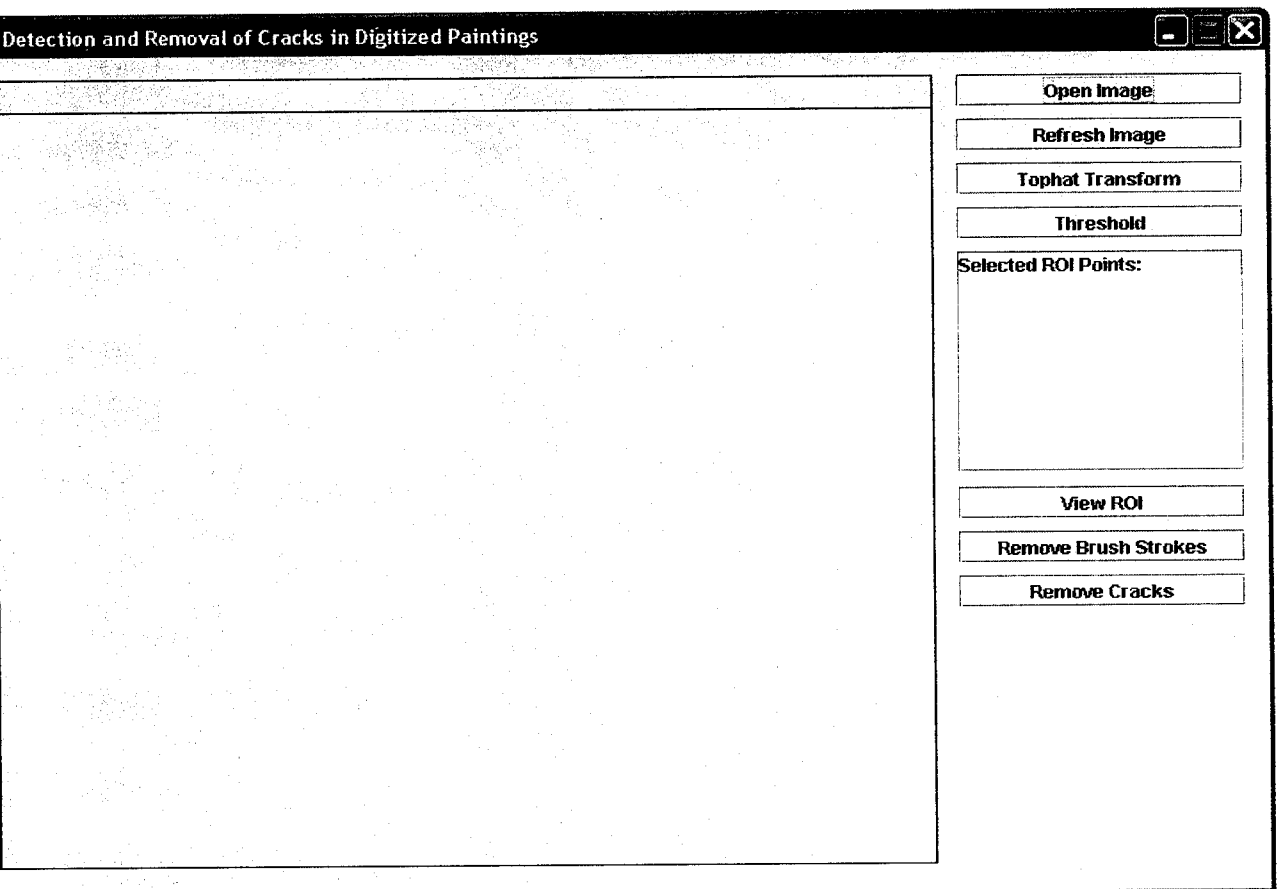


Figure A 1.1 shows the dialog box to execute the functions of the system

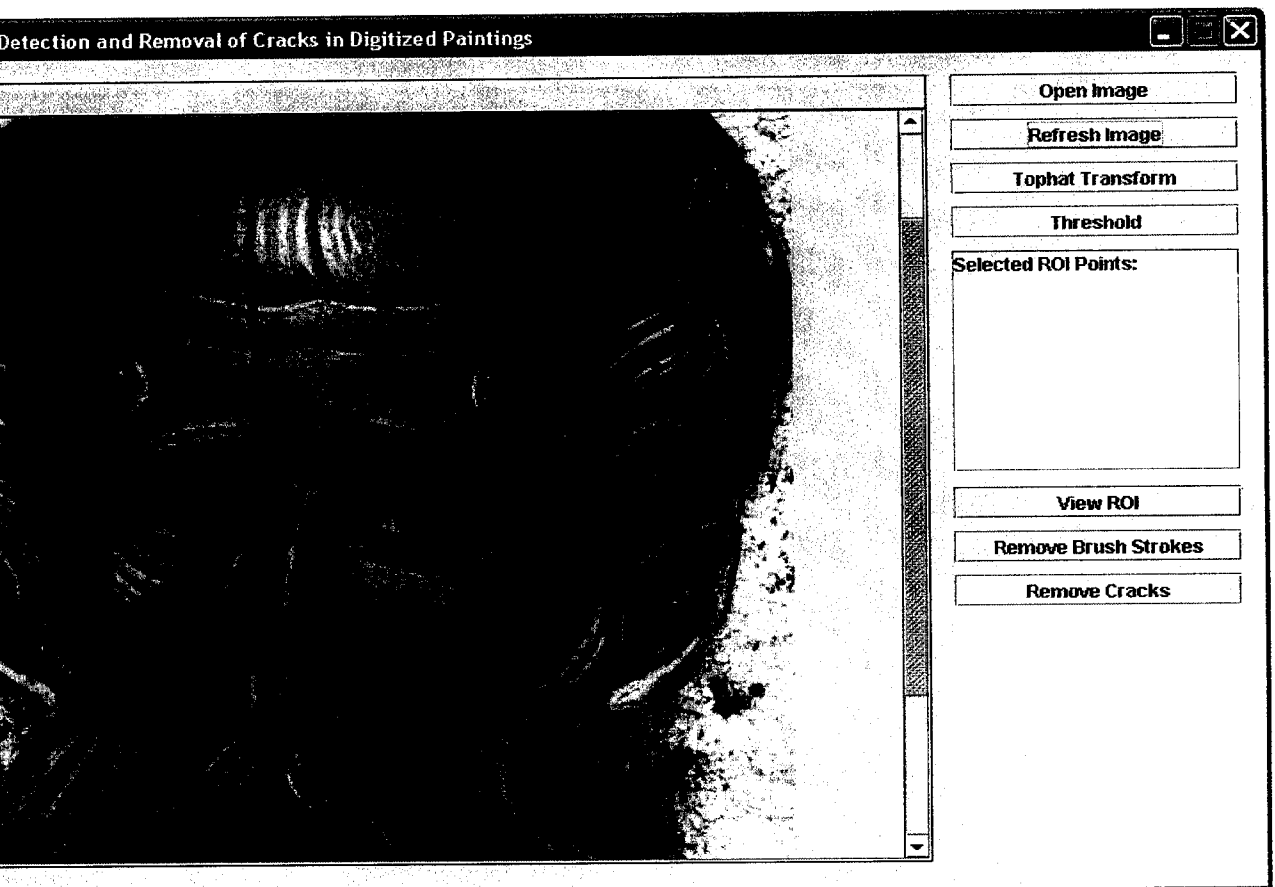


Figure A 1.2 shows the dialog box to open the image

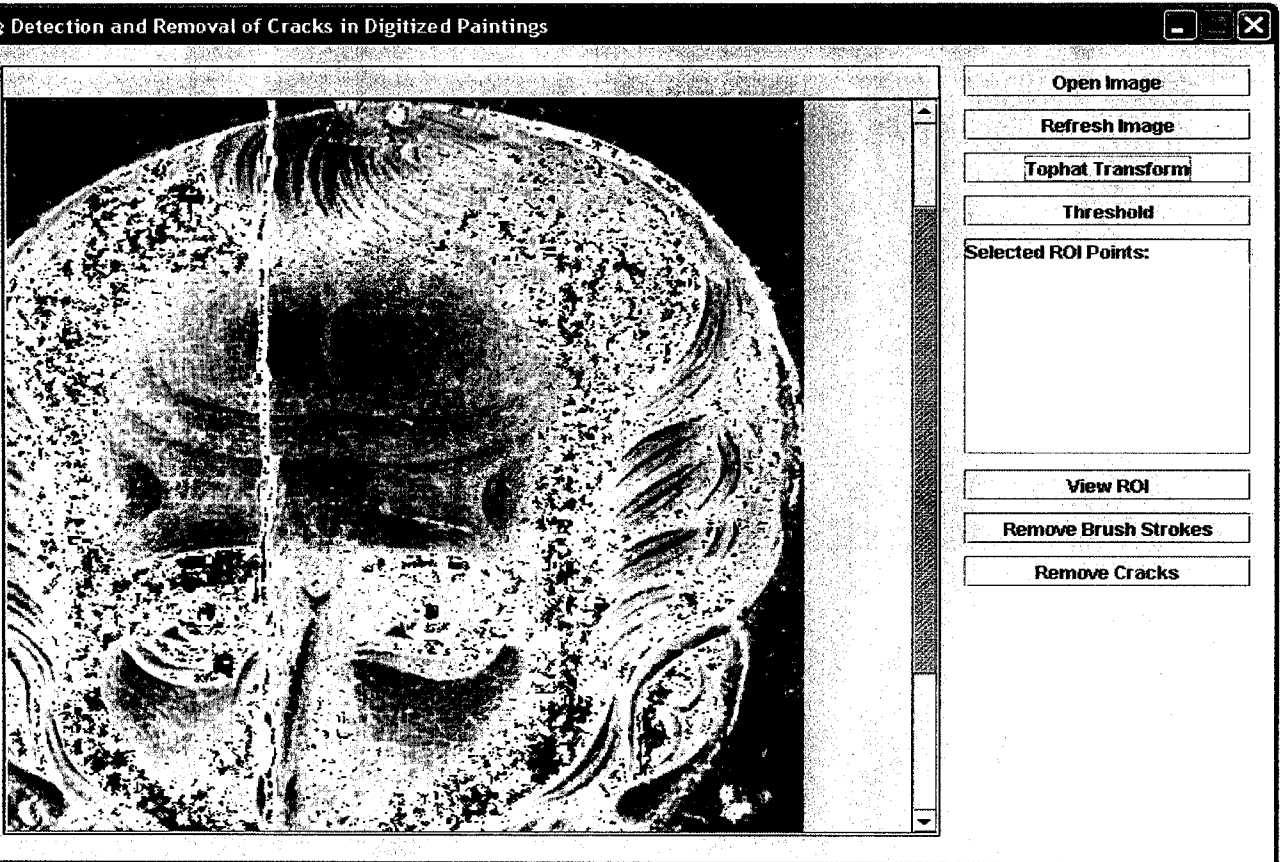


Figure A 1.3 shows the dialog box to execute Tophat transform

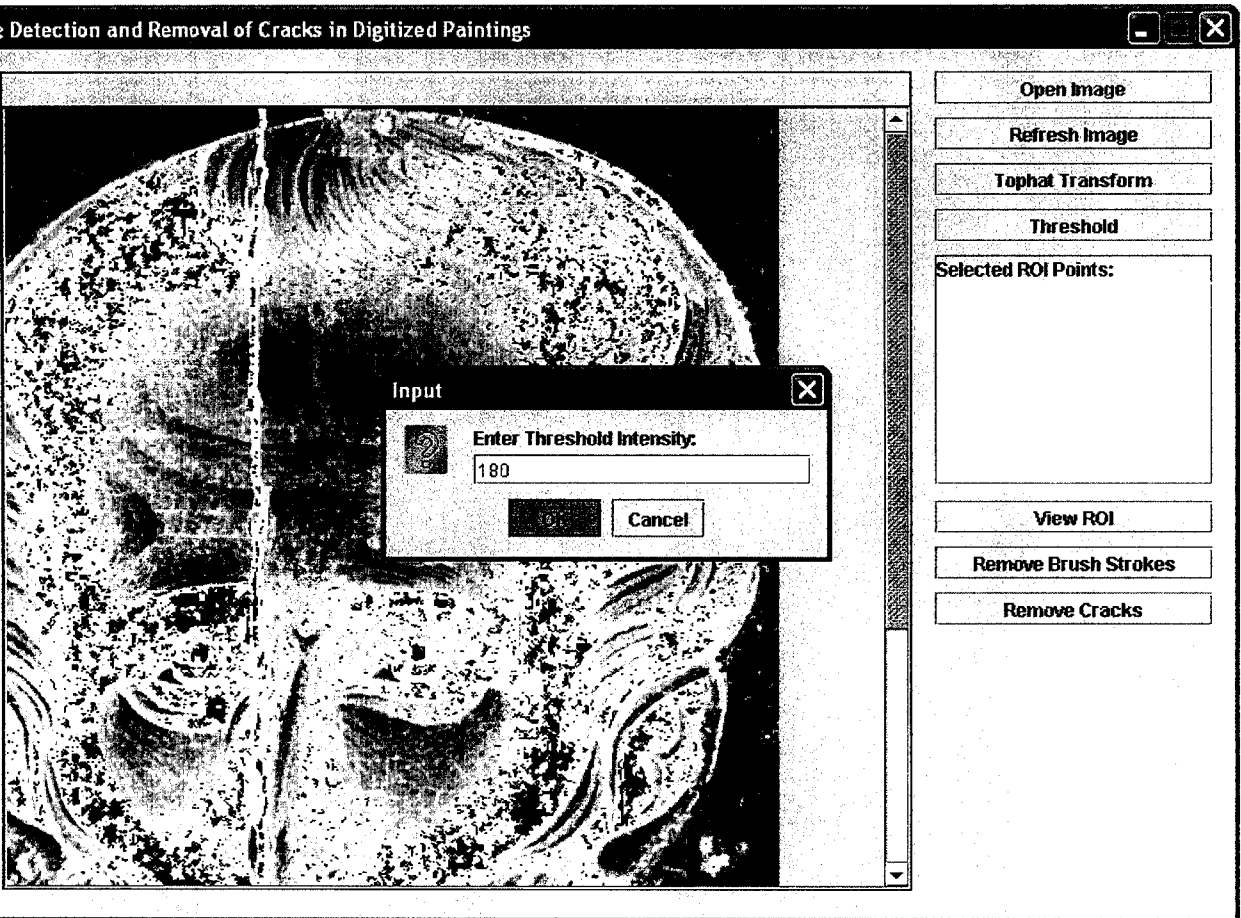


Figure A 1.4 shows the dialog box to enter the intensity value

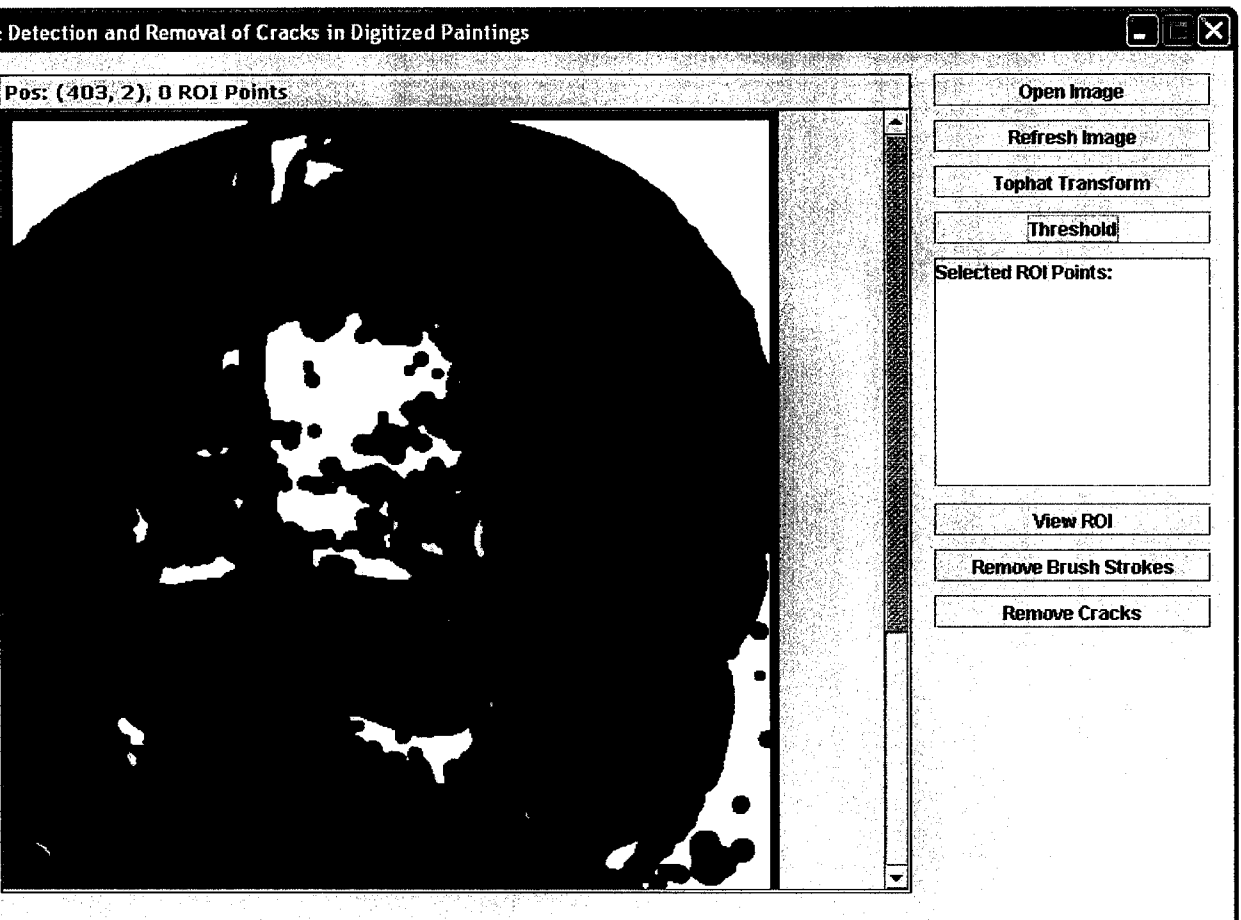


Figure A 1.5 shows the image status at intensity value 180

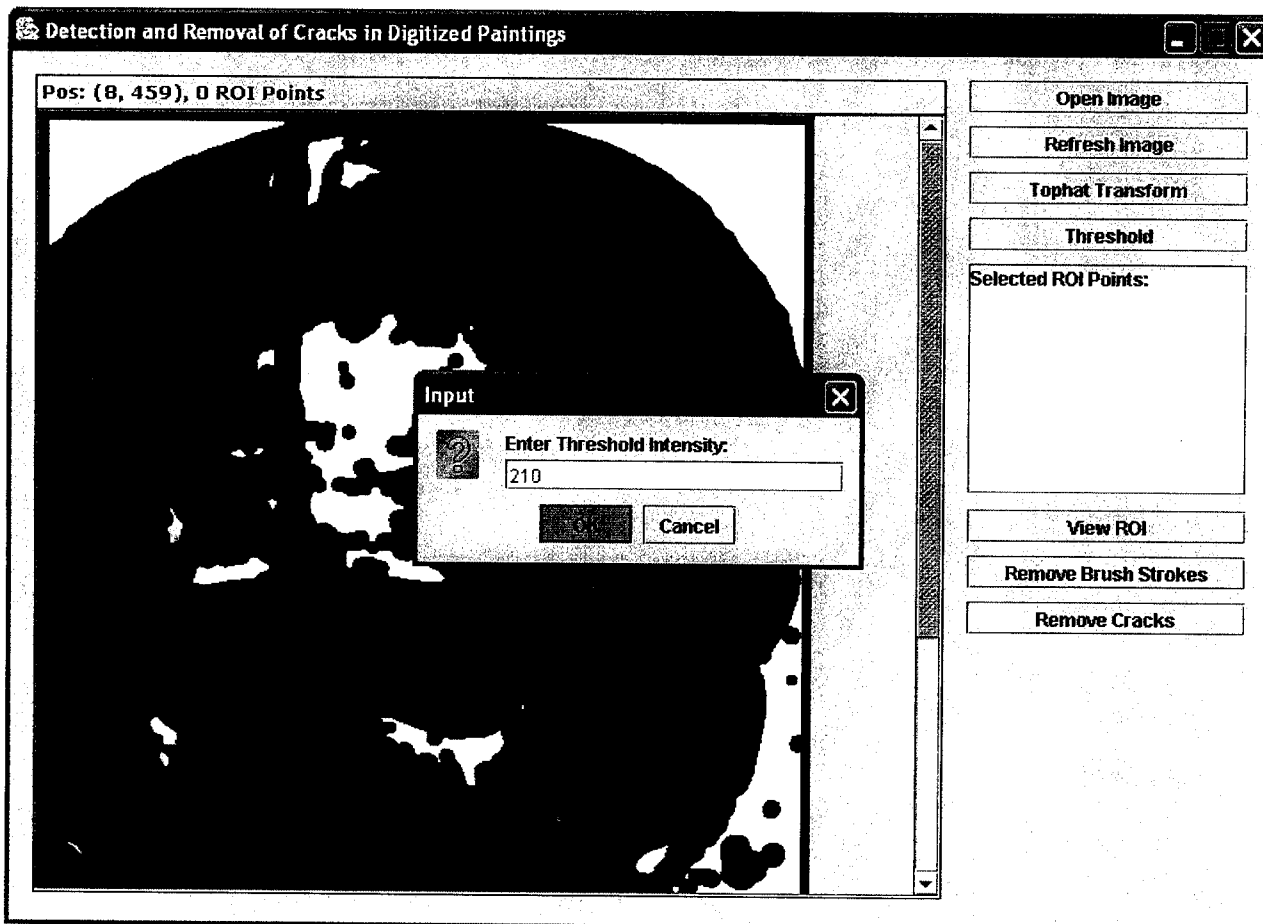


Figure A 1.6 shows the dialog box to enter the intensity value

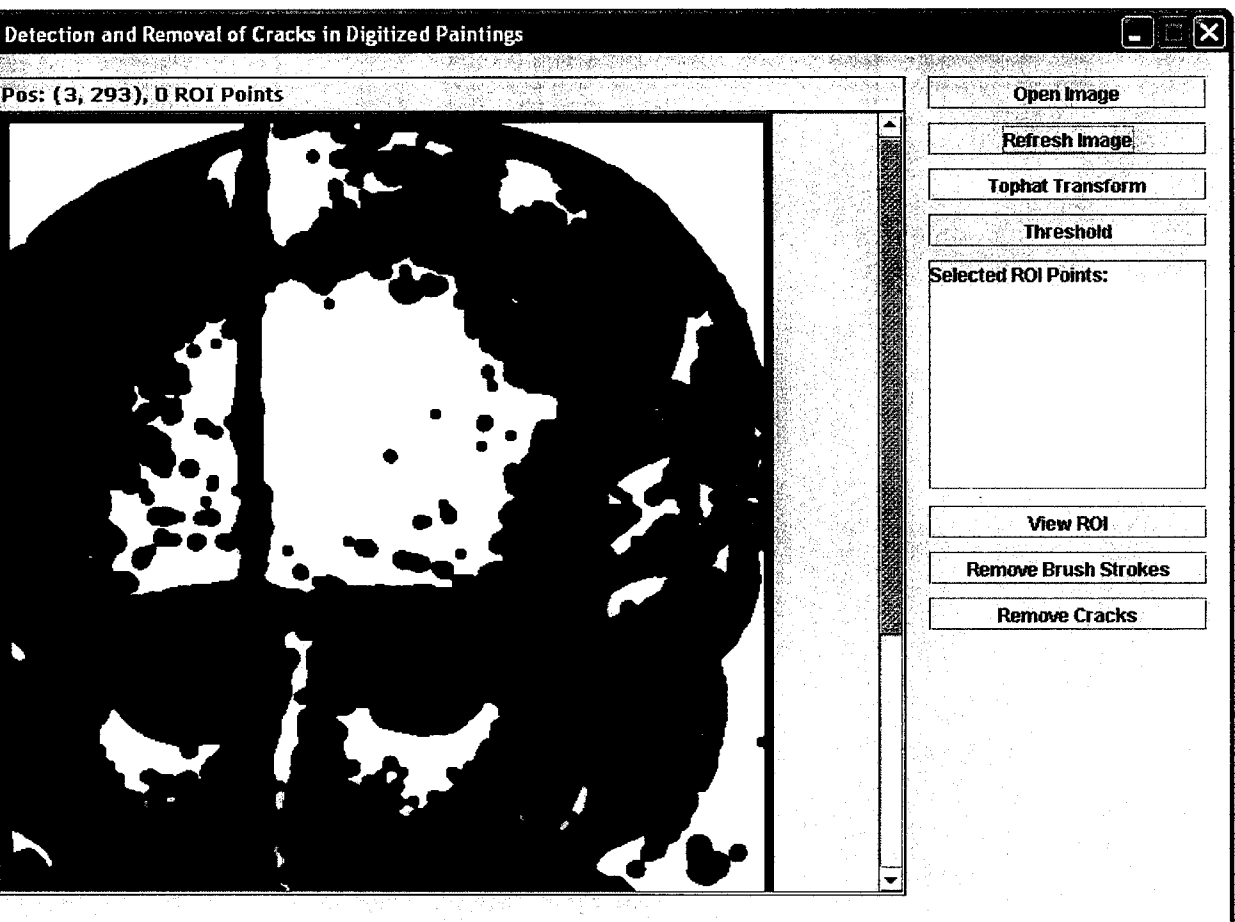


Figure A 1.7 shows the image status at intensity value 210

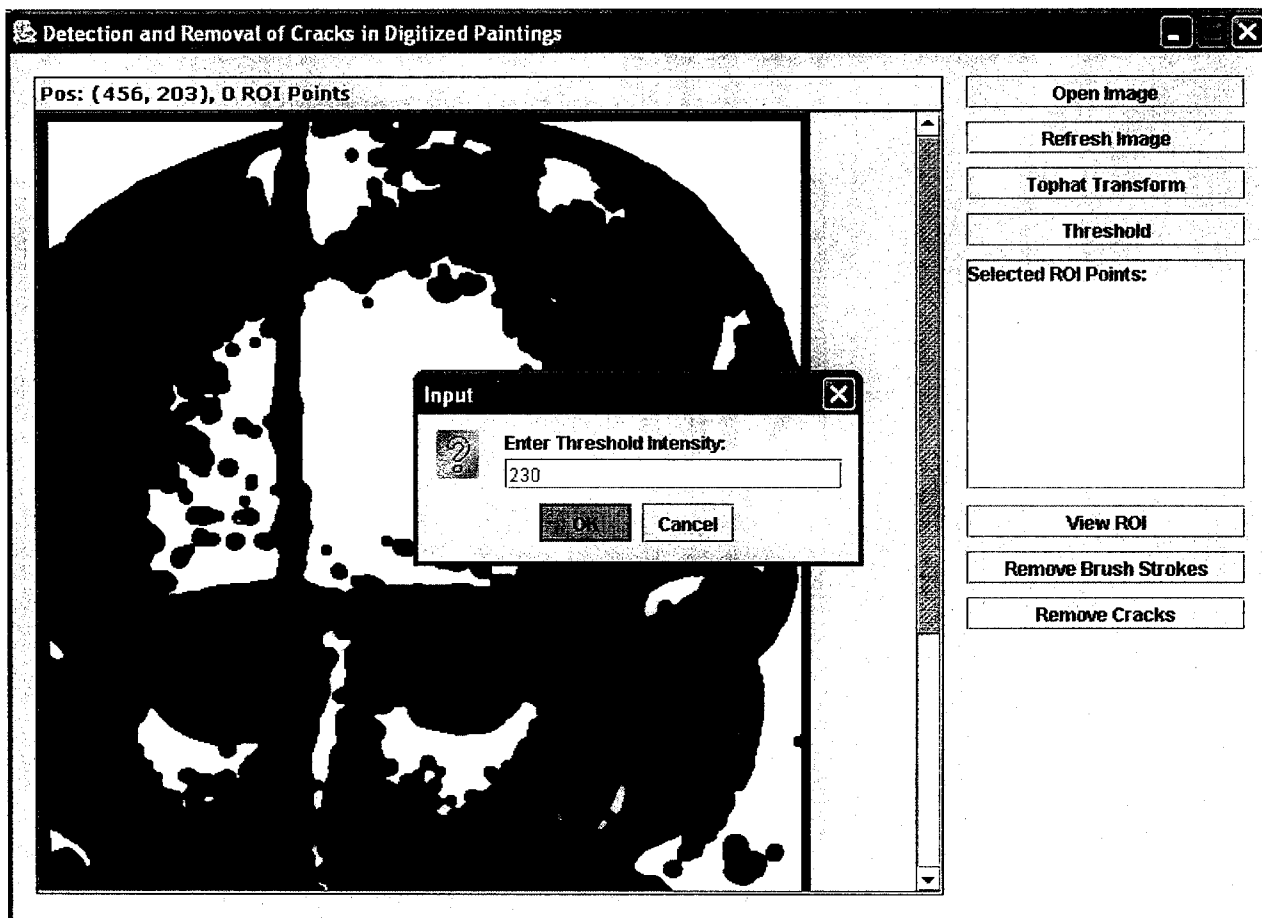


Figure A 1.8 shows the dialog box to enter the intensity value

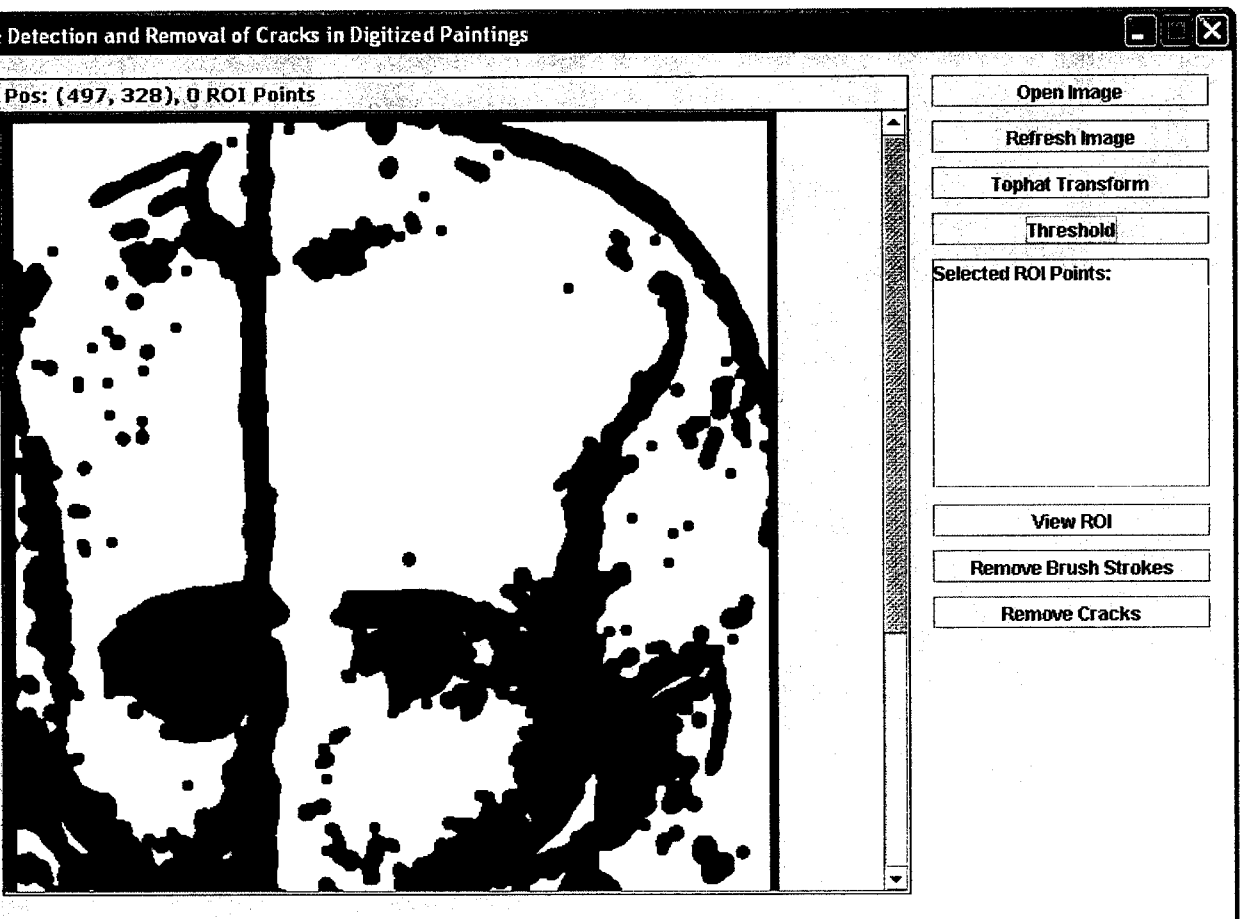


Figure A 1.9 shows the image status at intensity value 230

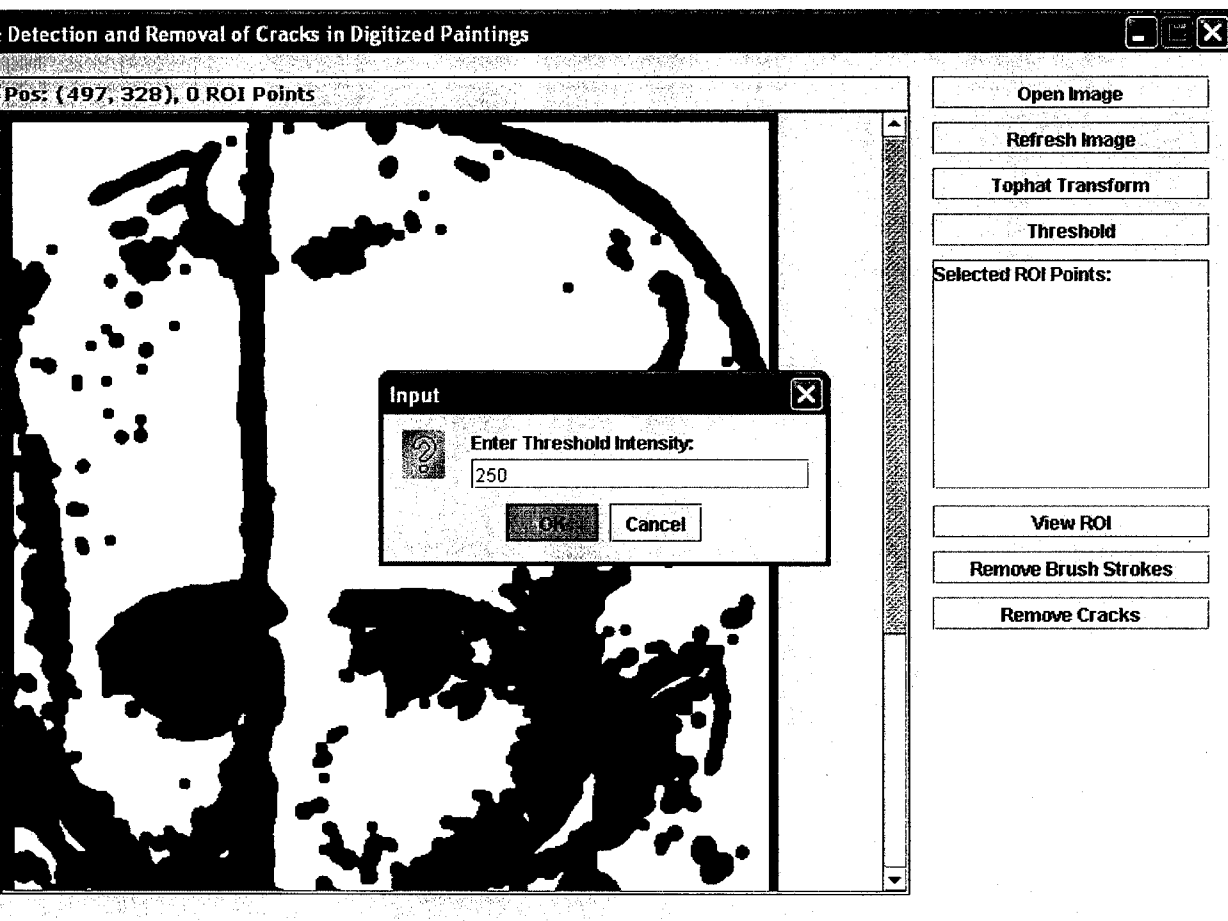


Figure A 1.10 shows the dialog box to enter the intensity value

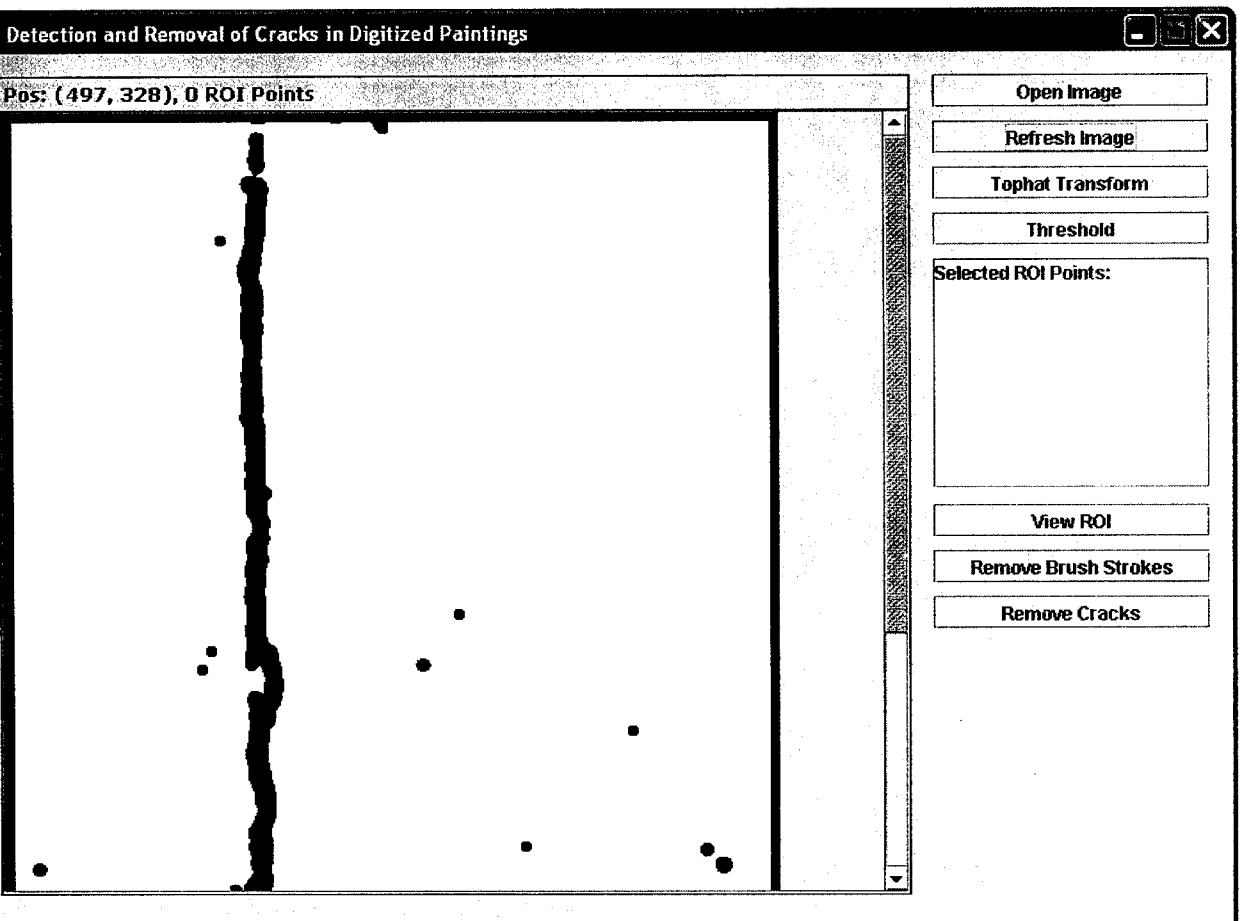


Figure A 1.11 shows the image status at intensity value 250

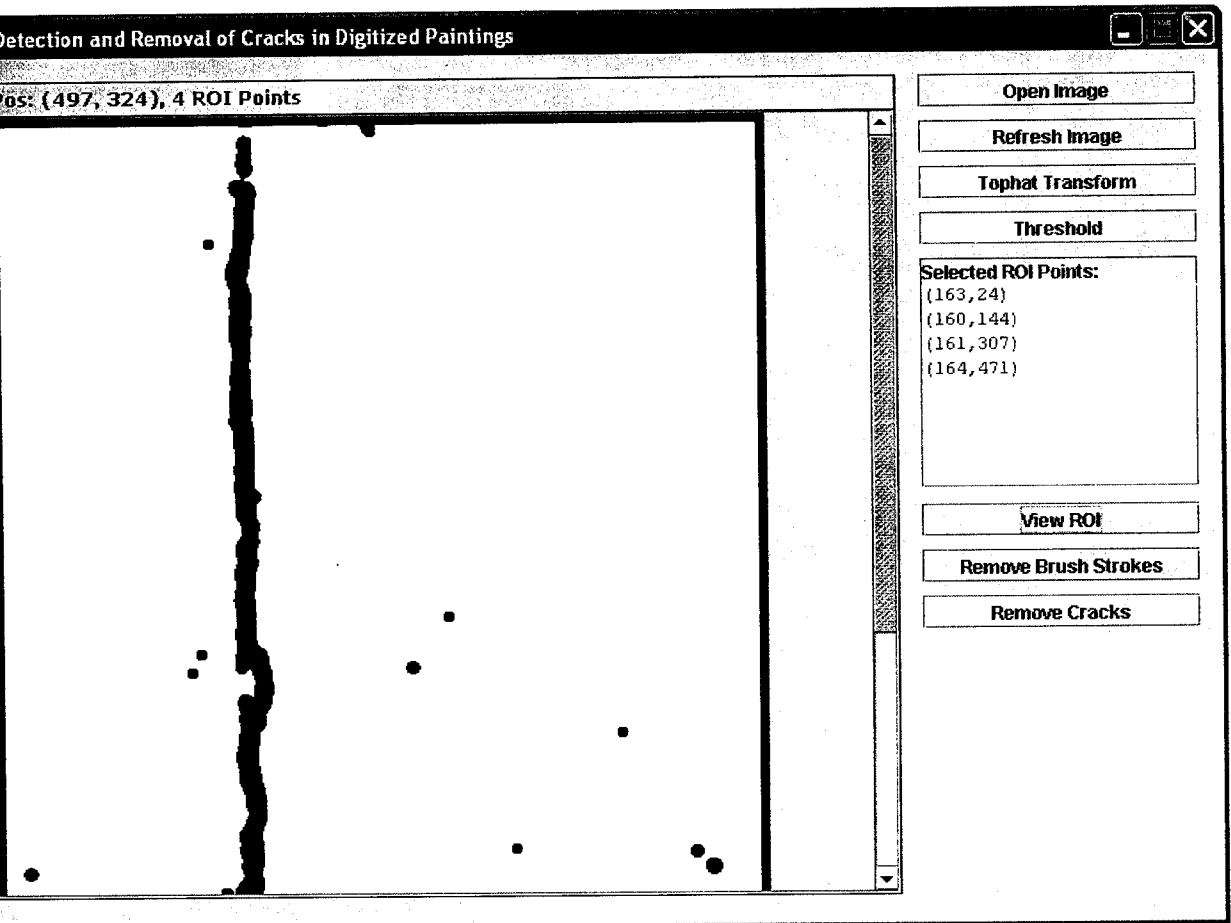


Figure A 1.12 shows the identified cracks and the brush strokes

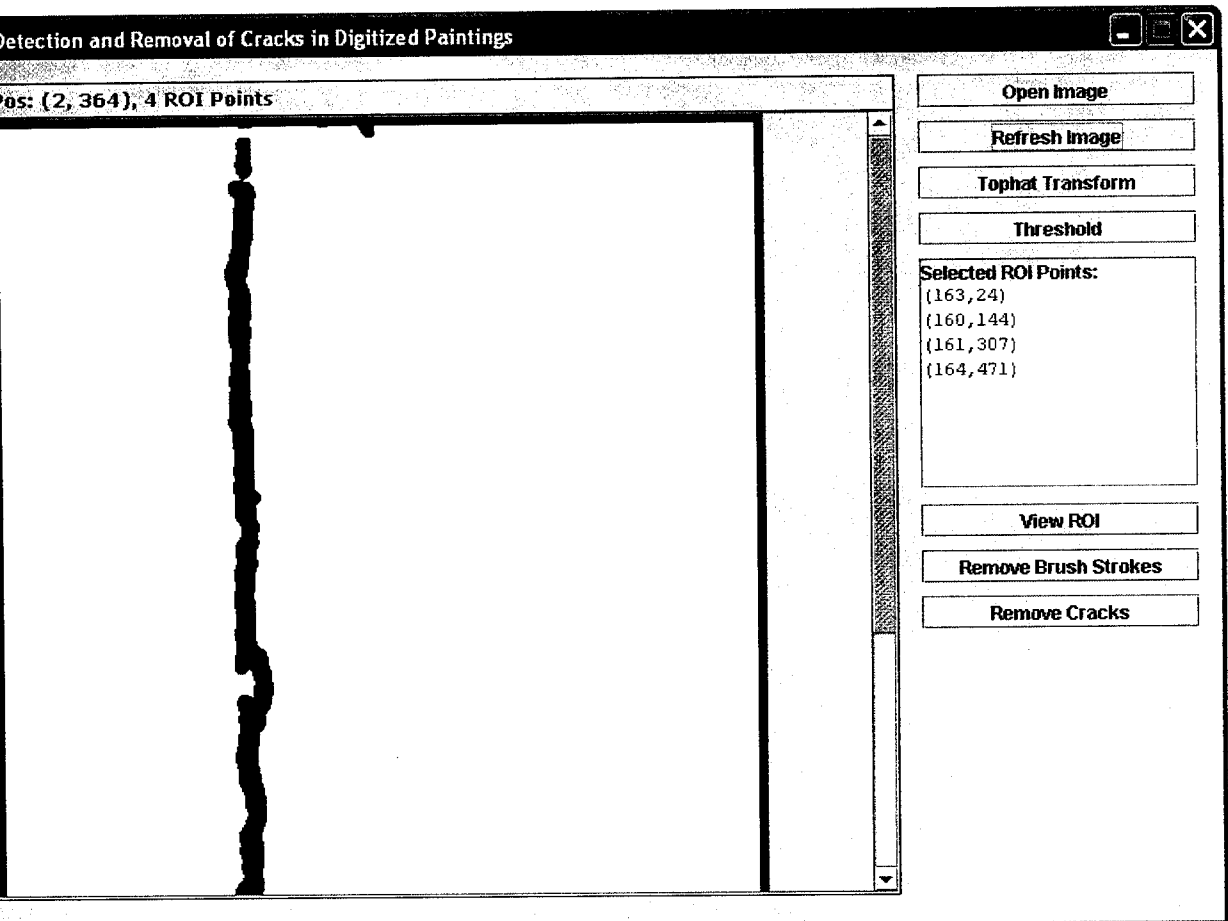


Figure A 1.13 shows the cracks of the image

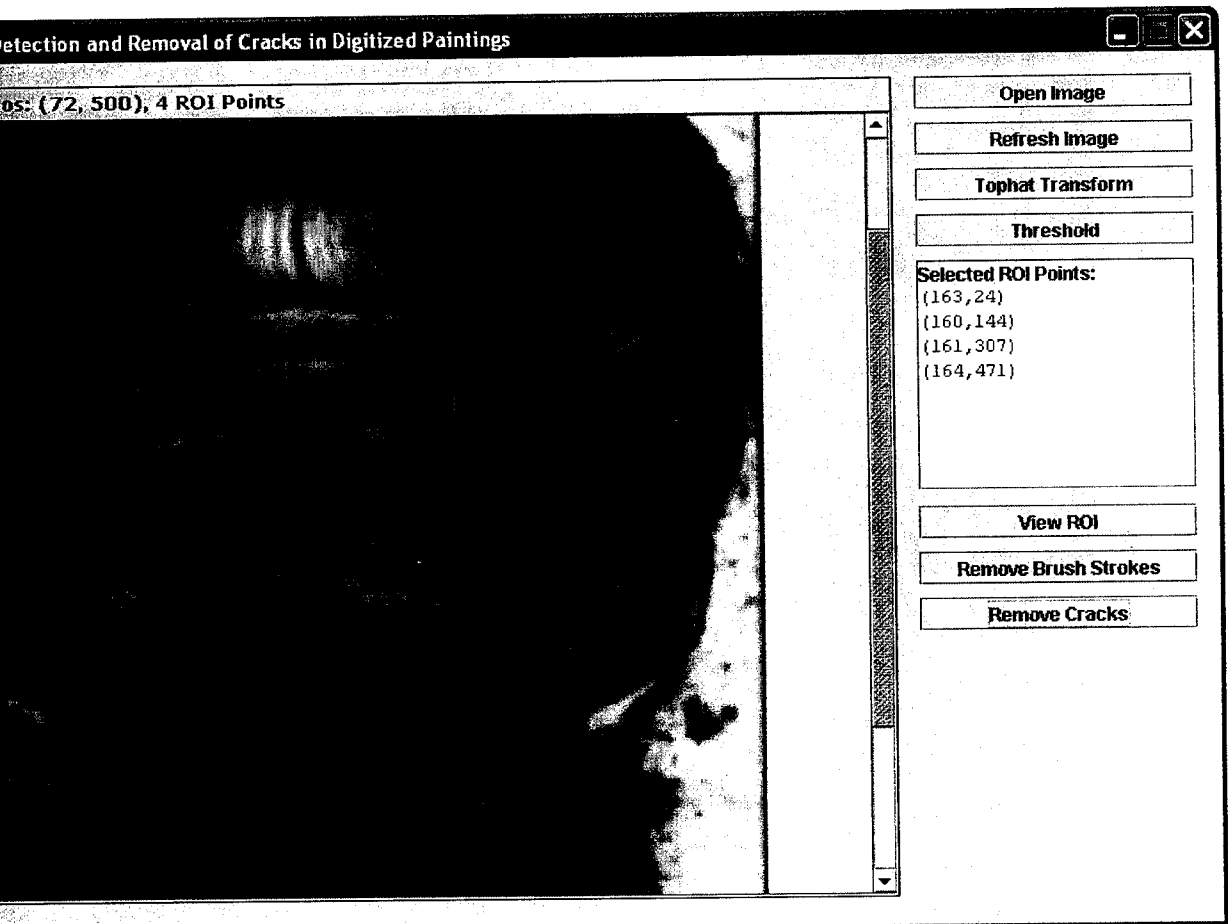


Figure A 1.4 shows the crack removed image

REFERENCES

BOOKS

- Herbert Schildt, 'The Complete Reference', Tata McGraw-Hill Publications, Second Edition, 2002.
- Joshua. Bloch, 'Effective Java Programming', Addison-Wesley Professional Publications, Third Edition, 2004.
- Roger S. Pressman, 'Software Engineering', Tata McGraw-Hill Publications, Sixth Edition, 2004.
- Patrick Naughton, 'Java Hand Book', Osborne McGraw-Hill Publications, Third Edition, 2001
- Mirdula Pariha, 'Programming Java', *Tata MacGraw-Hill publication, Third Edition 2002*

WEBSITES

- <http://www.sun.com/>
- <http://www.java.sys.com/>
- <http://www.jguru.com/>
- <http://www.java.sun.com>
- <http://www.javalobby.org>
- <http://www.javaboutique.com>