Development of a Mammographic Image Processing Environment Using MATLAB

THESIS

John L. Kelley
Captain, USAF

AFIT/GCS/ENG/94D-12

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/94D-12

Development of a Mammographic Image Processing Environment Using MATLAB

THESIS

John L. Kelley
Captain, USAF

AFIT/GCS/ENG/94D-12

AFIT/GCS/ENG/94D-12

Development of a Mammographic Image Processing Environment Using MATLAB

THESIS

John L. Kelley
Captain, USAF

AFIT/GCS/ENG/94D-12

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Preface

The goal of this research was to create a *Mammogram Image Processing Program* in a matrix-based environment that would support the future research and development of complex segmentation, feature extraction, and classification algorithms. McCall's *software quality factors* are used on a general level to assess the program's operational characteristics, ability to undergo change, and adaptability to new environments. An important criteria driving this research was to build an environment that would provide full source code accountability so that errors in earlier image processing stages could be fixed and not allowed to propagate through to later stages.

Few would argue that writing a thesis is a major undertaking, with success being dependent upon the dedication and discipline of the author; the sacrifices made by family members, along with the unwavering emotional support they offer; and the technical and personal support provided by the thesis advisor and the thesis committee. This would be the extent of the people associated with most thesis efforts, but this thesis was an exception, having benefited from the exceptional support provided by a physician, radiology technician, medical imaging systems service supervisor, system administrators, and a fellow graduate student.

Most of the credit for the successful completion of this thesis goes to my wife Yang S. Kelley, who took care of many of the day to day operations in our household; worked part time at a child day care development center; continued to strengthen her fellowship with God by attending Bible study twice a week and singing in the Church choir during Sunday Worship; took on many of the responsibilities of raising our four year-old son Jon L. Kelley; and gave birth to our daughter Mari V. Kelley during the late phases of this thesis. Thank you to my beloved son Jon L. Kelley, who made so many sacrifices that he is too young to even understand.

Dr. Steven K. Rogers, to thank him appropriately for the unending technical and personal support he has provided would take a chapter in itself. I have never met a teacher who was so dedicated to his work, concerned for the welfare of his students, and had remarkable interrelationship skills and patience with students. I sincerely believe it was God's grace that provided me the opportunity to have Dr. Steven K. Rogers as my advisor. To Capt Dennis W. Ruck and Dr. Mark E. Oxley, thank you for the enthusiastic willingness to be a part of my thesis committee even though I made my request well into the third quarter when the both of you were already members of a number of other committees.

Capt Jeffrey W. Hoffmeister is a Flight Surgeon who has recently earned a Master of Science Degree in Electrical Engineering. Even though Capt Jeffrey W. Hoffmeister has one of the most busiest schedules imaginable, he spent countless hours patiently answering my questions and providing technical support when needed. I feel extremely fortunate to have had him as a sponsor and sincerely hope that one day we will have the opportunity to work together on a research project.

Shireen Braner is a Radiology Technologist and the Mammography Coordinator for Grandview Hospital and Medical Center and for Southview Hospital and Family Health Center. She was more than enthusiastic to support this research effort, teaching us the principle of the Lorad Digital Spot Mammogram, and allowing us access to the mammogram database and the corresponding pathology reports. A number of visits, requiring her presence, were made to Southview Hospital and Family Health Center to download the mammogram digital images onto floppy diskettes. Shireen Braner was always willing to work around our schedule, even though I felt guilty and wanted to work around hers. Thank you Shireen Braner for your generosity.

Joel R. Arnold is a Medical Imaging Systems Service Supervisor working for Baldwin who taught us the technical aspects of the Lorad Digital Spot Mammogram and how to download the mammogram digital images onto floppy diskettes. Thank you for the personal time you donated in support of this research effort.

One of my weak areas, that I hate to disclose considering that I am a computer science major, is in system administration. The Air Force Institute of Technology has two of the best system administrators in the business in Dan A. Zambon and Dave Doak. Without their invaluable assistance, I would have wasted countless unproductive hours sitting in front of a computer terminal trying to resolve problems that are abstract to me but trivial for them. Only if private industry knew of these two ace system administrators, then . . . .

And finally I would like to thank Capt Curtis E. Martin, an Electrical Engineering Ph.D. candidate majoring in Pattern Recognition, for his invaluable technical support on MATLAB and constructive comments as a simulated user of the *Mammogram Image Processing Program.*

John L. Kelley

Table of Contents

Page

Preface....ii

Table of Contents....v

List of Figures....xi

Abstract....xiii

I. Introduction....1-1

1.1 Background....1-1

1.2 Definitions....1-4

1.3 Problem....1-6

1.4 Scope....1-7

1.5 Sequence of Presentation....1-8

II. Literature Review....2-1

2.1 Introduction....2-1

2.2 Screen-Film X-Ray Mammography: Principle, Signs of Breast Cancer, and Shortcomings....2-2

2.3 Fine Needle Aspiration Biopsy and Surgical Biopsy: Limitations....2-3

2.4 Stereotaxic Core Biopsy of Breast Lesions: Methodology and Advantages....2-4

2.5 Digital Mammography: Principle and Advantages....2-10

2.6 Computer-Aided Diagnosis: Definition and Purpose....2-11

2.7 Computer-Aided Diagnosis in Mammography: Purpose....2-13

2.8 Mammographic Computer Vision Environments....2-13

2.9 Conclusion....2-14

III. Requirements and Design..3-1

3.1 Introduction..3-1

3.2 *Mammgram Image Processing Program* Requirements.............3-2

3.2.1 Essential Utilities................................3-2

3.2.2 Optional Utilities.................................3-4

3.2.3 Function Templates.................................3-5

3.3 *Mammogram Image Processing Program* Design..................3-6

3.4 McCall's *Software Quality Factors*.........................3-12

3.4.1 A Closer Examination...............................3-12

3.4.2 Intended Use.......................................3-14

3.5 Summary...3-15

IV. Implementation, Functionality, and Operation....................4-1

4.1 Preliminaries...4-1

4.2 Database..4-2

4.3 Delete Image..4-9

4.4 Display Utilities.....................................4-11

4.4.1 Zoom..4-13

4.4.2 Label Image.....................................4-15

4.4.3 Reference Grid..................................4-16

4.4.4 Histogram.......................................4-22

4.4.5 Intensity Profile...............................4-24

4.4.6 Density Slice...................................4-27

4.4.7 Texture Map.....................................4-31

4.4.8 Delete Image....................................4-34

4.5 Image Processing......................................4-34

4.5.1 Region of Interest..............................4-35

4.5.2 Edge Extraction.................................4-41

4.5.3 Delete Image....................................4-44

4.6 Image Analysis..4-45

4.6.1 Load Region of Interest File.....................4-46

4.6.2 Templates.......................................4-48

4.6.3 Print to File....................................4-49

4.7 Quit Program..4-51

4.8 Summary...4-51

V. Conclusion..5-1

5.1 Assessment..5-1

5.1.1 Maintainability...5-1

5.1.2 Flexibility..5-2

5.1.3 Testability..5-2

5.1.4 Portability..5-2

5.1.5 Reuseability...5-3

5.1.6 Interoperability...5-4

5.1.7 Correctness..5-5

5.1.8 Reliability..5-5

5.1.9 Efficiency...5-5

5.1.10 Integrity..5-5

5.1.11 Usability..5-6

5.2 Final Comments..5-6

Appendix A *Mammogram Image Processing Program* Source Code..........A-1

A.1 image_processing.m..........A-1

A.2 database_menu.m..........A-4

A.3 load_image.m..........A-6

A.4 delete_image.m..........A-10

A.5 du_menu.m..........A-12

A.6 reload_image.m..........A-15

A.7 zoom.m..........A-18

A.8 label_image.m..........A-20

A.9 reference_grid.m..........A-22

A.10 histogram.m..........A-30

A.11 intensity_profile.m..........A-32

A.12 density_slice.m..........A-36

A.13 texture_map.m..........A-40

A.14 ip_menu.m..........A-43

A.15 region_of_interest.m..........A-45

A.16 edge_extraction.m..........A-50

A.17 ia_menu.m..........A-54

A.18 feature_extraction.m..........A-56

A.19 load_roi_file.m..........A-59

A.20 template1.m..........A-61

A.21 template2.m..........A-62

A.22 template3.m..........A-63

A.23 template4.m..........A-64

A.24 print_to_file.m..........A-65

A.25 segmentation.m..........A-69

A.26 classification.m..........A-70

Appendix B C/C++ Interface Technical Notes..........................B-1

B.1 C Mex-File Function Example (mxeng3.txt)..................B-1

B.2 C Mex-File Function Example (mxeng4.txt)..................B-3

B.3 How to Write a C++ MEX-File Function (mxeng5.txt)..........B-6

Appendix C Lorad Digital Spot Mammogram Specifications..............C-1

Appendix D Annotated Readings......................................D-1

Bibliography...BIB-1

Vita...VITA-1

List of Figures

Figure Page

1.1 Image Processing Stages..1-3

1.2 McCall's *Software Quality Factors*...............................1-7

2.1 Scout Stereoviews..2-6

2.2 Schematic of the Two-Part Cutting Needle........................2-7

2.3 Prefire Stereoviews..2-9

3.1 *Mammogram Image Processing Program* Overall Diagram..............3-6

3.2 Image Processing Main Menu Diagram...............................3-7

3.3 Database Menu Diagram..3-8

3.4 Display Utilities Menu Diagram..................................3-9

3.5 Image Processing Menu Diagram...................................3-10

3.6 Image Analysis Menu Diagram....................................3-11

3.7 Feature Extraction Submenu Diagram..............................3-11

3.8 McCall's *Software Quality Factors*...............................3-13

3.9 Criteria that Define the Correctness Quality Factor...............3-14

4.1 *Mammogram Image Processing* Program Main Menu...................4-2

4.2 Database Menu..4-3

4.3 Numeric Precision Options Menu..................................4-4

4.4 Variable image read record 4-5

4.5 Variables images read catalogue and offset 4-7

4.6 Needle Placement Stereoviews....................................4-9

4.7 Display Utilities Menu..4-13

4.8 Zoom Submenu...4-14

4.9 Label Image Submenu..4-15

4.10 Labelled Image...4-16

4.11 Reference Grid Submenu..4-17
4.12 Color Submenu...4-19
4.13 Line Style Submenu..4-20
4.14 Reference Grid..4-22
4.15 Histogram Submenu...4-23
4.16 Histogram...4-24
4.17 Intensity Profile Submenu.......................................4-25
4.18 Intensity Profile 4-27
4.19 Density Slice Submenu...4-28
4.20 Color Map Options Menu..4-29
4.21 Jet Color Map Demonstration.....................................4-30
4.22 Density Slice...4-31
4.23 Texture Map Submenu...4-32
4.24 Interpolation Options Submenu...................................4-32
4.25 Texture Map...4-33
4.26 Image Processing Menu...4-34
4.27 Region of Interest Submenu......................................4-35
4.28 Region of Interest File dn1swb.roi1.............................4-40
4.29 Coordinate File dn1swb.coordinate1..............................4-40
4.30 Edge Extraction Submenu 4-41
4.31 Directionality Factor Options Menu..............................4-42
4.32 Prewitt Nondirectional Edge Extraction..........................4-43
4.33 Prewitt Complemented Edge Extraction............................4-44
4.34 Image Analysis Menu...4-45
4.35 Feature Extraction Submenu......................................4-46
4.36 Print to File Submenu...4-49
4.37 LNKnet File...4-50

Abstract

Breast cancer is a disease that accounts for a disturbingly large number of deaths in females each year. Its prevalence is a topic of concern to all of us since it can affect our families, friends, and coworkers. Although mammographic screening is the most effective method currently available for the early detection of breast cancer, it is far from being an infallible procedure. Mammographic reading is error prone, partly because of the complexity of the task and partly because of the variability in human performance. Computers offer high reproducibility, and when used as an adjunct by the radiologist, may improve diagnostic accuracy and thus the mammographic screening process.

The goal of this research was to create using MATLAB version 4.2 (UNIX) and the MATLAB Image Processing Toolbox a menu-based, mouse-driven, and keyboard interactive *Mammogram Image Processing Program* that would support the future research and development of complex segmentation, feature extraction, and classification algorithms. McCall's *software quality factors* are used on a general level to assess the program's operational characteristics, ability to undergo change, and adaptability to new environments. An important criteria driving this research was to build a platform that would provide full source code accountability so that errors in earlier image processing stages could be fixed and not allowed to propagate through to later stages.

Development of a Mammographic Image Processing Environment Using MATLAB

I. Introduction

1.1 *Background*

Breast cancer is responsible for approximately 20% of all female deaths due to cancer [Gale and others, 1993]. Evidence exists that mammographic screening can reduce the mortality from breast cancer by 20 to 40% [Doi and others, 1993a]. Despite the training, practice, and experience of radiologists [Astley and others, 1993], and the fact that mammographic signs of early cancer are fairly well documented [Gale and others, 1993], mammographic interpretation is still an extremely demanding task, complicated by the diverse and often complex nature of normal and abnormal appearances, as well as the small and subtle characteristics of clinically significant lesions [Astley and others, 1993].

Disturbingly, radiological observer performance is error prone [Astley and others, 1993]. That is, 10 to 30% of women who have breast cancer and undergo mammography have negative mammograms [Doi and others, 1993a] [Giger, 1993a] [Giger, 1993b] [Giger and Vyborny, 1993]. Only 10 to 20% of the cases referred for surgical biopsy as a result of suspicious findings prove to be malignant [Giger, 1993b] [Giger and Vyborny, 1993]. Significant levels of intra- and inter-observer variability exists in mammography [Astley and others, 1993], and because of this inconsistency in human performance, it is unlikely that observer errors will ever be eliminated [Gale and others, 1993]. Despite the somewhat gloomy picture that has been painted here, computers offer highly reproducible performance [Gale and others, 1993] and when used as an adjunct by the radiologist, may improve the accuracy of interpretation.

An ideal computer vision system is one that could automatically detect and classify all forms of abnormality [Astley and others, 1993]. This system as described by [Astley and others, 1993] would contain the following features:

- Performance monitoring
- Image acquisition from digitized film images and from digital radiography systems
- Data compression
- Image display and manipulation
- Image restoration
- Image enhancement
- Pre-screening
- Prompting
- Analysis and classification
- On-line assistance
- Reporting
- Teaching and research

The design and development of an ideal computer vision system for mammographic image analysis will not be achieved for some time, and should be considered a long term goal [Astley and others, 1993]. The development of an ideal system is hindered by the unavailability of reliable methods for detecting all abnormal mammographic signs, and unanswered questions in regards to efficient digitization and the management of a substantial number of x-rays and massive digital data [Astley and others, 1993]. [Astley and others, 1993] mention that a more reasonable short term goal is to develop a system that aids the radiologist by providing such features such as image restoration; image manipulation; display tools; and the automatic detection, prompting, and analysis of a limited range of specific lesions. As applied in mammography, computer based approaches have been constrained to well-defined tasks such as the detection of microcalcifications, masses, and asymmetry [Gale and others, 1993]. As Figure 1.1 shows, the detection and identification process consists of three image processing stages: segmentation -- locating regions of interest within an image, feature extraction -- obtaining characteristic data for these regions of interest, and classification -- processing of the characteristic data to obtain a diagnosis [Rogers and others, 1994].

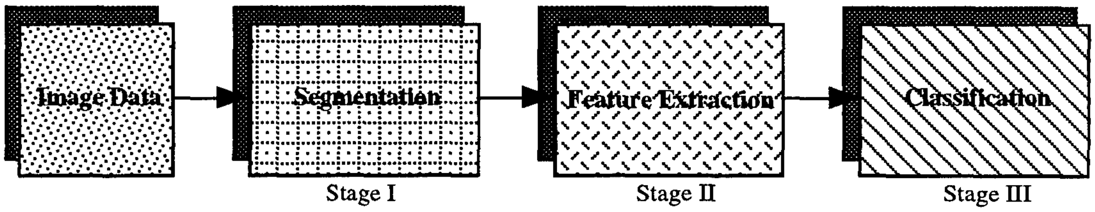


Figure 1.1 - Image processing stages

A valid question likely to be asked is "Why build an image processing environment when good ones already exist?" The answer is accountability. A user usually has no interest or need to be able to examine source code in detail. A developer on the other hand, often needs access to the source code, especially when the reused detection or analysis routine is complex and part of the preprocessing or postprocessing steps to the algorithm currently under development. When using commercially developed image processing programs, readily access to the source code does not exist. Even when access to the source code is available as in public domain routines, there is a lack of or no corresponding documentation. Thus the developer must carefully step through the routine in the hopes of being able to understand it and confirm its accuracy or trust it with blind faith. This becomes a significant issue considering the increasing complexity of the detection and analysis algorithms that are constantly being developed.

Once an environment is developed, it should be assessed using a quality standard as to its suitability for the future research and development of image processing algorithms. This is a crucial step and can be likened to ensuring a foundation is solid prior to building the frame. If the foundation is weak, then it either needs to be strengthened or a new one should be created. As image processing routines are incrementally added to the environment, they also need to be assessed but more so in terms of their value. As [Astley and others, 1993] state: ". . . it is unlikely that any system which cannot either perform at least as well as an expert radiologist, improve a less experienced observer's performance to that of an expert radiologist, or improve an expert radiologist's performance, will find clinical acceptance."

1.2 *Definitions*

The medical terminology used throughout this thesis are defined below as a convenience to the reader:

- ➠ Benign: "Not malignant; not recurrent; favorable for recovery." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Biopsy: "The removal and examination, usually microscopic, of tissue from the living body, performed to establish a precise diagnosis." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Carcinoma: "A malignant new growth made up of epithelial cells tending to infiltrate the surrounding tissue and give rise to metastases." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Circumscribed: "Bounded or limited; confined to a limited space." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Computer-aided diagnosis: "A more sophisticated and technologically challenging form . . . in which the image is directly analyzed by computer with a view to detecting, localizing, and quantitating abnormalities in the radiograph." [Shtern, 1992]
- ➠ Craniocaudal: "Direction from head to foot." [Thomas, 1989]
- ➠ False-negative: "An individual whose test results exclude him from a particular diagnostic group though he may truly belong to such a group." [Williams & Wilkins, 1990]

- False-positive: "An individual whose test results include him in a particular diagnostic group though he may not truly belong to such a group." [Williams & Wilkins, 1990]
- Histology: "That department of anatomy which deals with the minute structure, composition, and function of the tissues; called also microscopial anatomy." [W. B. Saunders Dictionary Staff, 1994]
- In situ: "In the natural or normal place; confined to the site of origin without invasion of neighboring tissues." [W. B. Saunders Dictionary Staff, 1994]
- Invasive: "Denoting or characterized by invasion." [Williams & Wilkins, 1990]
- Invasion: "Local spread of a malignant neoplasm by infiltration or destruction of adjacent tissue." [Williams & Wilkins, 1990]
- Malignancy: "A cancer, especially one with the potential to cause death." [W. B. Saunders Dictionary Staff, 1994]
- Malignant: "Tending to become progressively worse and to result in death." [W. B. Saunders Dictionary Staff, 1994]
- Margin: "An edge or border, such as the boundary of an organ or other anatomic structure." [W. B. Saunders Dictionary Staff, 1994]
- Mediolateral: "Concerning the middle and side of a structure." [Thomas, 1989]
- Metastases: "Plural of metastasis." [Williams & Wilkins, 1990]
- Metastasis: "The spread of a disease process from one part of the body to another, as in the appearance of neoplasms in parts of the body remote from the site of the primary tumor; results from the dissemination of tumor cells by the lymphatics or blood vessels, or by direct extension through serous cavities or subarachnoid or other spaces." [Williams & Wilkins, 1990]
- Necrosis: "Pathologic death of one or more cells, or of a portion of tissue or organ, resulting from irreversible damage." [Williams & Wilkins, 1990]

- ➠ Neoplasm: "An abnormal tissue that grows by cellular proliferation more rapidly than normal and continues to grow after the stimuli that initiated the new growth ceases." [Williams & Wilkins, 1990]
- ➠ Oblique: "Slanting, diagonal." [Thomas, 1989]
- ➠ Palpable: "Perceptible to touch; capable of being palpated." [Williams & Wilkins, 1990]
- ➠ Palpate: "To examine by feeling and pressing with the palms of the hands and fingers." [Williams & Wilkins, 1990]
- ➠ Parenchyma: "The distinguishing or specific cells of a gland or organ, contained in and supported by the connective tissue, or stroma." [Williams & Wilkins, 1990]
- ➠ Parenchymal: "Pertaining to or of the nature of the parenchyma." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Pathology: "That branch of medicine which treats of the essential nature of disease, especially of the structural and functional changes in tissues and organs of the body which cause or are caused by disease." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Spicule: "Sharp needle-like body." [W. B. Saunders Dictionary Staff, 1994]
- ➠ Subcutaneous: "Beneath the skin." [W. B. Saunders Dictionary Staff, 1994]

1.3 *Problem*

This research effort focuses on using the matrix-based environment of MATLAB to create a *Mammogram Image Processing Program* that will support the future research and development of complex segmentation, feature extraction, and classification algorithms. McCall's *software quality factors* shown in Figure 1.2 will be used on a general level to assess the program's operational characteristics, ability to undergo change, and adaptability to new environments.

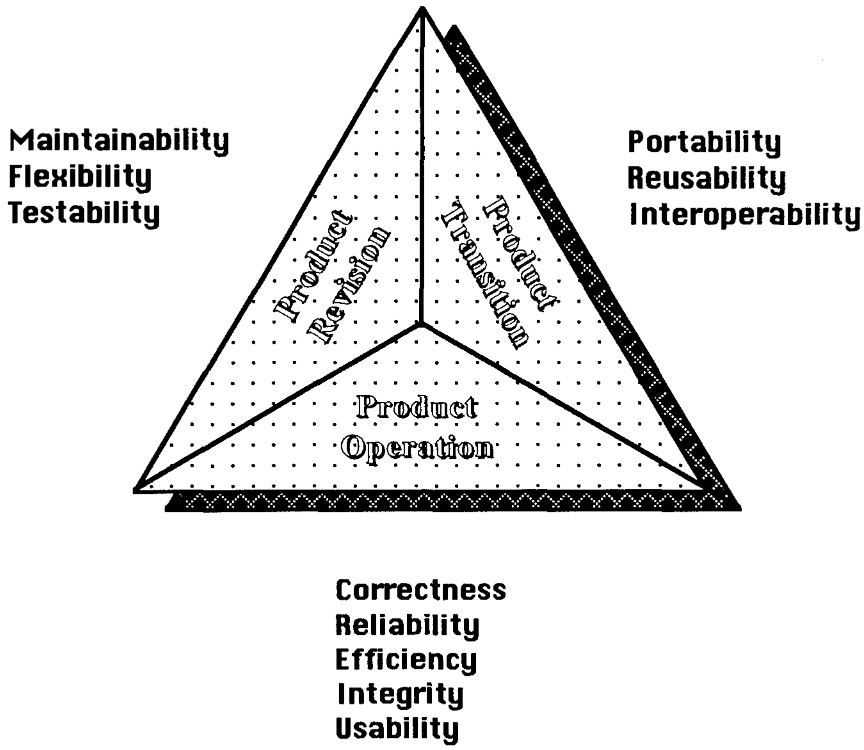


Figure 1.2 - McCall's *software quality factors* [Pressman, 1987]

1.4 *Scope*

This thesis effort will be constrained to using MATLAB version 4.2 (UNIX) and the MATLAB Image Processing Toolbox to develop a menu-based, mouse-driven and keyboard interactive *Mammogram Image Processing Program* that includes the following routines:

- Load digital mammographic images
- Delete digital mammographic images
- Display utilities
  - Zoom
  - Label image
  - Reference grid

- Histogram
- Intensity profile
- Density slice and color map demonstration
- Texture map
- Image processing
  - Roberts, Prewitt, Sobel, and Marr-Hildreth edge extraction
  - Region of interest
- Image analysis
  - Feature extraction test stubs
  - Print results in a LNKnet compatible format

LNKnet is a software package that includes more than twenty pattern classification, clustering, and feature extraction algorithms [Lippmann and others, 1993]. The reader interested in the use of LNKnet as it relates to the *Mammogram Image Processing Program* should refer to the thesis "Computer-Aided Breast Cancer Diagnosis" written by Capt Kate M. Kocur.

1.5 *Sequence of Presentation*

The remainder of this thesis is organized into four chapters. Chapter II provides a state of the art examination of mammography drawn from literature dated 1990 to current. Chapter III presents the requirements and design of the program. Chapter IV describes implementation aspects and overviews the functionality and operation of the program. The majority of the routines listed in section 1.4 include options and the program itself contains a number of error control features. Chapter V contains an assessment of the program described herein, using McCall's quality model. Four appendices are also included that may be of interest to the developer. Appendix A is a complete, commented source code listing of the program written in the matrix-based environment of MATLAB. Appendix B contains technical notes on how to interface MATLAB to routines written in the higher order languages of C or C++. Appendix C lists the specifications for the Lorad Digital Spot Mammogram. The digital mammograms from this stereotaxic core biopsy system were used in the coding and test phases of the program. Appendix D is an annotated listing of sixty papers, books, and software manuals read in the preparation of this thesis.

II. Literature Review

This chapter presents an introduction to the field of mammography, focusing on the technology currently being used and developed to reduce the subjectivity in this imperfect science. It describes screen-film x-ray mammography (conventional mammography), in terms of its diagnostic use, limitations, and associated biopsy techniques. The recent introduction of stereotaxic core biopsy of breast lesions, employing primary digital acquisition, is examined. Primary digital acquisition and film digitization, the two methods used to obtain digital mammographic images, are discussed. The field of computer-aided diagnosis uses these digital mammographic images to develop computer-based diagnostic and detection algorithms with the goal of increasing the accuracy of radiographic screening. The objective of applying computer-aided diagnosis in medicine, and to mammography in specific, are outlined. The chapter concludes with an examination of a couple of mammographic workstations that are, or in the process of, being used to support computer-aided diagnostic research.

2.1 *Introduction*

A staggering one out of eleven females will develop breast cancer by age 75 [Tanne, 1993] -- in 1991 alone, approximately 44,000 women died of this disease in the United States [Doi and others, 1993a]. Asymptomatic screening with mammography is the most effective method currently available for the early detection of breast cancer [Giger and others, 1993a], capable of reducing mortality by 20 to 40% [Doi and others, 1993a]. Hence, mammography is becoming one of the most standard, largest volume x-ray procedures interpreted by radiologists [Giger and others, 1993a]. However, only 10 to 20% of the surgical biopsies conducted as a result of suspicious findings prove to be malignant [Giger, 1993b] [Giger and Vyborny, 1993]. The large number of false-positive mammograms is compounded by the significant number of false-negatives, that is, 10 to 30% of women who have breast cancer and undergo mammography have negative mammograms [Doi and others, 1993a] [Giger, 1993a] [Giger, 1993b] [Giger and Vyborny, 1993]. In approximately 66% of these false-negative mammograms, the radiologist failed to detect the cancer that was evident retrospectively [Doi and others, 1993a] [Giger, 1993a] [Giger, 1993b] [Giger and Vyborny, 1993]. Errors can occur from a number of factors including insufficient clinical data, inexperience, distractions during the radiograph reading, ending the radiograph reading after the first definite finding, not reviewing prior radiographs, and concentration on another abnormal finding related to the suspect lesion in question [Giger and others, 1993a]. Eye fatigue, low conspicuity of the lesion, and poor image quality are possibly additional contributing factors [Doi and others, 1993a] [Giger, 1993b].

2.2 *Screen-Film X-Ray Mammography: Principle, Signs of Breast Cancer, and Shortcomings*

The following information on the principle of screen-film x-ray mammography is drawn from [Richter and Claridge, 1993]. In screen-film x-ray mammography, the x-ray source illuminates the breast. The beam is attenuated as it passes through the breast because of photonic scattering and absorption. The level of absorption is influenced by the x-ray source and the density and thickness of the breast tissue. The photons that do pass through the breast reach a screen-film image receptor. The absorption of the photons by the intensifying screen stimulates the emission of visible light that causes the film to darken.

The two main radiographic signs of breast cancer are the presence of clustered microcalcifications and the presence of a mass [Doi and others, 1993a]. These lesions are visible because of the very fine x-ray attenuation differences with normal breast tissue [Yaffe, 1993a]. The following discussion on these two important indicators is taken from the work of [Doi and others, 1993a] and [Vyborny, 1993]. Clustered microcalcifications are found in 30 to 50% of all cancers detected mammographically. Tissue necrosis or obstructive phenomena associated with early neoplasms are responsible for the formation of these microcalcifications within tiny mammary ducts. Microcalcifications, being only 100 to 300 microns in size, are very hard to detect against the quantum and film grain noise present on mammograms. To detect these inconspicuous lesions, radiologists must often resort to the time-consuming and laborious process of using a magnifying glass to examine the mammogram.

Contrary to what one would expect, masses are also difficult to detect. Masses, considering they often share features in common with localized normal parenchymal densities, can be misinterpreted as, or hidden by, normal tissue. The degree of spiculation and margin definition are two key features used in the assessment of whether a mass is benign or malignant. Unfortunately, the noise present on mammograms can degrade the visibility of margins.

Conventional film-screen mammography suffers from a number of limitations, with the presence of noise as mentioned previously being one. Other shortcomings include non-linear film response; compromise between spatial resolution and x-ray quantum efficiency; and reduction in contrast, the dynamic range of the detector, and signal-to-noise due to x-ray scattering [Yaffe, 1993a] [Yaffe, 1993b].

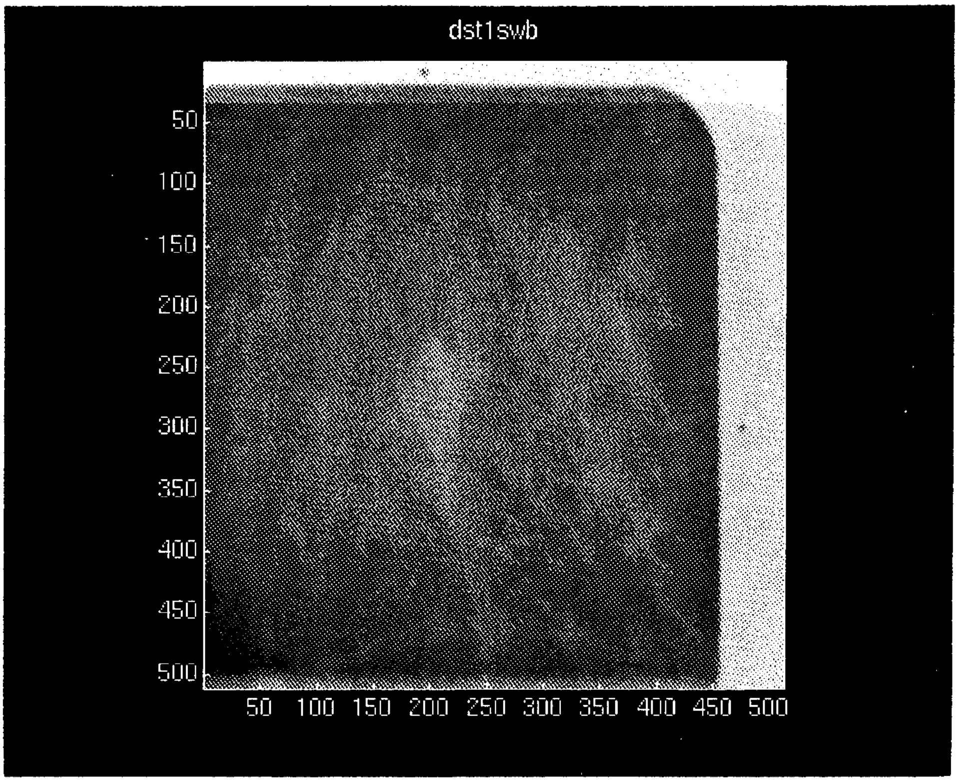
2.3 *Fine Needle Aspiration Biopsy and Surgical Biopsy: Limitations*

If a suspect lesion is detected during mammographic screening, the radiologist must decide whether the patient should return for another screening, be monitored through follow-ups, or scheduled for an immediate biopsy [Giger, 1993a] [Giger, 1993b]. Three techniques are currently used to obtain a breast tissue specimen: fine needle aspiration biopsy (cytology), surgical (excisional) biopsy, and stereotaxic core biopsy. The purpose of this section is not to detail the principle of fine needle aspiration biopsy and surgical biopsy, techniques that have been used for some time, but to identify some of their shortcomings which partly explains why the new technique of stereotaxic core biopsy discussed later is gaining acceptance in the medical community as an acceptable alternative.

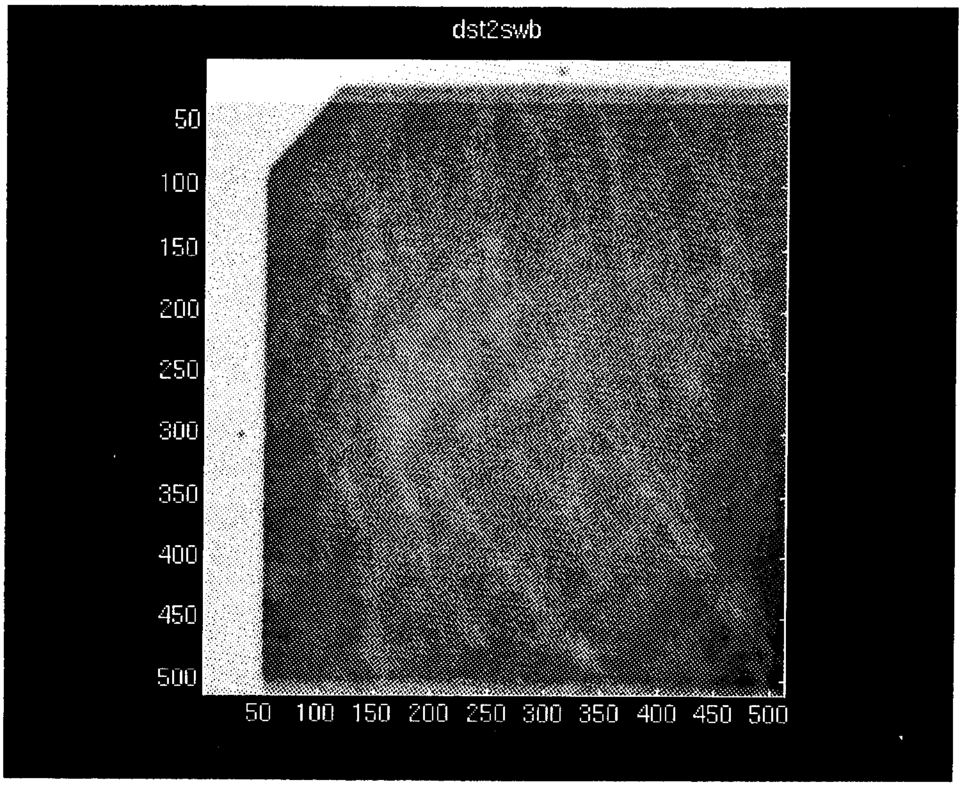
[Parker and others, 1990] list a number of weaknesses inherent in fine needle aspiration biopsy: (1) insufficient tissue sampling in 6 to 47% of cases; (2) false-negative findings in 1 to 31% of cases; (3) inability to differentiate in situ and invasive carcinoma; and (4) poor ability to make a definitive benign diagnosis. Surgical biopsies fare no better, described as "ritualistic rather than realistic" by [Parker and others, 1990]. Such a controversial remark is perhaps justified considering the shortcomings of surgical biopsy as pointed out in [Parker and others, 1990]: (1) profuse bleeding may occur, obscuring the operative field and prevent complete removal of deep lesions; (2) the surgeon can become disoriented and remove the wrong tissue if the wire marker inadvertently migrates, dislodges, or is transected; (3) poor communication between the radiologist and surgeon about the wire marker could cause the lesion not to be excised; and (4) the technique is time-consuming, potentially stressful for the patient, and requires significant resources. [Hendrick and Parker, 1993] identify additional drawbacks, including scarring that can confuse radiologic interpretation in the future and a significant risk of death.

2.4 *Stereotaxic Core Biopsy of Breast Lesions: Methodology and Advantages*

Stereotaxic core biopsy of breast lesions, employing primary digital acquisition, is a relatively new system. (Fine needle aspiration biopsy can also be performed with the stereotaxic unit, but is not discussed here.) The following description on the methodology of obtaining a breast tissue specimen with the Lorad Digital Spot Mammogram is derived from [Hendrick and Parker, 1993] [Parker and others, 1990], and the generous teachings of Shireen Braner, Radiology Technologist and Mammography Coordinator. The stereotaxic unit essentially consists of a table with a hole at the upper end that allows the protrusion of either breast as the patient lies face down. A compression plate, x-ray tube, image receptor, and biopsy apparatus are located underneath the table. The initial step is to acquire an image of the breast lesion undergoing biopsy to verify that it is within the window of the compression plate. Stereoviews are obtained by moving the x-ray tube +15° to the left and then -15° to the right of the centerline. Figure 2.1 shows the scout stereoviews of a malignant mass.



(a)



(b)

Figure 2.1 - Scout stereoviews (a) +15° and (b) -15°

For both computer images, a mouse pointer is positioned within the center of the lesion and then clicked. The computer uses these two reference markers to calculate the horizontal (x), vertical (y), and depth (z) of the lesion's center. The biopsy apparatus, or more specifically the biopsy gun, is then loaded with a sterilized, beveled slot inner needle shown in Figure 2.2 along with a covering sheath to be discussed later.

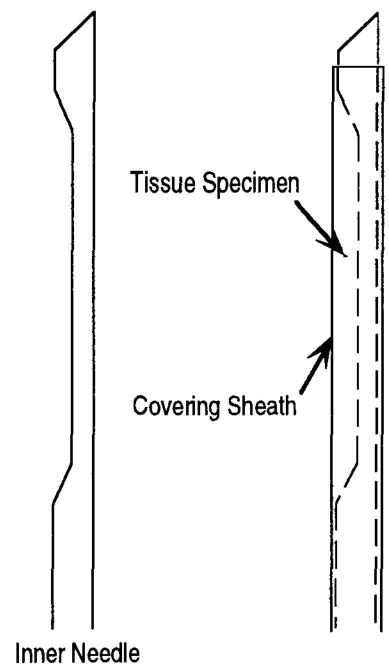
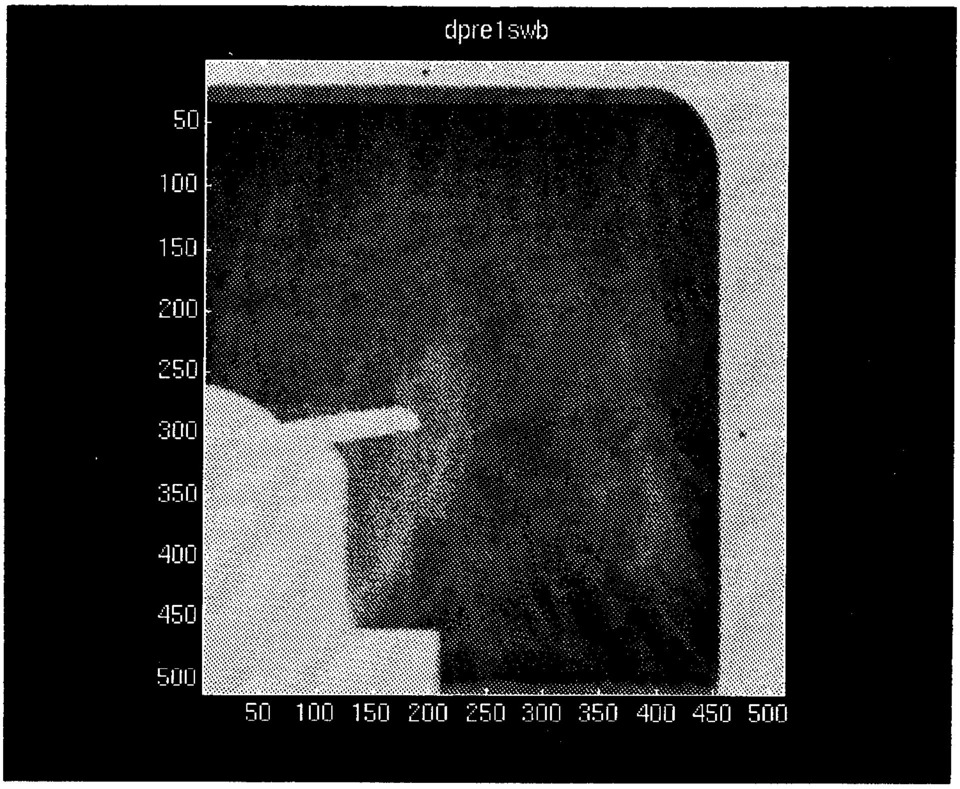
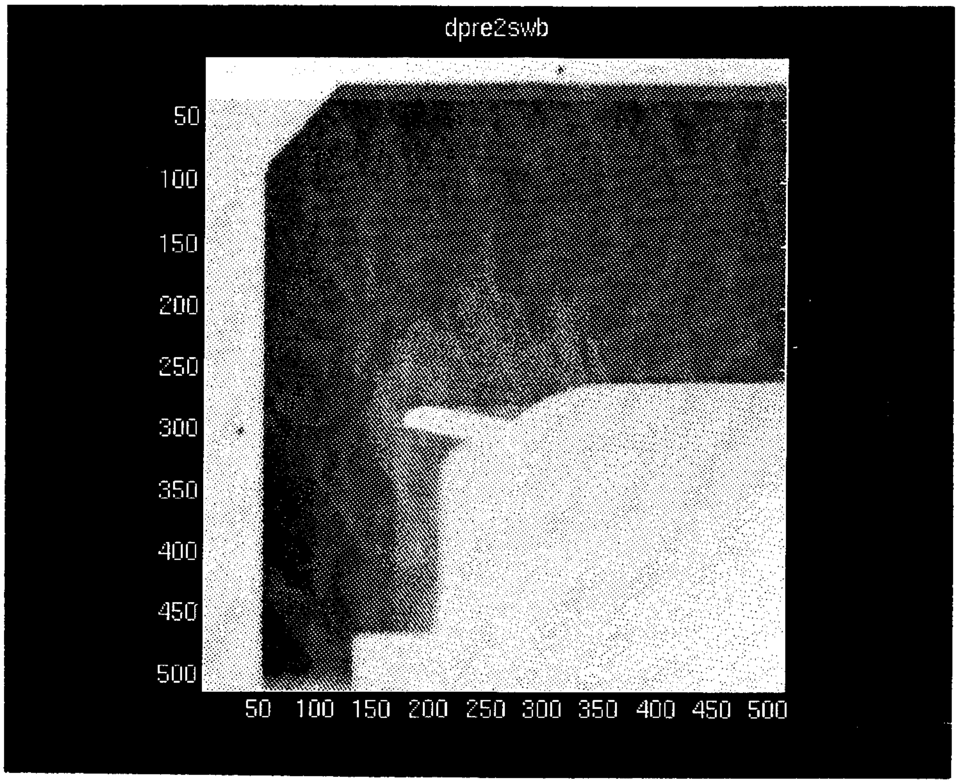


Figure 2.2 - Schematic of the two-part cutting needle [Hendrick and Parker, 1993]

The entry point on the breast is then prepped, which includes cleansing the site with a topical antiseptic, administrating a local anesthetic subcutaneously, and making a small incision less than a quarter-inch in length with a sterile surgical blade. Using the calculated coordinates, the biopsy apparatus is manually advanced until the tip of the needle is 5 mm from the border of the lesion. Additional stereoviews are obtained to confirm proper placement of the needle and to assist with minor adjustments that may have to be made if the patient has moved. Figure 2.3 shows the prefire stereoviews.


dpre1swb
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500

(a)



(b)

Figure 2.3 - Prefire stereoviews (a) +15° and (b) -15°

Once the position of the needle is correct, the biopsy gun is fired. The firing consists of two rapid sequential steps with the inner needle first penetrating the lesion, followed by a covering sheath that slides over the inner needle and cuts the sampled breast tissue within the beveled slot. Another set of stereoviews are obtained to ensure the inner needle and covering sheath have traversed the lesion. The needle placement stereoviews were used to demonstrate the *Mammogram Image Processing Program* and are shown in Figure 4.6. The inner needle and covering sheath are withdrawn, and the core tissue sample is removed and placed in a specimen bottle containing formalin. Four more lesion samples are obtained 3 to 5 mm directly above, below, to the left, and to the right of the centerline. The additional breast tissue samples are placed in the same specimen bottle and sent to pathology for analysis. A final stereoview is obtained to be used for comparative purposes with future radiographs.

Studies were conducted by [Dronkers, 1992] [Parker and others, 1990] and [Parker and others, 1991] comparing stereotaxic core biopsies with corresponding surgical core biopsies, yielding a 91%, 87%, and 96% agreement in the histologic results respectively. Interestingly, nearly all the patients considered the core biopsy procedure to be painless and easy, and the administration of premedications or sedatives were not required. In the thirteen-month study conducted by [Parker and others, 1990] using 103 patients with nonpalpable mammographically suspicious lesions, followed by a twelve-month study [Parker and others, 1991] using an additional 102 patients, no patients had any immediate or substantial complications. Several other advantages in using stereotaxic core biopsy versus surgical biopsy were identified including reduced costs from 25 to 50%, and prompt scheduling of the 20 to 30 minute procedure, even on the same day as the mammogram. Unlike fine needle aspiration biopsy which can have a high insufficient tissue sample rate, stereotaxic core biopsy obtains adequate tissue samples and enables definitive diagnoses of both benign and malignant lesions [Parker and others, 1991].

2.5 *Digital Mammography: Principle and Advantages*

Digital mammographic images are obtained either by (1) digitizing conventional film-screen images with a high-resolution optical scanner or (2) by means of primary digital acquisition as used in the Lorad Digital Spot Mammogram [Shtern, 1992]. In digital mammography, taking excerpts from [Hendrick and Parker, 1993] [Krupinski and Roehrig, 1994] [Richter and Claridge, 1993] [Romano, 1993] [Yaffe, 1993b], the image is composed of a two-dimensional matrix of numerical values, each equating to a pixel of the picture. A pixel's value is proportional to the magnitude of the x-ray transmitted on a path through the breast that intersects the location of the pixel on the screen-film image receptor or screen charged coupled device array image receptor. A film-screen image must be digitized to convert it into a numerical format suitable for storage in the computer's memory. This time-consuming, film handling and processing step is eliminated with primary digital acquisition since the image is already computerized. Primary digital acquisition does have a major drawback, the images as used in the Lorad Digital Spot Mammogram are restricted in size to 5 cm x 5 cm because of limitations in photodetector technology -- specifically, the charged coupled device array. Nevertheless, it is still a significant technological achievement that is serving as a step towards primary digital acquisition of the entire breast.

Digital mammography offers substantial advantages, including image processing, computer-aided diagnosis, and teleradiology [Shtern, 1992]. Image processing is used to optimize image contrast and enhance lesion detection [Shtern, 1992], with techniques ranging from contrast enhancement to histogram modification and spatial frequency filtering [Yaffe, 1993b]. The role of digital mammography in computer-aided diagnosis will not be discussed here since it is described in the following sections. In teleradiology, the image can be archived, retrieved, and displayed simultaneously at multiple locations [Yaffe, 1993b]. This will provide small localities without an experienced radiologist provisions for mammographic interpretation by experts [Yaffe, 1993b]. A number of shortcomings in teleradiology still need to be resolved, including the deterioration of spatial resolution resultant of digitization, transmission, and reconstruction [Shtern, 1992]; and slow transmission of images on current networks [Romano, 1993].

2.6 *Computer-Aided Diagnosis: Definition and Purpose*

[Doi, 1991a] [Doi, 1991b] [Doi and others, 1991] [Doi and others, 1993a] [Doi and others, 1993b] [Giger, 1993b] define computer-aided diagnosis as ". . . a diagnosis made by a radiologist who takes into consideration the results of an automated computer analysis of radiographic images." H. MacMahon in [Shtern, 1992] provides a much more specific definition: "A more sophisticated and technologically challenging form . . . in which the image is directly analyzed by computer with a view to detecting, localizing, and quantitating abnormalities in the radiograph." A very important point that needs to be kept in mind here is that computer-aided diagnosis is a tool that will provide the radiologist a "second opinion".

The purpose of computer-aided diagnosis are (1) to improve the reproducibility of image interpretation and (2) to improve diagnostic accuracy by reducing the number of false-positive diagnoses [Doi, 1991b] [Doi and others, 1991] [Doi and others, 1993a]. The method used to reach these goals entail bringing to the radiologist's attention potential lesions and objective measurements of normal and abnormal radiological patterns [Doi, 1991a] [Doi and others, 1993a]. According to [Vyborny, 1993], bringing potential lesions to the attention of the radiologist has been demonstrated to enhance the radiologist's identification performance.

The computerized algorithms that are being researched and developed for the detection of lesions in radiography are quite extensive and diverse. For example, at the Kurt Rossmann Laboratories for Radiologic Image Research, Department of Radiology, The University of Chicago, computer-aided diagnostic schemes are being developed to (1) detect pulmonary nodules and pneumothoraces, and characterize and detect interstitial disease and cardiomegaly in chest radiography; (2) detect clustered microcalcifications and masses in mammography; and (3) analyze stenotic lesions, and determine instantaneous and average blood flow rates in angiography [Doi, 1991a] [Doi, 1991b] [Doi and others, 1990] [Doi and others, 1991] [Doi and others, 1993a] [Doi and others, 1993b] .

2.7 *Computer-Aided Diagnosis in Mammography: Purpose*

Recapitulating [Giger, 1993b], the purpose of computer-aided diagnosis as it relates specifically to mammography are (1) to improve the screening procedures by assisting the radiologist as to the location of suspect lesions on mammograms, and (2) to reduce patient morbidity, as well as the number of surgical biopsies performed and their associated complications, by objectively analyzing and extracting the characteristics of suspect lesions.

A diversity of computerized techniques to detect and classify lesions in mammograms have been developed or are being researched. These techniques include but are not limited to the use of neural networks, filters, fuzzy logic, region growing, and gray level thresholding. The interested reader should consult [Astley and others, 1993] and [Giger, 1993b] for a synopsis of the computerized methods being developed by a number of research groups in this field.

2.8 *Mammographic Image Processing Environments*

The number of papers written on image processing algorithms as applied to mammography is nothing short of overwhelming. Yet in these papers the environment used to develop these algorithms is mentioned briefly or not at all. In the few papers written specifically on mammographic workstations -- two of which are briefly discussed below, the environment is not described in adequate detail or in relation to a framework. Thus it becomes practically impossible to judge the relative merits of like environments so as to determine which is the most suitable for a collaborative effort and possibly universal adoption.

The "Intelligent" Mammography Workstation as described in [Doi and others, 1993b] [Giger and others, 1993b] and [Giger and others, 1993c] is being used at the Kurt Rossmann Laboratories for Radiologic Image Research to develop computer-aided detection schemes for abnormalities in chest radiographs and mammograms. The descriptions of the workstation emphasize the platform specifications, but provide little information in regards to the environment other than screen buttons are used to command case selection, gray scale manipulations, and computer-aided diagnostic results.

CREDO (Computed Radiology, Enhancement, Display and Output) as described in [Giles and others, 1993] is a prototype clinical workstation for digital mammography being developed by FAXIL, Leeds General Infirmary, United Kingdom. The environment consists of (1) a graphical user interface designed using Suns OPENLOOK running under X-windows; (2) a file manager system that allows selection of cases either by examination, patient name, or patient ID; (3) an alignment system that lets correlated regions to be placed next to each other for comparison; (4) a x2 or x4 magnification utility; (5) a feature measurement system that uses a pair of cross hair calipers; (6) a default manager that permits individual preferences to be set; (7) feature enhancement algorithms; and (8) report writing tools. The semi-structured description of the workstation does not address such issues as the environment's ability to easily accommodate growth, execute on other platforms, and communicate with other environments.

2.9 *Conclusion*

Significant technological improvements are being made in mammography, with two of the more notable being core biopsy of breast tissue and primary digital acquisition. Although computer-aided diagnosis is in the early phases of development and clinical refinement [Doi and others, 1990] [Doi, 1991a] [Doi, 1991b] [Doi and others, 1991] [Doi and others, 1993a] [Doi and others, 1993b] [Giger, 1993a] [Giger, 1993b], continued research is of utmost importance considering the subjective nature of mammography where radiologists do make errors. Lodwick in [Doi and others, 1993a] echoes this point but extends it to a much more general level, commenting "The frequency of human error in medical decision making, including the interpretation of different types of radiographic images, is astonishingly high." [Doi and others, 1993a], citing Kundel and Revesz, state that "a major element for this error is the failure of the reader to detect the pattern of disease, which are often embedded in a background of structured noise and radiographic mottle."

The discussion of mammographic image processing environments tends to take on a subordinate role in relation to the discussions on computer-based diagnostic and detection algorithms. Yet the environments supporting the research of these algorithms are perhaps just as important. When the environments are discussed, the use of a framework is uncommon and thus certain capabilities and limitations tend to be overlooked. These ad hoc assessments make it impossible to judge the suitability and thus potential value of the various environments used to support computer-aided diagnostic research. In the succeeding chapters, the requirements, design, implementation, functionality, and operation of the *Mammogram Image Processing Program* are discussed. A quality model is used to assess the environment in the final chapter.

III. Requirements and Design

This chapter defines the attributes of each routine that was written to support the future research and development of mammographic detection and diagnostic algorithms. These routines are arranged according to a hierarchical architecture as shown using a number of diagrams. Finally, McCall's quality model as introduced in the first chapter is expanded upon and its intended use in regards to this research effort is defined.

3.1 *Introduction*

One of the valid concerns in the field of computer-aided diagnosis in mammography is the use of nonstandard mammogram databases by different research groups, making it impossible to determine the validity and usefulness of the detection and diagnostic algorithms that are being developed [Shtern and others, 1993]. Interestingly, the image processing environments being used to conduct the research and development of these algorithms are perhaps just as divergent, but not receiving comparable attention. When these environments are discussed, certain attributes tend to be overlooked, making it difficult to assess their potential value or even to make comparisons with like environments. Thus it becomes practically impossible to determine which environment is the most suitable and perhaps should be considered for universal adoption. The importance of a standard environment is perhaps nowhere more evident than in collaborative efforts where resources need to be shared. This raises a valid question of whether image processing environments used in this field should be evaluated against a bench mark such as McCall's *software quality factors*.

3.2 *Mammogram Image Processing Program Requirements*

The following section describes the attributes of essential routines that were developed to support the future research and development of complex segmentation, feature extraction, and classification algorithms. The attributes of several potentially useful, but optional utilities created as the capabilities of the MATLAB Image Processing Toolbox were explored are also discussed. The section concludes with a discussion of setting up function templates so that future researchers will be able to add image processing routines with relative ease. All routines were developed using MATLAB version 4.2, the MATLAB Image Processing Toolbox, and a Sun SPARC Workstation 2 utilizing SunOS 4.1.3 and OpenWindows version 3.0.

3.2.1 *Essential Utilities*

3.2.1.1 *Load Image.* A crucial design requirement of this program is that it is able to load multiple digital mammographic images of varying dimensions and gray levels. This is a necessity because multiple views can be associated with each case as in patients who undergo a stereotaxic core biopsy as described in section 2.4. As a side note, two standard views are used for each breast in conventional film-screen mammography, mediolateral oblique and craniocaudal [Paredes, 1993], yielding a total of four films [Giger and others, 1991]. All displayed images are annotated with their filename so as to eliminate confusion arising from a user who loads multiple images from different cases. Each image also includes x and y axis with tick marks in increments of 50 so that the user has a reference as to location and size of the suspect lesion(s). The program repeats the prompt for a user to enter an image filename that could not be located previously due to a misspelling, and disregards leading and trailing blanks accidentally entered along with the filename. These precautions needed to be accounted for to prevent the program from "crashing", and resulting in a long and productive work session being lost.

3.2.1.2 *Delete Image.* To prevent images from being accidentally deleted, an interactive safeguard is used. This safeguard simply asks the user to respond with a yes or no in regards to whether a particular image should be deleted. Any images that are still displayed when the user decides to terminate the work session are deleted.

3.2.1.3 *Zoom.* Magnification is an essential feature considering lesions can be small and/or subtle in nature and the images themselves can be of poor quality. The user can zoom in and out of an image by a factor of two, and is able to return to the original configuration without having to repeatedly zoom in or out.

3.2.1.4 *Label Image.* The current status of this utility is that it allows images to be annotated with text. Hopefully in the future, this utility will allow a radiologist to annotate images not only with text, but with arrows, circles, or a combination thereof, as to which regions of interest should be extracted by the researcher and run through various feature extraction routines. As a side note, the output from the feature extraction routines is then used to train a classifier to differentiate between what is normal, benign, or malignant. The label image utility allows additional radiologists to annotate the same, but unmarked set of images. The radiologists' interpretations can then be compared as a control measure to ensure a database of high radiographic standard is created.

3.2.1.5 *Reference Grid.* This utility displays a reference grid so that the user has a perspective as to the size of the region of interest that is being extracted via a non-graphical technique. The user can turn the grid on and off, alter the grid block size, and change the color and style of the grid lines.

3.2.1.6 *Region of Interest.* This utility allows the pixel values for a rectangular subsection of the image to be extracted via a mouse-driven and keyboard interactive technique. The user can extract rectangular regions of varying dimensions; assign a truth of either normal, benign, or malignant to these regions to train a classifier; and save these regions in an accountable manner so that they can be input to various feature extraction routines in the same or a later work session. The utility also saves the coordinates of the regions along with which image they were selected from. If the region of interest window extends beyond the boundaries of the image, the user is notified that the selection is invalid and will not be saved.

3.2.1.7 *Print to File.* The output from the execution of feature extraction routines is saved in an input data file format that is compatible to the particular classifier that is being used, in this case LNKnet. The image filename, or more specifically the region of interest file that was run through the various feature extraction routines is included as a comment in the input data file for accountability purposes. The user can print the results to a new file or to append them to a preexisting file.

3.2.2 *Optional Utilities*

3.2.2.1 *Histogram.* A histogram is a plot of the gray-level (pixel value) distribution for a particular image. This utility can be useful in identifying normalized images that in order to be displayed may require readjusting -- that is, additional code to be added to the program. A user can delete any histograms that are created.

3.2.2.2 *Intensity Profile.* An intensity profile is a plot of the gray-levels (pixel values) along a single path in a particular image. This optional utility can be used to note how the gray-level changes as the single path traverses suspect lesions. A user is able to use the mouse to get reference points, interactively enter the endpoints of the line, display multiple intensity profiles, and delete any profiles that are created.

3.2.2.3 *Density Slice.* A density slice is a color image generated by mapping pixel value ranges of the original gray-scale image to various colors. A user can enter different ranges, up to the maximum supported by the platform;

exploit all possible color maps provided by the application software; display multiple density slices using the same or different color maps; and delete any density slices that are created.

3.2.2.4 *Texture Map.* A texture map is a three-dimensional landscape mapping of the pixel values for a particular image onto a surface grid, where the elevation of the landscape is proportional to the brightness of the image [Bankman and others, 1993]. The user can display the texture map and the image simultaneously, and delete any texture maps that are created.

3.2.2.5 *Edge Extraction.* An edge extractor detects and displays sudden changes in pixel values. This optional utility can be used to examine the periphery of suspect lesions. A user is able to exploit all possible edge extractors provided by the application software, in this case Roberts, Prewitt, Sobel, and Marr-Hildreth; look for vertical, horizontal, or nondirectional edges if applicable to the detector; display multiple edge extractions using different detectors; complement edge extractions, and delete any edge extractions that are created.

3.2.3 *Function Templates*

The addition of an image processing routine written in MATLAB should be just a matter of adding a function call to the main script file. To make the addition process simple, a couple of function templates (test stubs) were set up to serve as a guide to future users of the program on how to add image processing routines. The test stubs proved to be helpful in verifying that: (1) region of interests could be passed to a function, processed, and the result(s) returned back to the main script file; and (2) that the print routines work properly.

3.3 *Mammogram Image Processing Program Design*

A top level interface diagram of the *Mammogram Image Processing Program* is shown in Figure 3.1. The mammogram image format(s) supported and the constructs of the input data file created for LNKnet will be discussed in the next chapter. As mentioned previously, the user interacts with the program by means of the keyboard and mouse. The focus of this section is on providing an overall layout of the program's architecture through the use of high-level diagrams. The program, arranged hierarchically through the use of menus, was created using a bottom-up approach. That is lower-level tested routines performing a similar subfunction were assembled together and tested. This combining and testing process was repeated until the integration of the main (driver) routine.

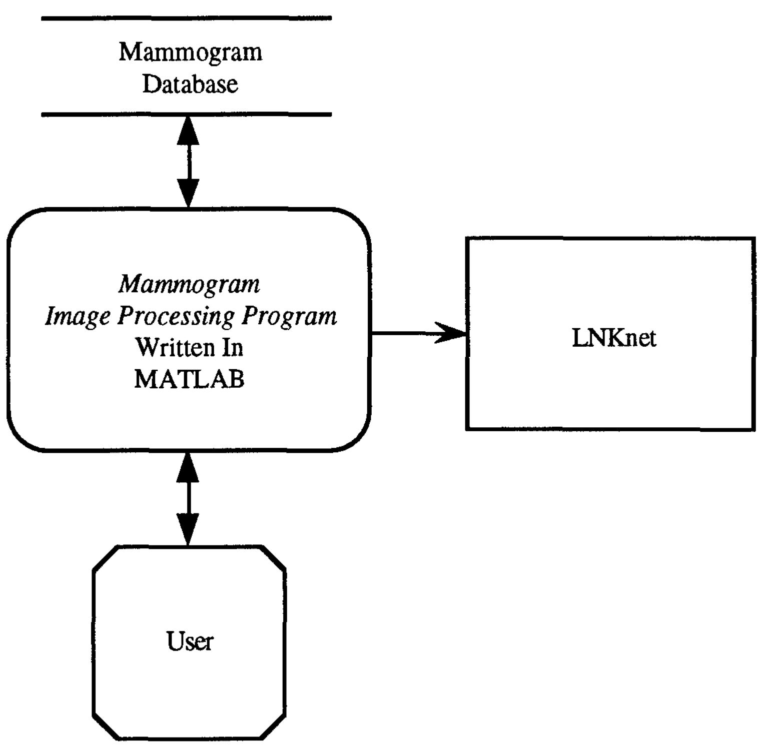


Figure 3.1 - *Mammogram Image Processing Program* overall diagram

The main menu, shown in Figure 3.2, offers the option to delete an image, or to show the database, display utilities, image processing, or image analysis menu. Not diagrammed is the option to terminate the program. The convention used in this and following figures is that a rectangle represents a leaf M-file (function), that may contain its own submenu, while an oval depicts an M-file that has children. Each figure also includes the M-file names. Although not diagrammed, all menus except the main provide the option to return to their respective parent menu.

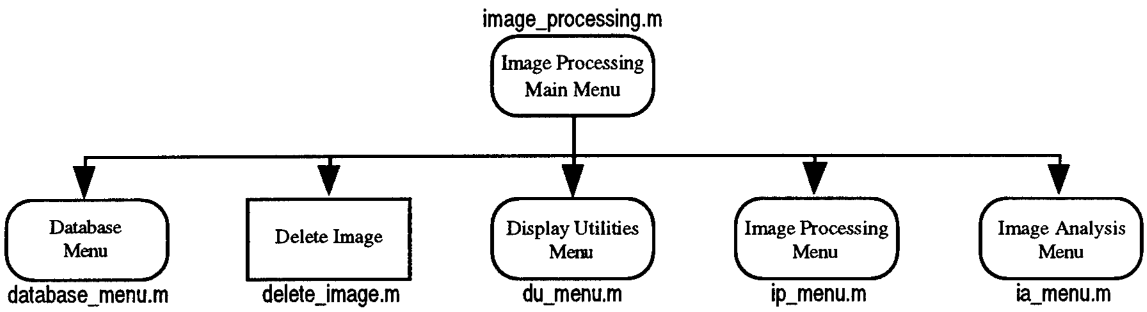


Figure 3.2 - Image processing main menu diagram

The program was tested using images from a Lorad Digital Spot Mammogram and a University of Cincinnati database. The selection of either database calls the same function as shown in Figure 3.3. The algorithm to load images from these databases is slightly different. To display the Lorad Digital Spot Mammogram images, the bytes had to be swapped using the UNIX *dd* convert and copy file command, followed by a raw data transpose and color map reversal in MATLAB. The images obtained from the University of Cincinnati did not require these extra preprocessing steps. Therefore when a database is selected, a flag is set which is used to determine if the extra preprocessing code should be executed.

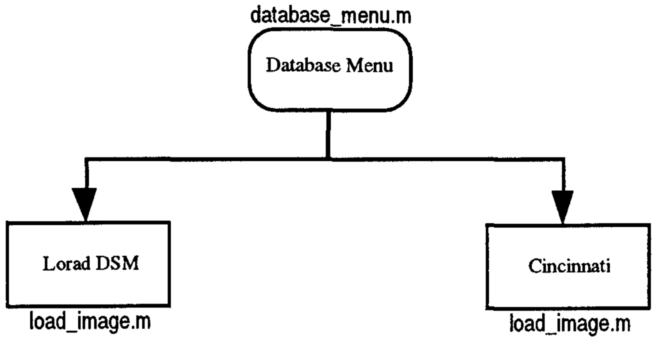


Figure 3.3 - Database menu diagram

Figure 3.4 shows the selections that are available in the display utilities menu. A user can generate a histogram, intensity profile, density slice, or texture map; display a reference grid; label an image; zoom; or delete an image. Each of these utilities other than delete image contains its own menu and possibly even submenu(s). For example, the density slice utility provides the options to demonstrate the color maps, display a density slice, and to return to the previous menu which in this case would be the display utilities menu. The dashed rectangle represents an internal routine used to reload any of the displayed images as the current one for processing. The flag set during database selection is used to ensure extra preprocessing code is executed if warranted.

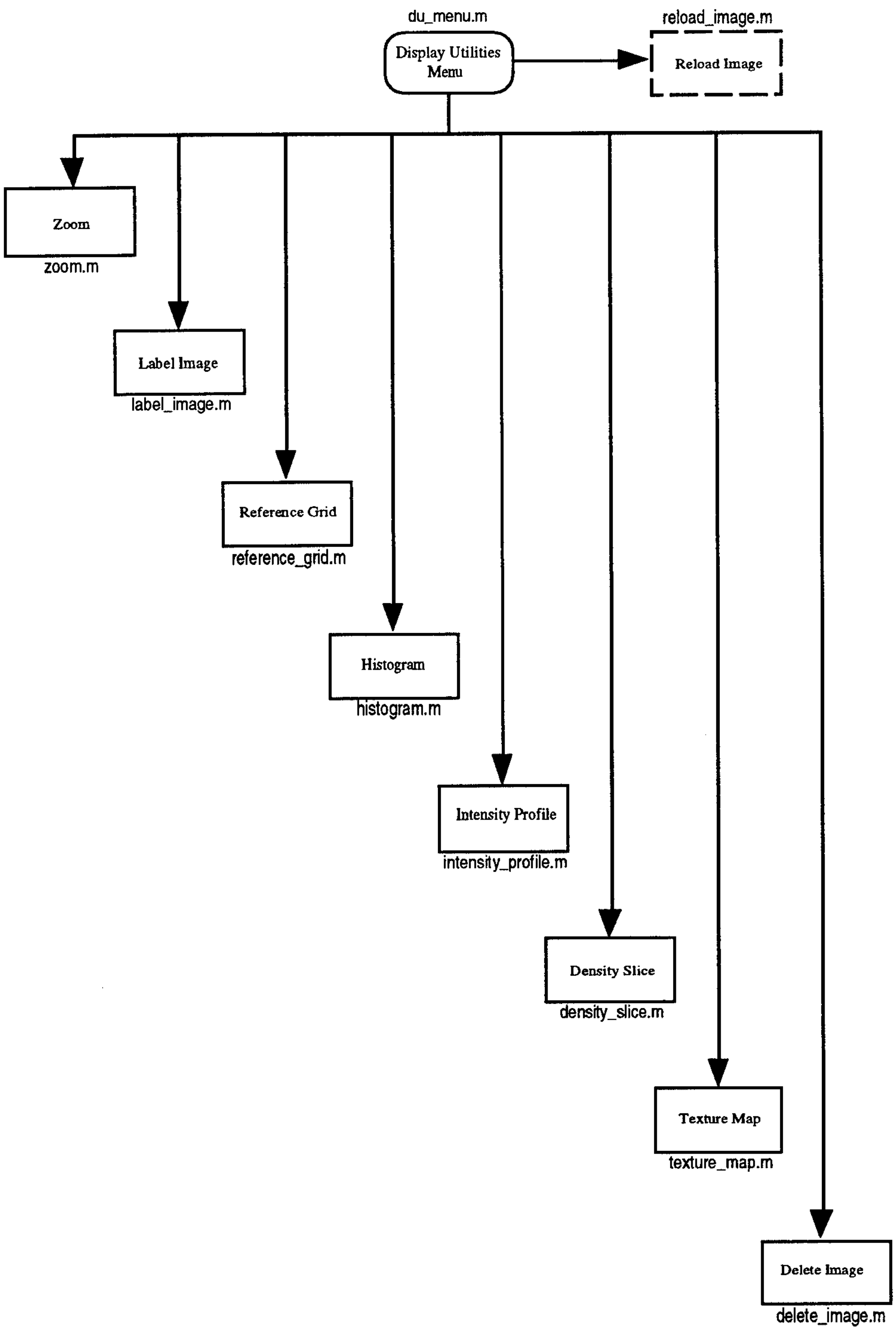


Figure 3.4 - Display utilities menu diagram

The image processing menu, shown in Figure 3.5, offers the choices to extract regions of interest and save them to a file, use four different edge detectors, or delete an image. The region of interest and edge extraction options contain their own submenus.

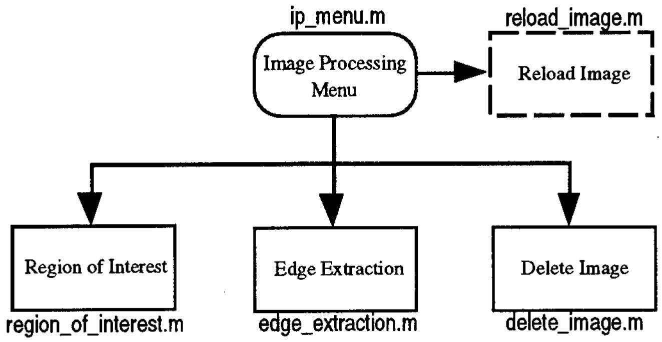


Figure 3.5 - Image processing menu diagram

Diagrammed in Figure 3.6 is the image analysis menu which provides the options to display the feature extraction, segmentation, or classification submenu. The feature extraction submenu, shown in Figure 3.7, offers the choices to load a region of interest file, execute any or all of the test stubs, and print the results to a file. The segmentation and classification entries do not in actuality contain submenus, but were included to guide future developers as to where these routines should be placed. These two stubs were tested for print verification of entry and return.

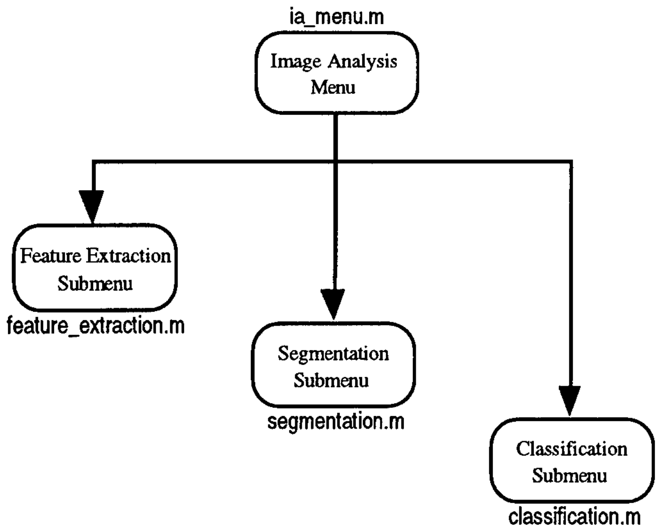


Figure 3.6 - Image analysis menu diagram

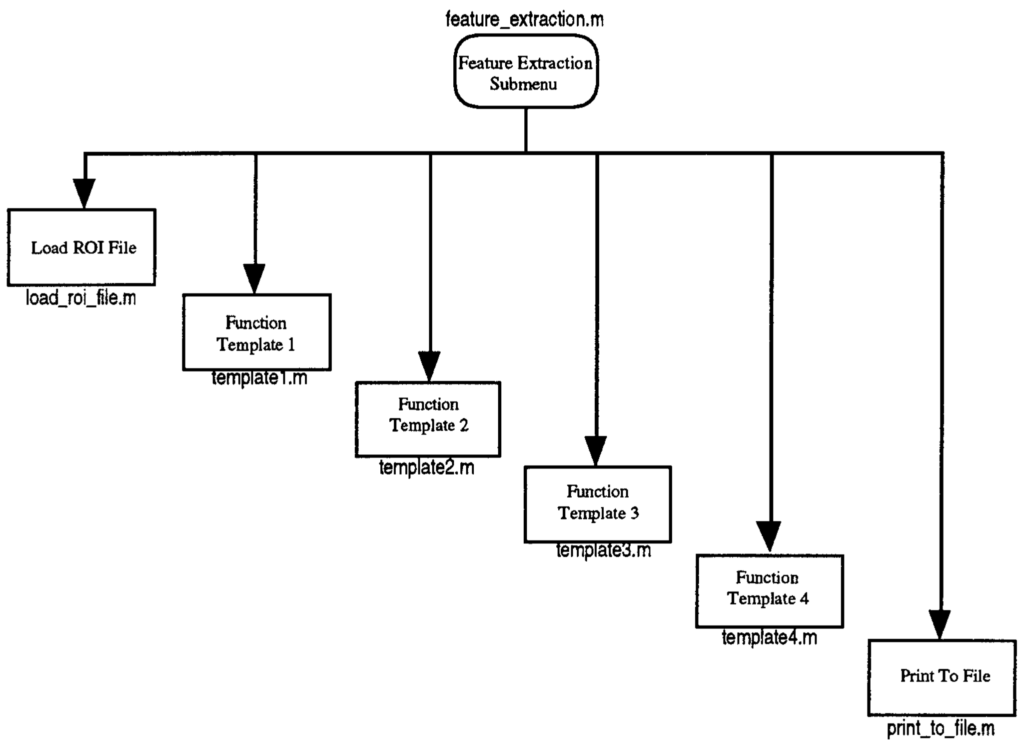
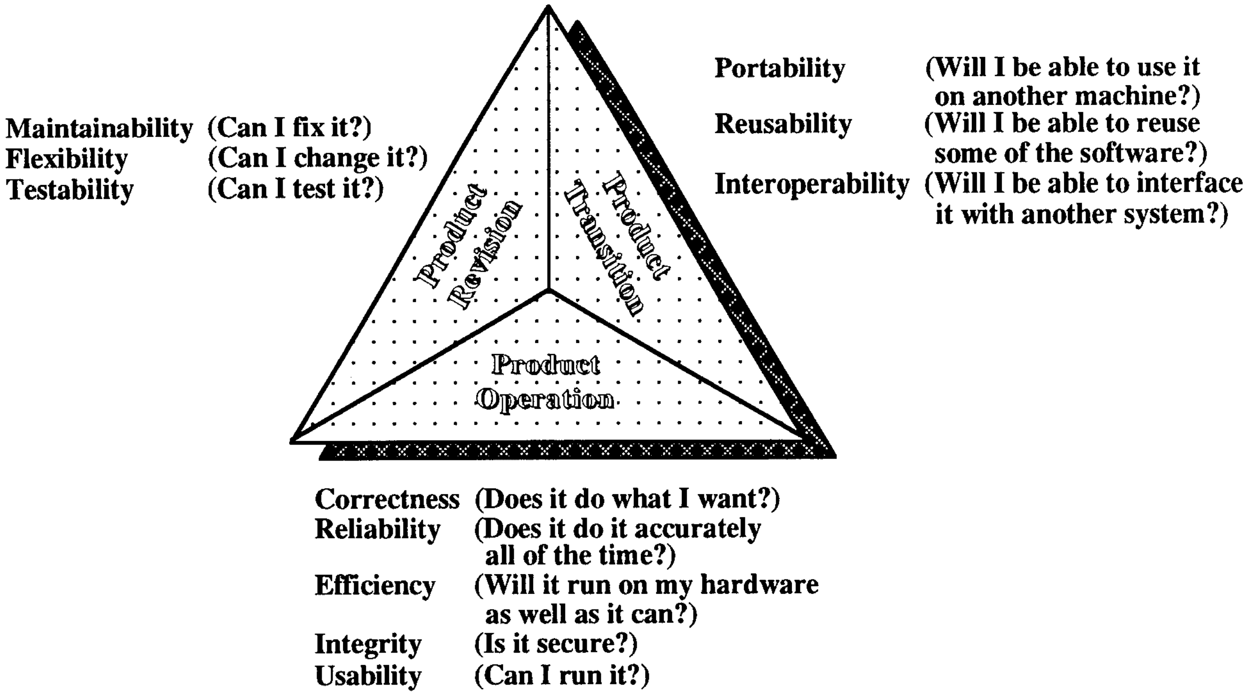


Figure 3.7 - Feature extraction submenu diagram

3.4 *McCall's Software Quality Factors*

In this section, McCall's *software quality factors* are examined in closer detail and their intended use in regards to this research effort is explained.

3.4.1 *A Closer Examination.* McCall's *software quality factors* is a hierarchical model first introduced in 1977 [Gillies, 1992]. This model serves as the foundation for much quality work even today [Gillies, 1992]. As shown in Figure 3.8, the model is used to assess three areas of a software program: (1) *product operation* focuses on performance and design issues, (2) *product revision* focuses on design and adaptation issues, and (3) *product transition* focuses solely on adaptation issues [Evans and Marciniak, 1987]. These three areas are comprised of eleven quality factors shown also in Figure 3.8.



>>>>>>>>>>>>>>>>>>>> QUALITY FACTORS <<<<<<<<<<<<<<<<<<<<

CORRECTNESS - The extent to which a program satisfies its specification and fulfills the customer's mission objectives.

RELIABILITY - The extent to which a program can be expected to perform its intended function with required precision.

EFFICIENCY - The amount of computing resources and code required by a program to perform its function.

INTEGRITY - The extent to which access to software or data by unauthorized persons can be controlled.

USABILITY - The effort required to learn, operate, prepare input, and interpret the output of a program.

MAINTAINABILITY - The effort required to locate and fix an error in a program.

FLEXIBILITY - The effort required to modify an operational program.

TESTABILITY - The effort required to test a program to ensure that it performs its intended function.

PORTABILITY - The effort required to transfer the program from one hardware and/or software system environment to another.

REUSABILITY - The extent to which a program (or parts of a program) can be reused in other applications. This is related to the packaging and the scope of the functions that the program performs.

INTEROPERABILITY - The effort required to couple one system to another.

Figure 3.8 - McCall's *software quality factors* [Pressman, 1987]

Each quality factor itself is related to independent attributes called *criteria* [Cooper and Fisher, 1979], or *metrics* by [Pressman, 1987], that can be used to judge, define, and measure quality [Cooper and Fisher, 1979]. Figure 3.9 shows the *criteria* that are used to define the correctness quality factor. These *criteria* along with regression coefficients are used to assign each *software quality factor* an expression [Pressman, 1987]. Unfortunately, many of these *criteria* can only be measured subjectively [Pressman, 1987] and brings to question the necessity of extending the assessment to this level, especially when only a rough assessment of the environment's quality is desired.

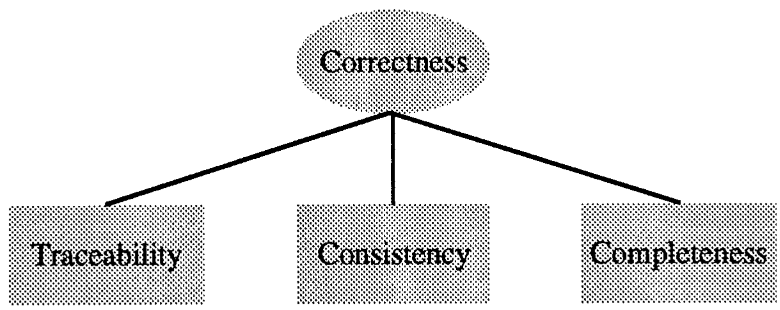


Figure 3.9 - *Criteria* that define the correctness quality factor [Cooper and Fisher, 1979]

3.4.2 *Intended Use.* McCall's model will not be extended to the equation level in order to assess the potential value of the *Mammogram Image Processing Program.* This is keeping in line with the overall objective of this research effort which is not to conduct an extensive quantitative software engineering analysis of a program, but to create an image processing environment and assess it according to a standard framework that would reduce the likelihood of certain system features being omitted due to accidental oversight and allow rough comparisons to made between similar environments if desired.

3.5 *Summary*

In this chapter, the routines written for the *Mammogram Image Processing Program* were discussed in terms of their attributes and hierarchical arrangement. The functionality and operation of these routines are detailed in the next chapter. A question posed in this chapter was whether or not environments should be evaluated against a standard framework, leading to a revisit and expanded discussion of McCall's quality model. In the final chapter, the program is assessed using a nonmathematical application of McCall's *software quality factors.*

IV. Implementation, Functionality, and Operation

This section discusses how the requirements and design outlined in the previous chapter were translated into an operational program. The features and use of each utility are described in a user's manual format.

4.1 *Preliminaries*

The hierarchical, bottom-up designed *Mammogram Image Processing Program* consists of twenty five functions and one script file called image_processing.m. The twenty six M-files are

```
classification.m            load_image.m
database_menu.m             load_roi_file.m
delete_image.m              print_to_file.m
density_slice.m             reference_grid.m
du_menu.m                   region_of_interest.m
edge_extraction.m           reload_image.m
feature_extraction.m        segmentation.m
histogram.m                 template1.m
ia_menu.m                   template2.m
image_processing.m          template3.m
intensity_profile.m         template4.m
ip_menu.m                   texture_map.m
label_image.m               zoom.m
```

The user can either run MATLAB from the directory containing these M-files, or change the working directory within MATLAB using the *cd* command. On-line help is available for each file and can be accessed by using the *help* command followed by the filename without the dot extension. For example, to read information on the histogram function, type *help* histogram.

To start the program and display the main menu, enter image_processing at the MATLAB prompt. The menu, as shown in Figure 4.1, contains the options to load an image from a database; delete an image; show the display utilities, image processing, or image analysis menus; or terminate the program. Following the menu title is a parenthesized Lx -- where L is for level and x is an integer. This abbreviation represents how many menus the user has traversed downwards from the main menu. In this case the user is at level one -- the root menu.

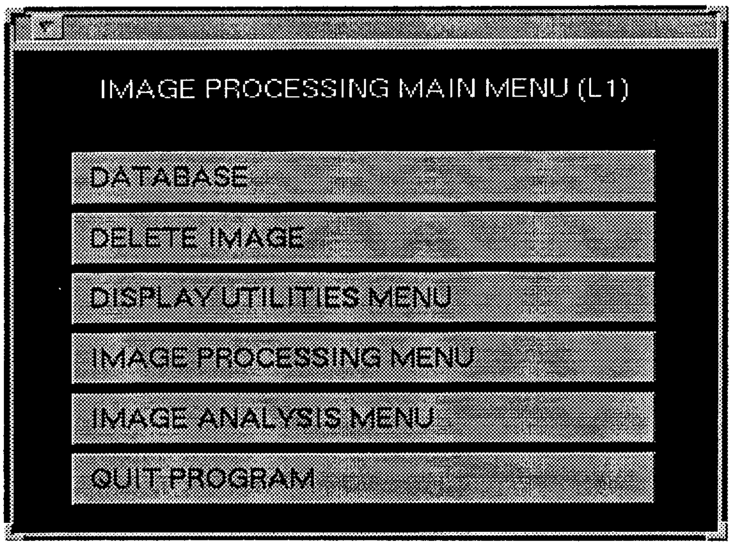


Figure 4.1 - *Mammogram Image Processing Program* main menu

4.2 *Database*

When this option is selected, a database menu pictured in Figure 4.2 is displayed, offering the alternatives to load images from a Lorad Digital Spot Mammogram (DSM) or a University of Cincinnati database. The algorithm used to load images from these databases is slightly different, with the Lorad DSM requiring a transpose of the raw matrix and a color map reversal. (The font is used to denote a variable name.) A database flag is used to discriminate for these differences.

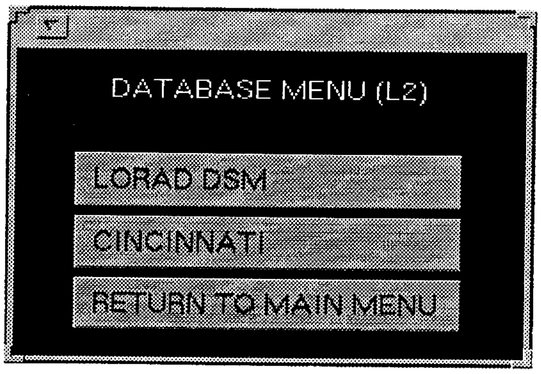


Figure 4.2 - Database menu

Choosing **LORAD DSM** or **Cincinnati** invokes a number of prompts starting with

```
Please enter the name of the image [dn1swb]:
```

The user can either enter a carriage return to select the default in the square brackets or type in the name of the file. As a side note, a carriage return is a valid choice for any prompt containing square brackets that identifies a default. Leading and trailing blanks entered along with the image filename are discarded since MATLAB considers these as part of the string and will search for such a file. If an invalid filename is entered, the user will be prompted again after the error message

```
File could not be located.
```

The next three prompts following entry of a legal filename are

```
Please enter the vertical dimension (rows) of the image [512]:
Please enter the horizontal dimension (columns) of the image [512]:
Please enter the gray level of the image [4096]:
```

A menu pictured in Figure 4.3 is used to request the numeric precision of the image, or in other words, the number of bits/pixel.

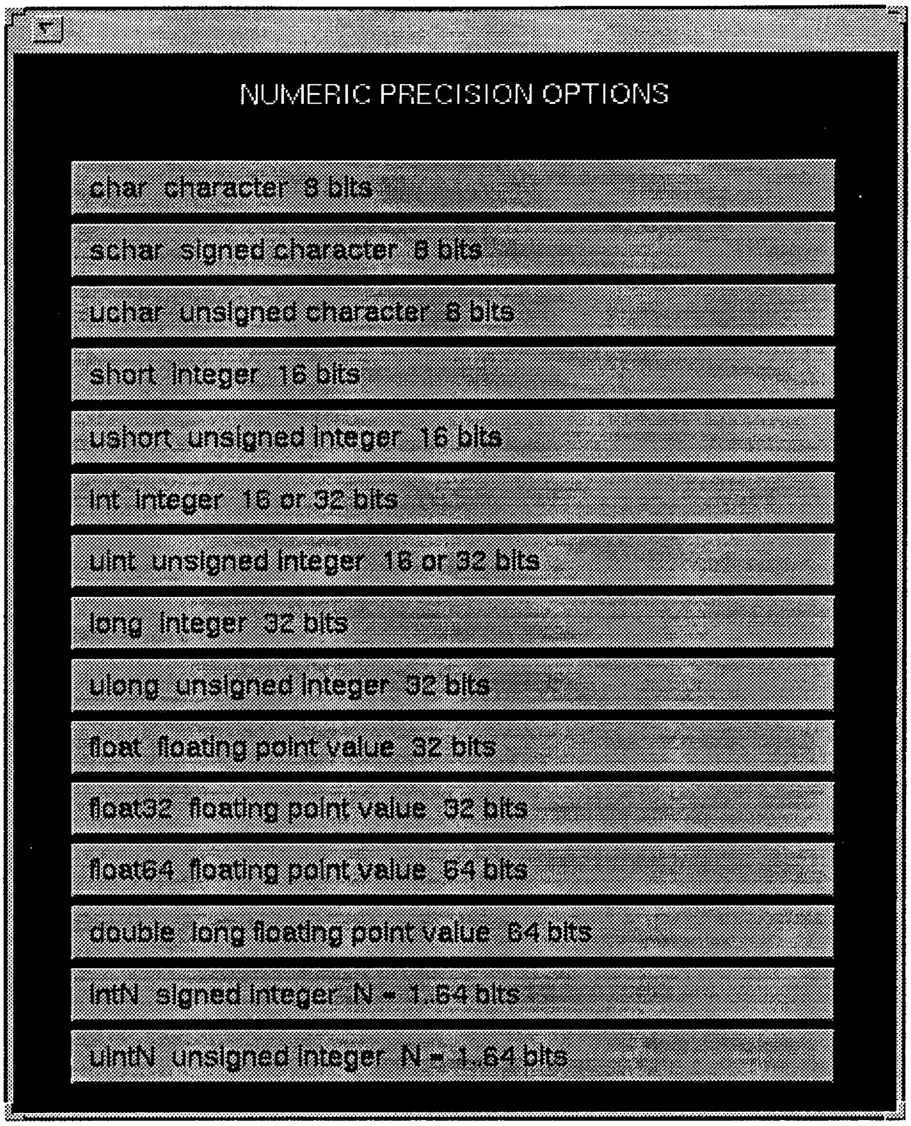


Figure 4.3 - Numeric precision options menu

The vertical dimension, horizontal dimension, and gray level are converted to strings prior to storage in a matrix (record) requiring this data type. Figure 4.4(a) illustrates the storage of image read, vertical dimension, horizontal dimension, gray level, numeric precision, and 'unloaded' in a seven field record called image read record. (Single quotes are used to denote a string.) The MATLAB command *str2mat* is used to insert each string (text parameter) into an image read record, padding each string with spaces in order to form a valid matrix. The last field in the record is reserved for the grid state described in reference grid. Figure 4.4(b) illustrates the record for an unloaded image, that is one derived from a loaded image. Histograms, intensity profiles, density slices, texture maps, and edge extractions are considered unloaded images. These images are discussed further in display utilities and image processing.

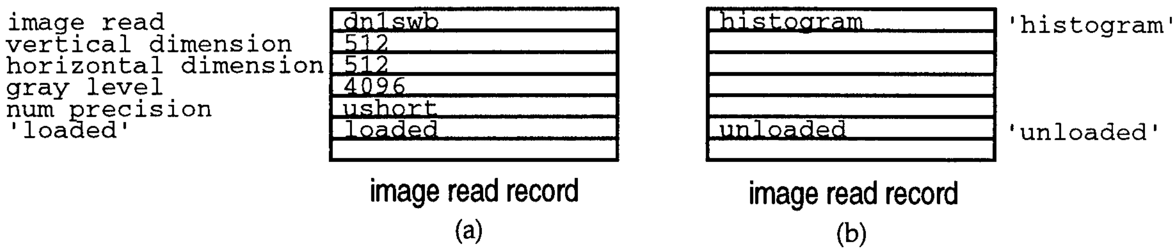


Figure 4.4 - Variable image read record (a) loaded image (b) unloaded image

Once the image read record is created, it is stored in images read catalogue, which is in essence the "kernel" of the program. For illustration purposes, the scenario that will be used is that the user, prior to loading an image, has been using the program for some time and thus images read catalogue contains a number of entries as depicted in Figure 4.5. The left margin represents the image figures currently displayed while the right margin shows the corresponding offset for each figure number that can be calculated by using the offset equation in Figure 4.5. Thus the current work session would have six images displayed, two loaded and four unloaded. Image figures three and eight represent deleted images as symbolized by the blank entries.

A linear search algorithm is used to insert the image read record into the catalogue. The first part of the algorithm creates an images read catalogue having the same dimensions as the image read record if image loaded is 'false', that is an image was not loaded previously. Image loaded must be 'true' in this case since a catalogue exists; therefore, the second part of the algorithm is executed. The second part starts off by calculating the number of records in the catalogue and assigning the result to index. The index is computed by dividing the number of rows in the images read catalogue by the number of fields in the image read record. Thus for the images read catalogue in Figure 4.5

```
index = 56 / 7 = 8
```

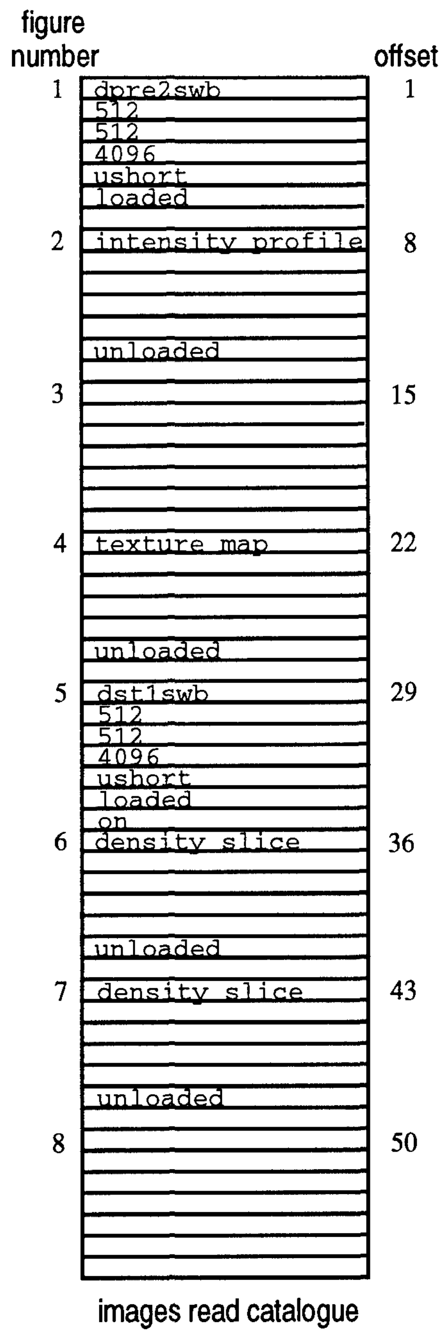
The index and offset equation are used to sequentially retrieve each string at the offset position. (Index is used in place of figure number since the catalogue contains empty records. Figure number is used for delete image, display utilities, and image processing.) If the retrieved string is blank, then the record is inserted into images read catalogue at the offset position. For example, when index equals one, 'dpre2swb' is retrieved since

```
offset = index + (6 * (index - 1))
       = 1 + (6 * (1 - 1))
       = 1 + (6 * 0)
       = 1 + (0)
       = 1
```

When index equals two, 'intensity profile' is retrieved. A blank string is retrieved when index equals three since

```
offset = index + (6 * (index - 1))
       = 3 + (6 * (3 - 1))
       = 3 + (6 * 2)
       = 3 + (12)
       = 15
```

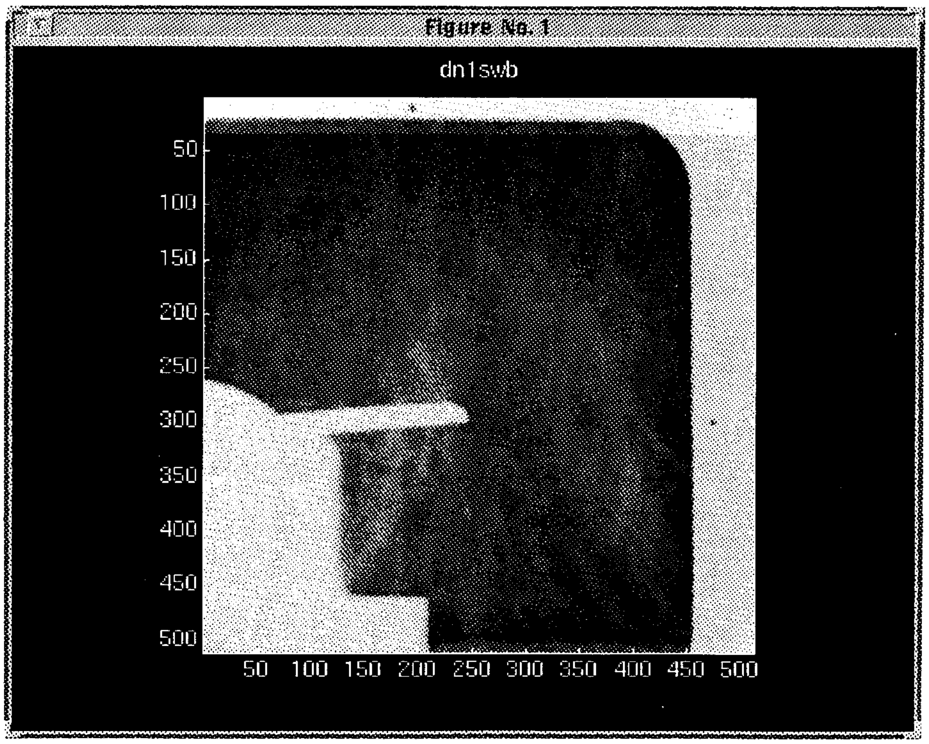
Thus the image read record will be inserted into images read catalogue at entries fifteen to twenty one, and the image will be assigned a figure number of three. The final part of the algorithm appends the image read record to the end of the images read catalogue if no empty records are found. MATLAB command *str2mat* is used to insert a record into images read catalogue. The command uses each record as a text parameter.



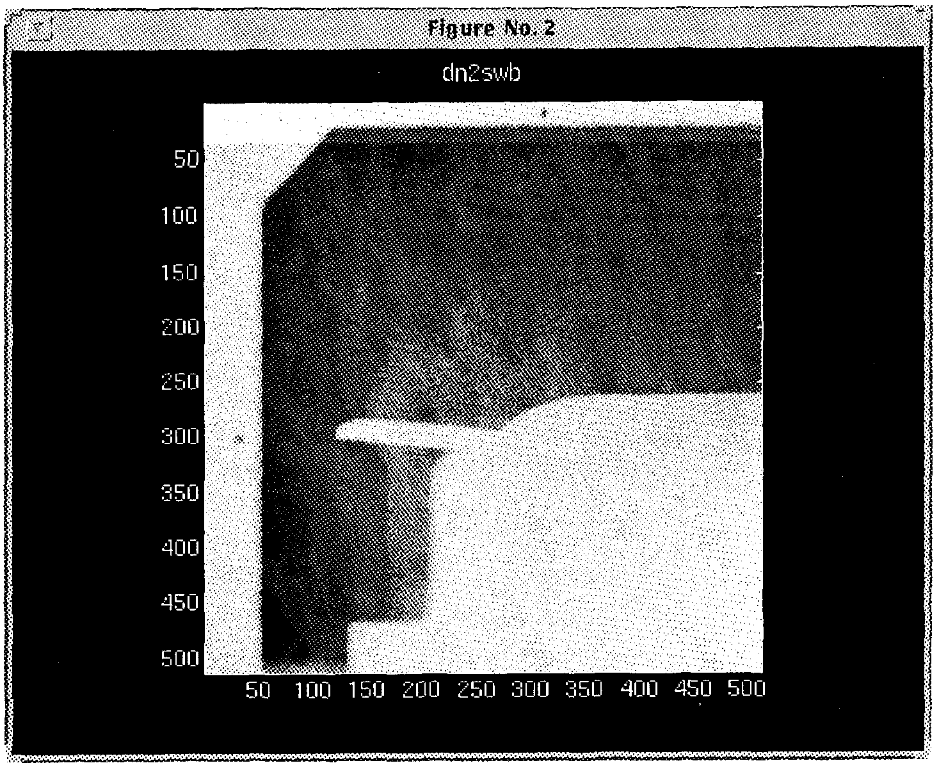
offset = figure number + (6 * (figure number - 1))

Figure 4.5 - Variables images read catalogue and offset

Vertical dimension, horizontal dimension, and numeric precision are used by MATLAB function *fread* to read the binary mammogram into raw matrix. MATLAB function *mat2gray* is used to convert the raw matrix to an intensity matrix whose double precision elements range in value from 0.0 to 1.0. MATLAB function *gray2ind* is used to transform the intensity matrix to an index matrix whose integer elements range from 1 to gray level. Two points should be mentioned about the conversion process: (1) The MATLAB Image Processing Toolbox does not support direct conversion from raw matrix to index matrix. (2) Some of the program and MATLAB functions require an intensity matrix argument while others require an index matrix. The index matrix is used by MATLAB function *imshow* to display the mammograms, two of which are pictured in Figure 4.6. The axes ticks and labels are specified using MATLAB function *set*, while MATLAB function *title* is used to place the filename above the image. Note that there is no association between the image figure numbers and the images read catalogue in Figure 4.5 used strictly for illustration purposes.



(a)



(b)

Figure 4.6 - Needle placement stereoviews (a) +15° and (b) -15°

4.3 *Delete Image*

Selection of this option when there are no images generates the error message

```
No images are displayed.
```

If there is at least one image, the user receives the prompt

```
What image figure number would you like to delete?
```

An entered figure number less than one, greater than the maximum figure number, or corresponding to a previously deleted image results in the error message

```
Invalid figure number entered.
```

The maximum figure number is calculated in the same manner as for the number of rocords in the images read catalogue described for the database utility. To determine if the entered figure number corresponds to a previously deleted image, the string at the offset position is assigned to image name and compared with a blank string. Prior to making this comparison, five is added to the offset and the string at this entry is assigned to image type. The offset is calculated by using the equation in Figure 4.5. If the figure number six is entered for example, the calculated offset equals thirty six, corresponding to a blank entry. The user would receive an error message in this case.

If a blank string is not retrieved, as by entering figure number two, the user is asked

```
Are you certain figure 2 should be deleted?
Enter yes or no [default]:
```

Offset and the six succeeding fields are overwritten with blank strings if the answer is affirmative. In this example entries eight to fourteen would be overwritten with blanks equivalent to the columnar width of matrix images read catalogue. An answer other than a lowercase or uppercase 'n', 'no', 'y', or 'yes' will cause the user to be prompted again after the error message

```
Please reenter your choice.
```

The final step compares image type to 'loaded', and if they are similar then number of loaded images is decreased by one; otherwise number of unloaded images is decreased. The user should note that a delete does not decrease the size of images read catalogue.

4.4 *Display Utilities*

The memory requirements of MATLAB and the size of the digital mammograms which can be as large as 2 Mb each necessitates the concept of a current loaded image. In this implementation methodology, multiple images can be displayed, but the memory expensive raw, intensity, and index matrices, as well as associated parameters such as image dimensions and gray level, are loaded in random access memory (RAM) only for the most recent loaded image. Another loaded image can be made current through a process of reloading which overwrites the same variables for the previous current image. Display and image processing routines can only be performed on the current loaded image, with minor exceptions for unloaded images. (Utilities zoom and label image can be used on unloaded images.) Thus when **display utilities menu** or **image processing menu** are selected, the user is queried as to which loaded image should be made current.

At least one image has to be loaded, but not necessarily displayed to use the display utility hierarchy. (Histograms, intensity profiles, density slices, and texture maps are created using the variables associated with the loaded image, not the display itself.) Thus selecting **display utilities menu** prior to loading an image results in the error message

```
Please load an image before selecting this option.
```

If an image was loaded previously, the user receives the prompt

```
What loaded image figure number would you like to make current?:
```

The algorithm used to check if the entered figure number is valid is similar to that described for delete image, with a few additions, including new image read used in place of image name. An entered figure number less than one or greater than the maximum figure number produces the error message

```
Invalid figure number entered.
```

A figure number corresponding to a previously deleted image causes the error message

```
No such figure exists.
```

An 'unloaded' image type can not be made current and thus invokes the error message

```
An unloaded image figure number is not a valid entry.
```

If new image read is the same as image read, that is the most recently loaded image, then the user receives the feedback

```
figure number is the current image.
```

The feedback is followed by the display of the display utilities menu pictured in Figure 4.7. If the entered figure number does not correspond to any of the criteria mentioned previously, then the specified image is reloaded prior to the display of the display utilities menu. In addition to new image read and image type which were read in the early part of the algorithm, the vertical dimension, horizontal dimension, gray level, and numerical precision of the image must be retrieved. The additional parameters are acquired by adding, respectively, one, two, three, and four to the offset. Padded blanks are truncated from all retrieved parameters, and the vertical dimension, horizontal dimension, and gray level are reconverted back to integers. The reloaded image is displayed in the same manner as described for database. The final part of the algorithm assigns new image read to image read, followed by the feedback

```
figure number is the current image.
```

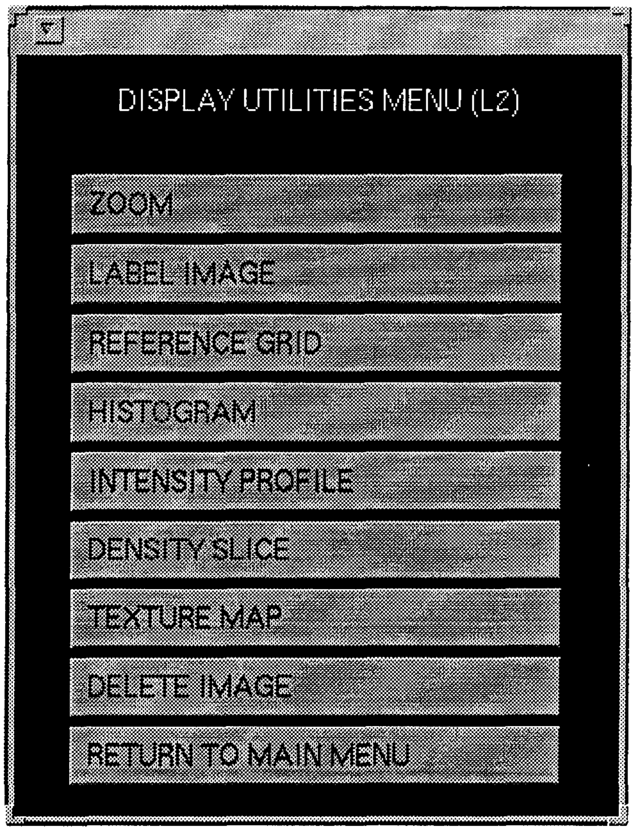


Figure 4.7 - Display utilities menu

The following utilities are demonstrated using dn1swb as the current loaded image.

4.4.1 *Zoom.* If there is at least one image when this utility is chosen, the zoom submenu pictured in Figure 4.8 is displayed; otherwise, the user receives the error message

```
An image must be displayed to select this option.
```

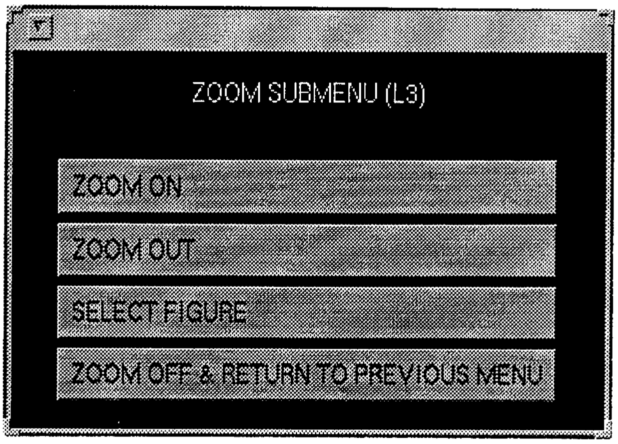


Figure 4.8 - Zoom submenu

Selection of **zoom on** prints the instructional message

```
Use the left mouse button to zoom in and the right mouse
button to zoom out.
```

The mouse pointer is used to define the zoom center, with each mouse click increasing or decreasing magnification of the image by x2 -- the image data is not changed, only the axes limits. Figure 4.14(b) is a sample x2 magnification obtained by positioning the mouse pointer near the tip of the biopsy needle of the image in Figure 4.14(a), then clicking the left mouse button once. To return the image to its original configuration without having to repeatedly zoom in or out, the user only has to click on **zoom out**. Zoom mode is toggled off for the current figure by selecting **zoom off**. All zoom features are performed by MATLAB function *imzoom*.

Select figure is used to make another image current, and thus allow the zoom features to be used on it. Choosing this alternative invokes the prompt

```
What image figure number would you like to select?:
```

The algorithm used to check if the entered **figure number** is valid is similar to that described for delete image. That is the **figure number** is inspected to see if it is less than one, greater than the **maximum figure number**, or corresponds to a deleted image.

4.4.2 *Label Image.* This utility when selected uses the same error control mechanism as zoom to check if there are any images. If a display does exist, the label image submenu pictured in Figure 4.9 is displayed.

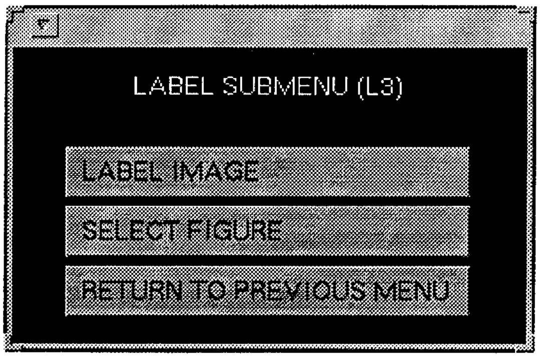


Figure 4.9 - Label image submenu

Label image is used to annotate the image, the selection of which generates the prompt

```
Enter the text:
```

A text entry followed by a carriage return prints the instructional message

```
Place the mouse pointer over the image where the text
is to be entered.  Click any mouse button to enter
the text on the image.
```

MATLAB function *gtext* is used to write the text onto the image at the desired position. Figure 4.10 demonstrates this feature, where the suspect lesion is annotated as a 'malignant mass'.

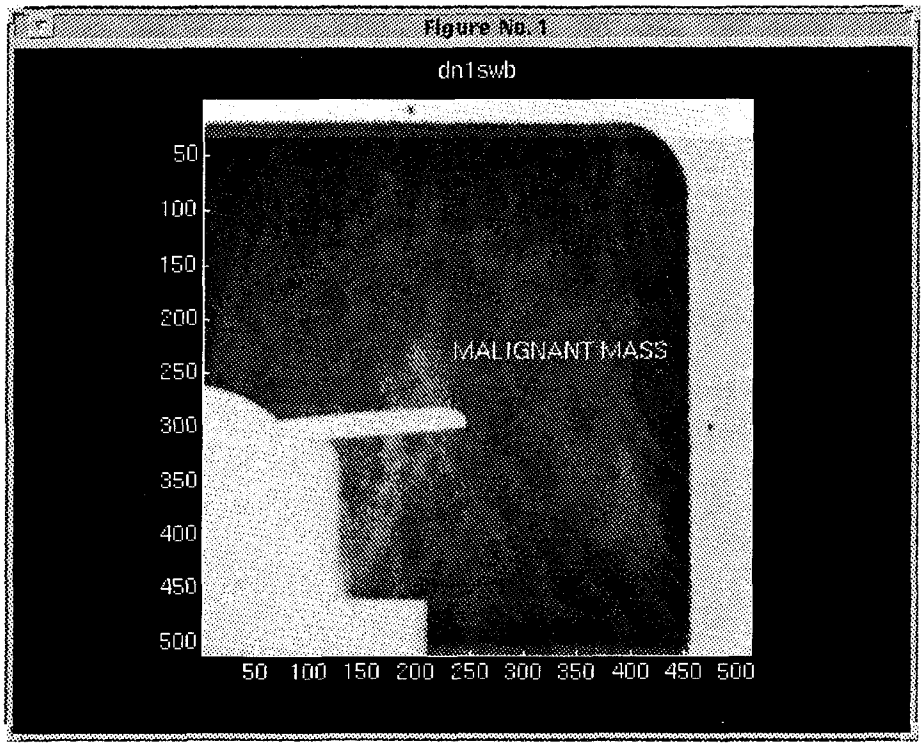


Figure 4.10 - Labeled image

The operation of **select figure** is identical to that described for zoom.

4.4.3 *Reference Grid.* Choosing this utility displays a reference grid submenu pictured in Figure 4.11.

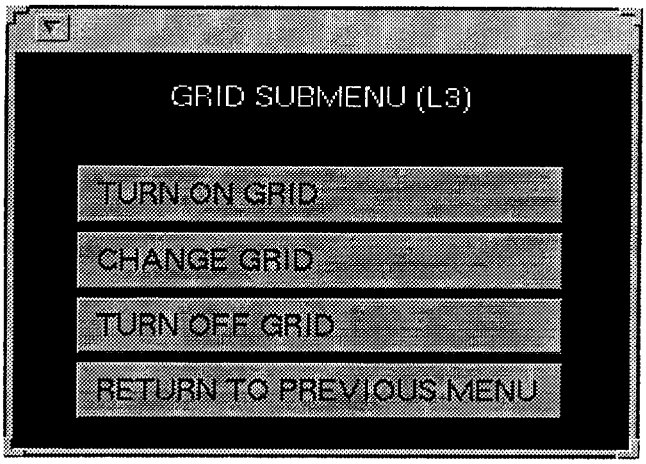


Figure 4.11 - Reference grid submenu

This utility can only be applied to loaded images. Thus selection of any alternative when the number of loaded images is zero results in the error message

```
There must be a loaded image to use this option.
```

If **turn on grid** is selected and there is at least one loaded image, the user receives the prompt

```
What loaded image figure would you like to turn the grid on for?:
```

The method used to check if the entered figure number is valid is similar to that described for display utilities. That is the figure number is inspected to see if it is less than one, greater than the maximum figure number, or corresponds to a deleted image. The grid can not be turned on for an 'unloaded' image type and thus invokes the error message

```
Grid can not be turned on for an unloaded image.
```

If the grid state is 'on' for a loaded image, then the user receives the error message

```
The grid is already turned on.
```

Grid state is retrieved from images read catalogue by adding six to the offset. If the figure number passes all checks, the user is asked

```
Please enter the grid block size [16]:
```

MATLAB function *grid* is used to turn the grid on. The grid spacing and axes labels are specified using MATLAB function *set*. To insert 'on' in the seventh field of a record in images read catalogue, 'on' is first padded with blanks. The padding is equivalent to the columnar width of images read catalogue minus two. Six is then added to offset to designate the insertion point of the padded 'on' into images read catalogue.

Change grid is similar to **turn on grid** up to the check to see if image type is 'unloaded'. An 'unloaded' image type results in the error message

```
Grid can not be changed for an unloaded image.
```

If the grid state is 'off' for a loaded image, then the user receives the error message

```
The grid must first be turned on.
```

If the figure number is valid, the user is asked

```
Change the grid block size yes or no [default]?:
```

A change block answer other than a lowercase or uppercase 'n', 'no', 'y', or 'yes' will cause the user to be prompted again after the error message

```
Please reenter your choice.
```

An affirmative change block invokes the prompt

```
Please enter the new grid block size:
```

The block size is not changed if a carriage return is entered. The user is next provided the option to change the grid color and is asked

```
Change the grid color yes or no [default]?:
```

The user is prompted again if the entry is invalid. An affirmative change color displays the color submenu pictured in Figure 4.12. The selection is used as an argument by MATLAB function *set* to change the grid color.

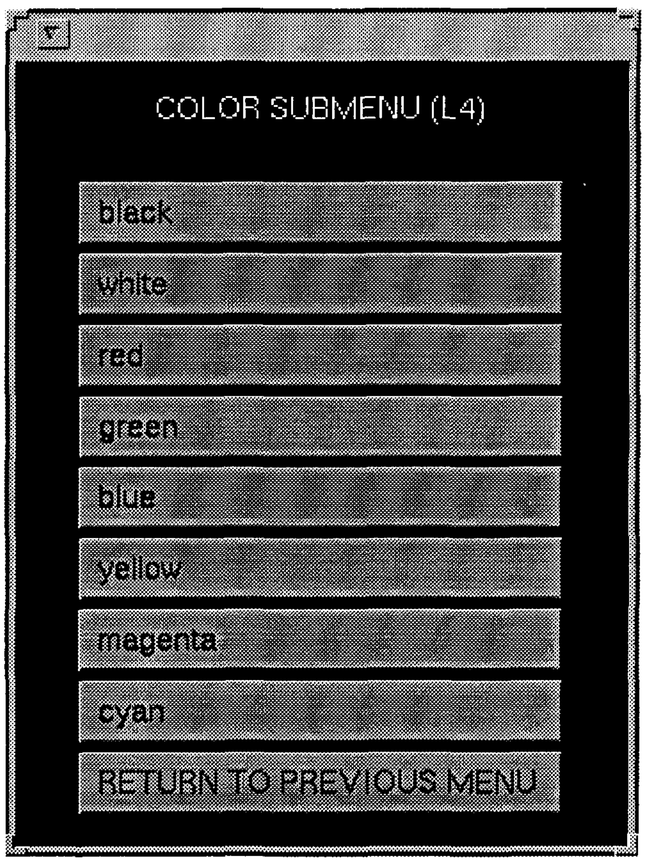


Figure 4.12 - Color submenu

The user is next provided the final option to change the grid line style and is asked

```
Change the grid line style yes or no [default]?:
```

The user is prompted again if the answer is not legal. An affirmative change line style displays the line style submenu pictured in Figure 4.13. MATLAB function *set* uses the selection as a parameter to alter the grid line style.

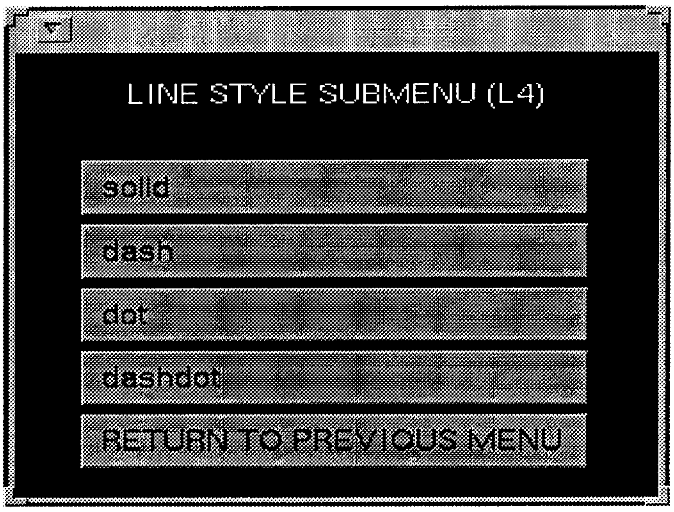
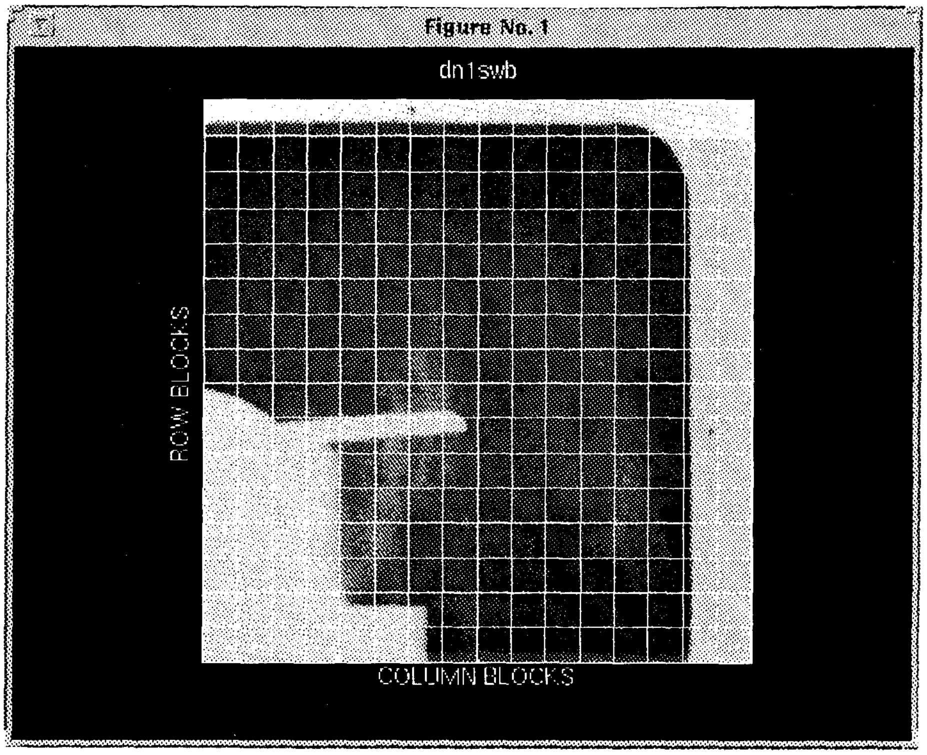
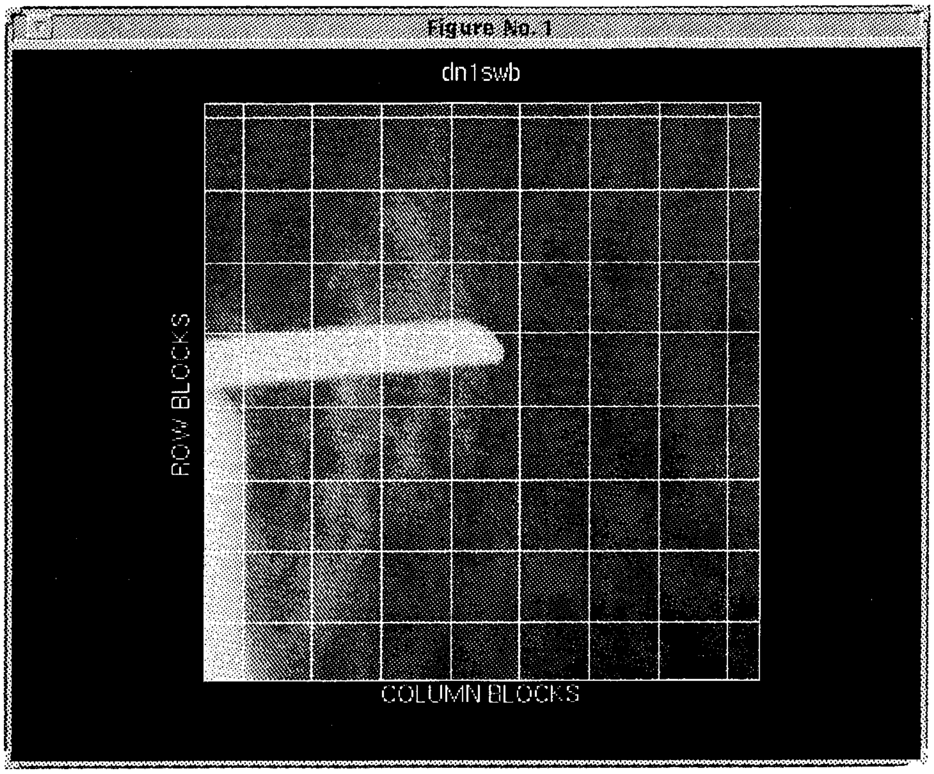


Figure 4.13 - Line style submenu

Figure 4.14 demonstrates an application of this utility using a 32 grid block size, yellow grid color, and solid line style.



(a)



(b)

Figure 4.14 - Reference grid (a) normal and (b) zoomed

Turn off grid is the exact opposite of **turn on grid** with the only differences being that off is used in place of on for all messages, the user is not prompted for the grid block size, and the blank padding used for 'off' is equivalent to the columnar width of images read catalogue minus three.

4.4.4 *Histogram.* This utility, along with functions intensity profile, density slice, and texture map, creates unloaded images by using the variables of the current loaded image. Thus an image is not required to use this utility, the selection of which displays a histogram submenu pictured in Figure 4.15.

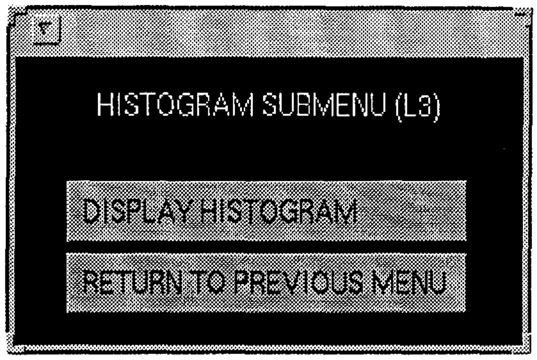


Figure 4.15 - Histogram submenu

Display histogram generates a histogram using the MATLAB function *imhist*. Figure 4.16 is a sample histogram of the current loaded image where the horizontal axis represents the gray level range and the vertical axis represents the pixel value distribution.

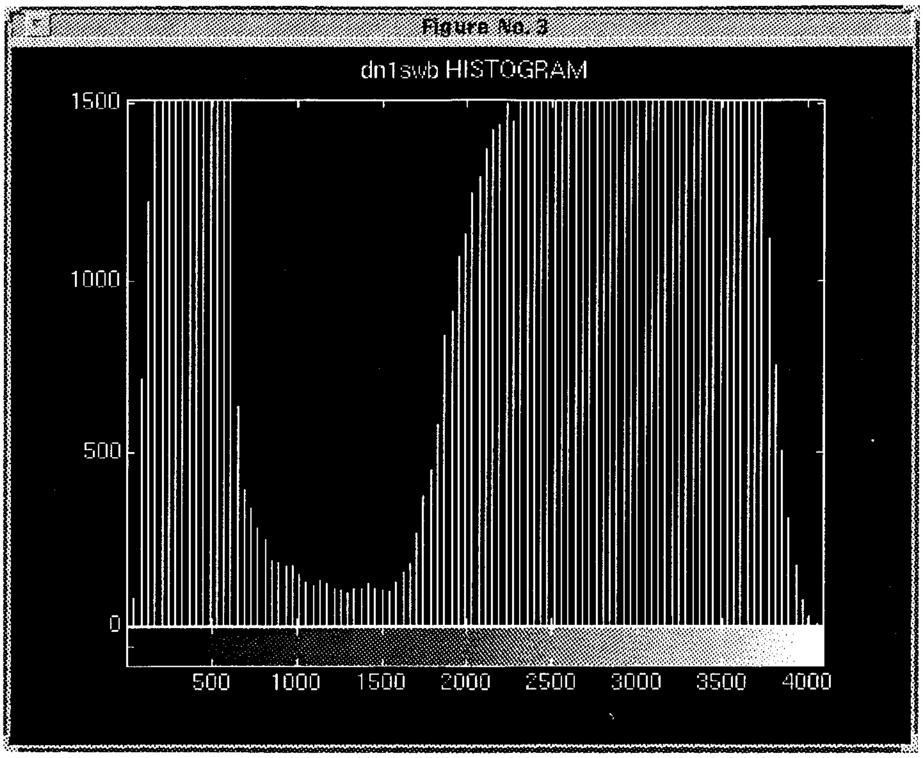


Figure 4.16 - Histogram

Once the histogram is displayed, the strings 'histogram' and 'unloaded' are stored in an image read record, with the record inserted into images read catalogue according to the algorithm discussed for database. The last step increments number of unloaded images by one.

4.4.5 *Intensity Profile.* Selection of this utility displays an intensity profile submenu pictured in Figure 4.17.

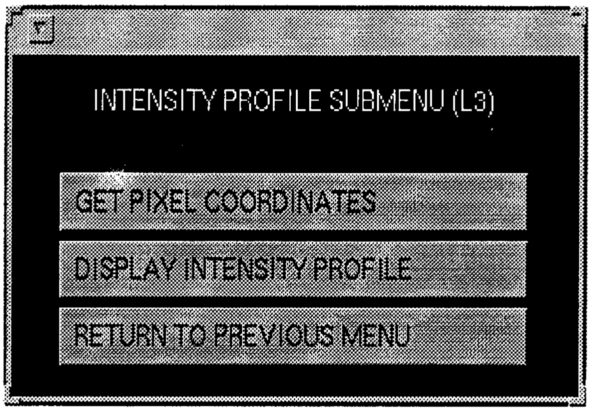


Figure 4.17 - Intensity profile submenu

The axes labels provide a rough estimate of the position of a lesion whereas **get pixel coordinates** determines its exact position. Selecting this alternative when there are no images results in the error message

```
An image must be displayed to select this option.
```

If there is at least one image, the user receives the prompt

```
What image figure number would you like to select?
```

The algorithm used to check if the entered **figure number** is valid is similar to that described for delete image. That is the **figure number** is inspected to see if it is less than one, greater than the **maximum figure number**, or corresponds to a deleted image. A legal entry makes the image figure current and prints the instructional message

```
Use any mouse button to get the pixel coordinates.
```

To obtain a set of coordinates, the mouse cursor is positioned over the desired image point, followed by a single mouse click. MATLAB function *ginput* is used to determine the float coordinates of the image point defined by the mouse cursor. The coordinates are rounded and converted to strings prior to being echoed back to the user in the format

```
The x coordinate is --> x string
The y coordinate is --> y string
```

Display intensity profile is used to create an intensity profile of a single path traversing the current loaded image. Upon selecting this alternative, the user is asked for the endpoints of the line using the prompts

```
Please enter x1:
Please enter y1:
Please enter x2:
Please enter y2:
```

MATLAB function *improfile* is used to plot the bilinear interpolated data of the line segment, where the total number of points plotted is equal to the number of pixels used to trace out the line. Figure 4.18 is a sample plot created using coordinates (1, 275) and (512, 275). The horizontal axis in the figure represents x position while the vertical axis represents the intensity, or gray level.

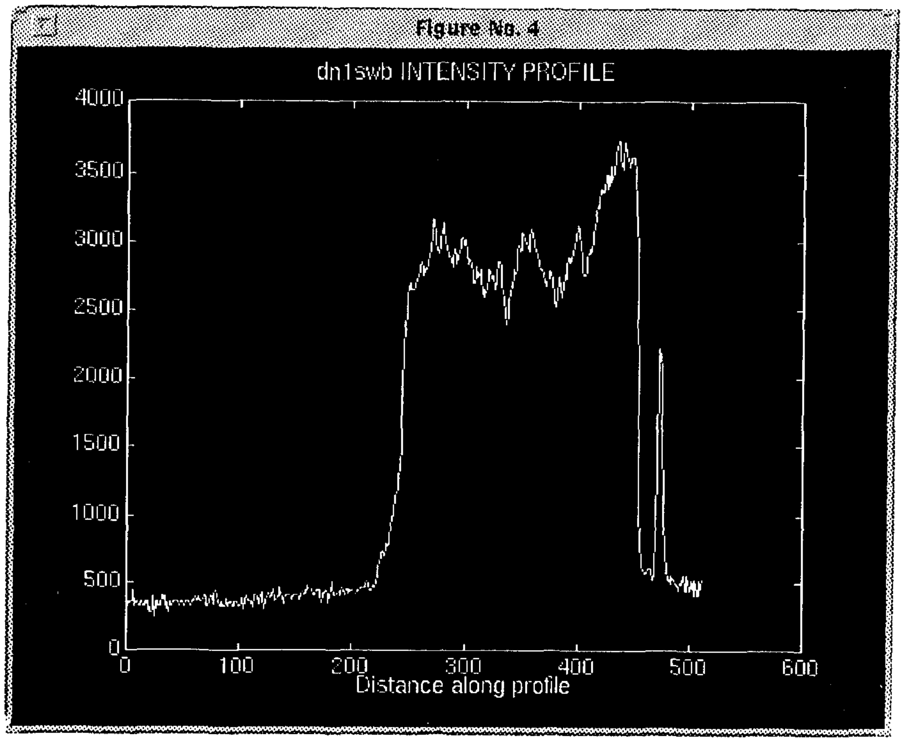


Figure 4.18 - Intensity profile

The plot is catalogued in the same manner as for a histogram, with the only difference being that 'intensity profile' is used in place of 'histogram'.

4.4.6 *Density Slice.* Choosing this utility displays a density slice submenu pictured in Figure 4.19.

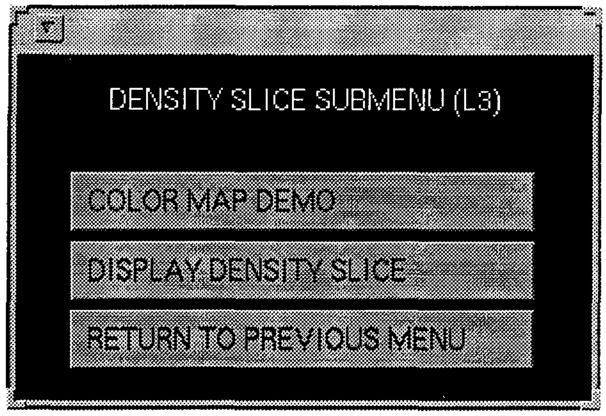


Figure 4.19 - Density slice submenu

Color map demo is useful for displaying the color maps supported by MATLAB. Selection of this alternative displays a color map options menu pictured in Figure 4.20. A color map preference is succeeded by the prompt

```
Please enter the number of color map levels [64]:
```

The color map level specifies the number of rectangular cells of the selected color map that are to be drawn by MATLAB function *pcolor*. A jet color map with 32 levels is shown in Figure 4.21. The same figure is used for additional color map demonstrations, and is automatically deleted when alternative **EXIT** is chosen.

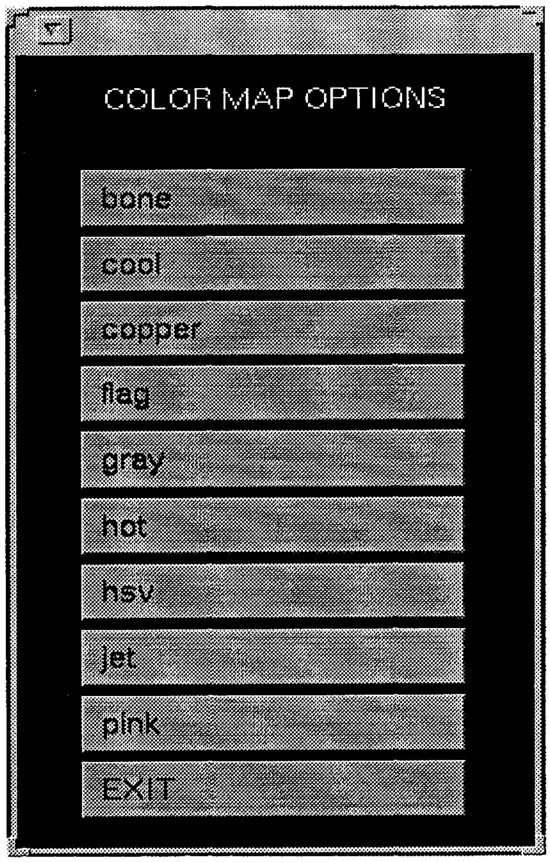


Figure 4.20 - Color map options menu

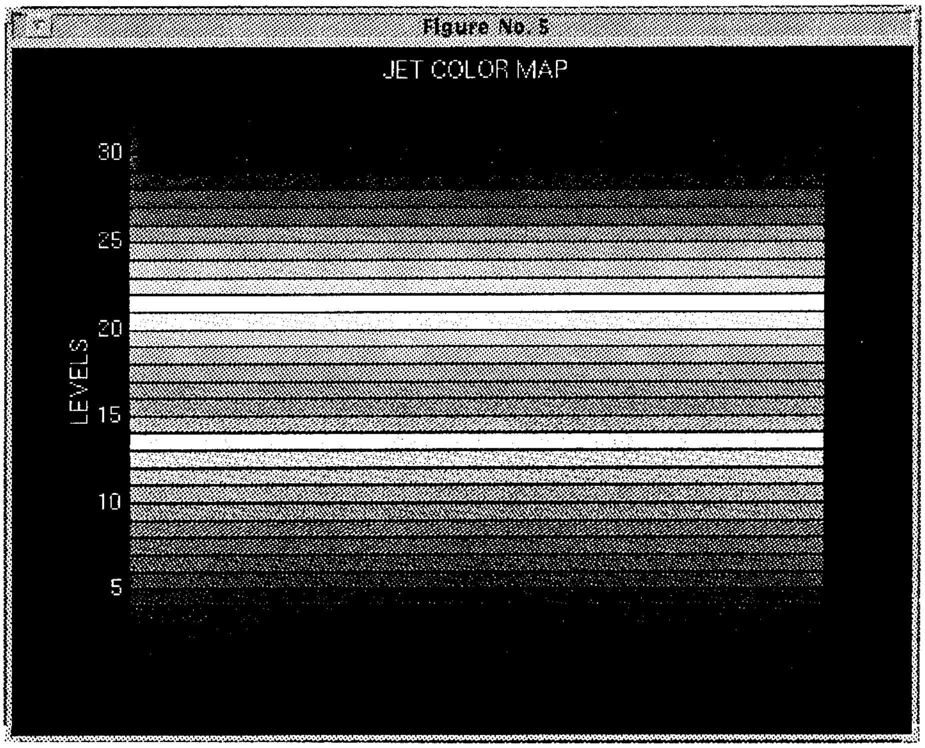


Figure 4.21 - Jet color map demonstration

Display density slice maps the image to the specified color map. Choosing this alternative displays the same color map options menu pictured in Figure 4.20. A color map choice is succeeded by the prompt

```
Please enter the number of slicing planes [20]:
```

Number of slicing planes enumerates the intensity bins that are to be used by MATLAB function *grayslice* in mapping the intensity matrix to the selected color map. A jet density slice with 32 levels is shown in Figure 4.22. The color bar to the right of the density slice was created using MATLAB function *colorbar*.

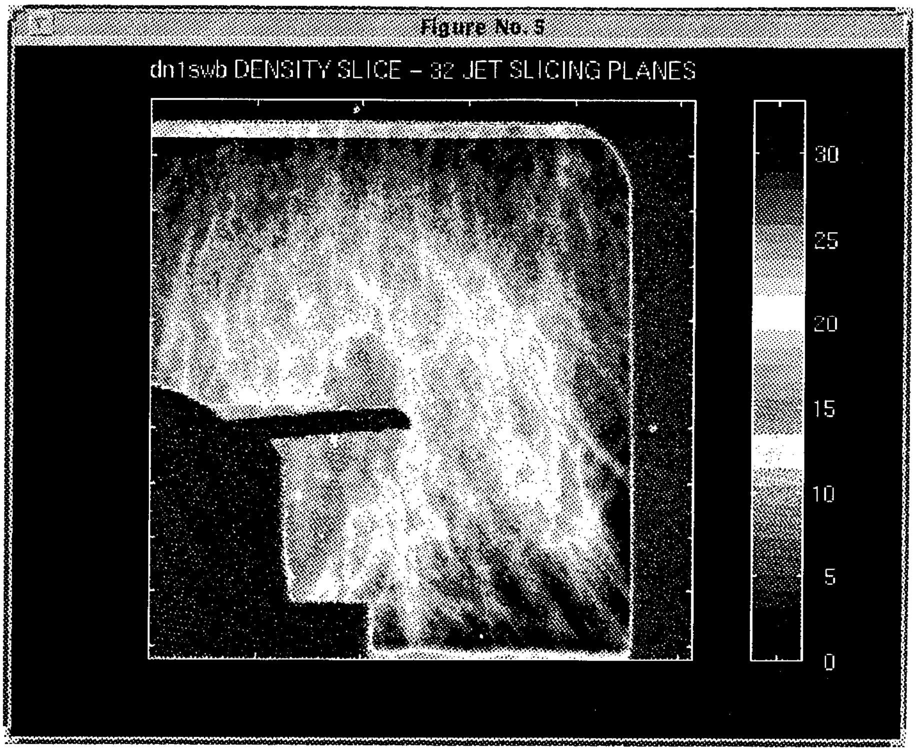


Figure 4.22 - Density slice

All density slices are annotated in **images read catalogue** according to the scheme described for a histogram, with the exception being that 'density slice' is used in place of 'histogram'.

4.4.7 *Texture Map.* A texture map submenu pictured in Figure 4.23 is displayed when this utility is chosen.

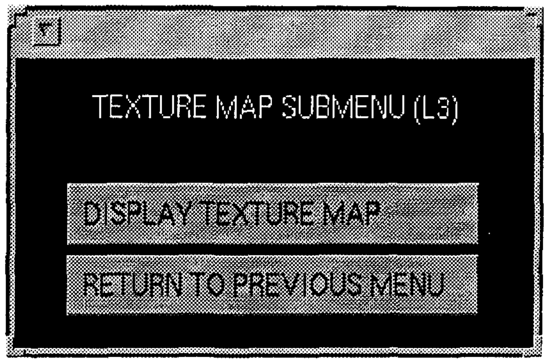


Figure 4.23 - Texture map submenu

Selection of **display texture map** invokes the prompt

```
Please enter the scale factor to be used to change the image
size [0.1]:
```

The size of the intensity matrix is reduced if scale factor < 1 and increased if scale factor > 1. Once a value is entered, an interpolation options menu pictured in Figure 4.24 is displayed.

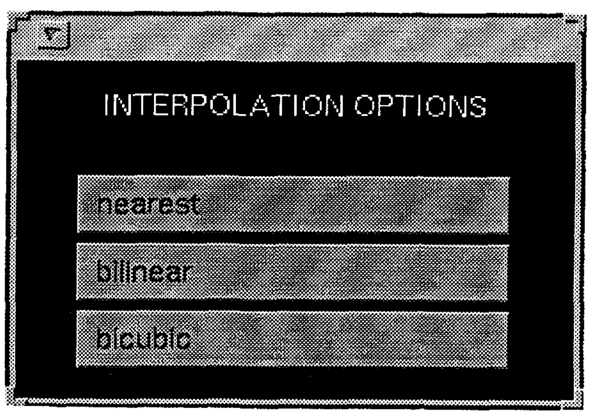


Figure 4.24 - Interpolation options submenu

For **nearest**, the magnitude of an interpolated pixel is the value of the closest pixel. For **bilinear**, the magnitude of an interpolated pixel is a combination of the values of the four closest pixels. And finally for **bicubic**, the magnitude of an interpolated pixel is a combination of the values of the sixteen closest pixels. The option chosen is assigned to interpolation, and this variable along with scale factor are used by MATLAB function *imresize* to change the size of the intensity matrix. MATLAB function *meshgrid* transforms the resized matrix dimensions to X and Y arrays used for 3-D plots. Resized matrix is also used by MATLAB function *mesh* which creates the 3-D intensity mesh plot. Lastly, the image display is placed below the mesh plot with MATLAB function *warp*. Figure 4.25 shows a sample texture map created using the default scale factor and bilinear interpolation.

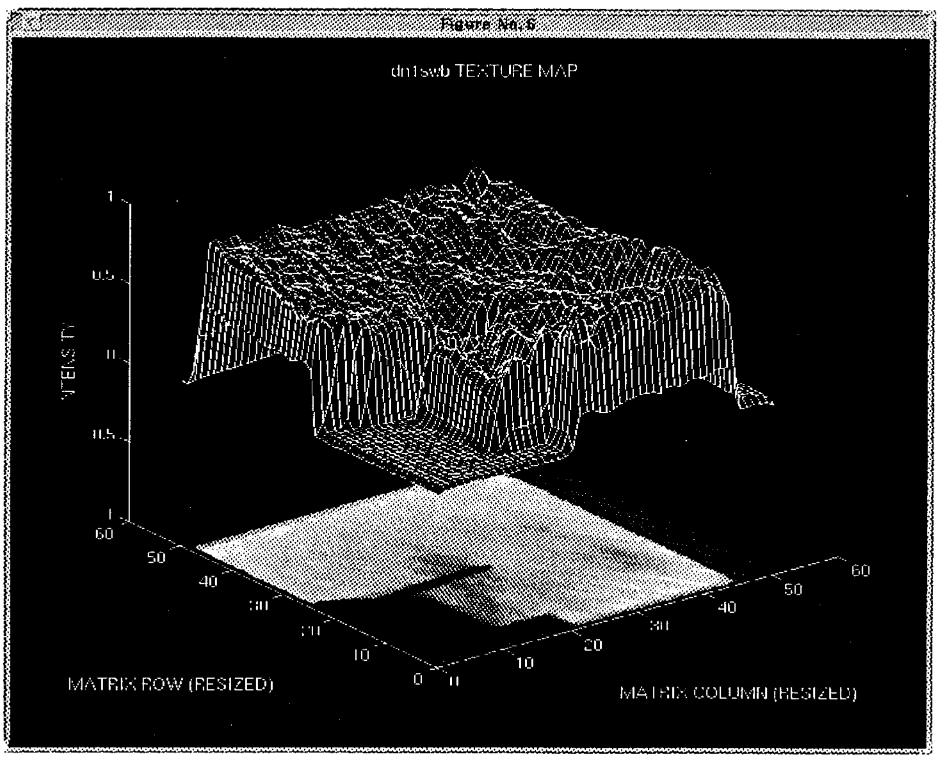


Figure 4.25 - Texture map

The creation of a texture map is recorded in images read catalogue in the same manner as for a histogram, with the only difference being that 'texture map' is used in place of 'histogram'.

4.4.8 *Delete Image.* This utility was described earlier but included as an option so that the user would not have to return to the main menu to delete an image.

4.5 *Image Processing*

Selection of this option executes the identical reloading process described in display utilities prior to the display of the image processing menu pictured in Figure 4.26. The image processing hierarchy allows single or multiple variable sized, rectangular regions of interest (ROI) to be extracted from an image. These regions of interest along with their assigned truth are saved in text files so that they can be processed by feature extraction routines either in the same or a different work session. The hierarchy also includes the Roberts, Prewitt, Sobel, and Marr-Hildreth edge detectors.

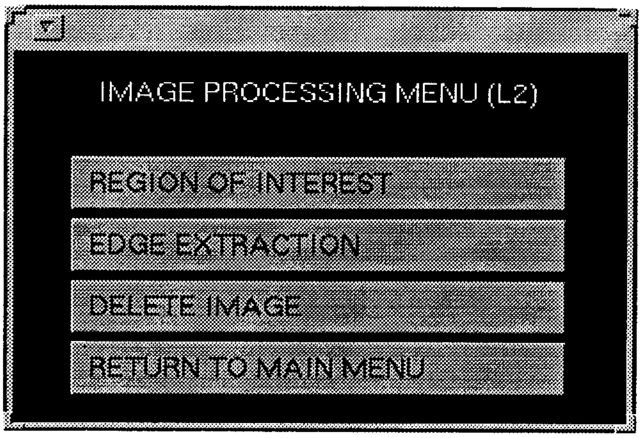


Figure 4.26 - Image processing menu

4.5.1 *Region of Interest.*

The submenu pictured in Figure 4.27 is displayed when **region of interest** is selected.

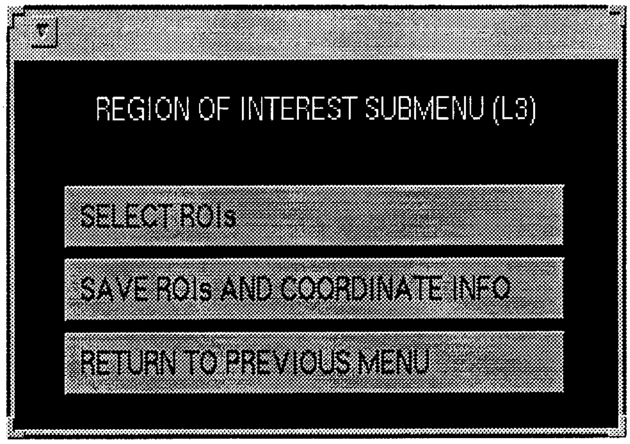


Figure 4.27 - Region of interest submenu

Choosing **select ROIs** invokes the two consecutive prompts

```
Please enter the vertical dimension (rows) of the ROI [16]:
Please enter the horizontal dimension (rows) of the ROI [16]:
```

The entry of **roi number of rows** and **roi number of columns** is followed by the instructional message

```
Use any mouse button to select the upper left
x and y coordinates of the regions of interest.  The
lower right x and y coordinates for each region of
interest will be calculated by the program.
After the coordinates are selected, keep the mouse
pointer in the image window and press return.
```

MATLAB function *ginput* is used to get either the single or multiple float coordinates and store them into column vectors, upper left x and upper left y. (MATLAB uses the matrix coordinate system for direct image matrix subscripting. In this system, the upper left corner of the image is the origin, and the rows increase downward while the columns increase to the right.) The coordinate(s) selected are echoed to the user as

```
upper left x =
  upper left x
      •
      •
      •

upper left y =
  upper left y
      •
      •
      •
```

The vectors are stored in a two column matrix called coordinates, where the first column is upper left x and the second column is upper left y. The carriage return marks the end of the selection process and brings about the prompt

```
Are the regions normal (N, n), benign (B, b) or malignant (M, m)
[default]?:
```

A truth other than a lowercase or uppercase 'n', 'normal', 'b', 'benign', 'm', or 'malignant' will cause the user to be prompted again after the error message

```
Please reenter your choice.
```

Coordinates is indexed through a row at a time to check if an upper left x or upper left y is beyond the image boundary. Roi number of rows is added to each rounded upper left x and roi number of columns to each rounded upper left y so as to check if the region of interest window crosses the right or bottom edge of the image. If any of these conditions hold, the invalid coordinate is discarded and the user receives the error message

```
Invalid --> upper_left_x = coordinates[index : index, 1 : 1]
            upper_left_y = coordinates[index : index, 2 : 2]
```

Each valid coordinate in coordinates is rounded and assigned to x upper left and y upper left respectively. Roi number of rows minus one is added to x upper left to obtain x lower right and roi number of columns minus one is added to y upper left to obtain y lower right. The four points are stored row wise in a coordinate matrix and the sequence is repeated until all valid points in coordinates have been processed. The points defining the region of interest window(s) and their truth, as well as the pixel values within the window(s) are echoed to the user in the format

```
                         ROI COORDINATES SELECTED
                         ************************
upper left x   upper left y   lower right x   lower right y   truth

upper left  x = coordinate matrix(index : index, 1 : 1)
upper left  y = coordinate matrix(index : index, 2 : 2)
lower right x = coordinate matrix(index : index, 3 : 3)
lower right y = coordinate matrix(index : index, 4 : 4)
truth = truth


>>>  region of interest index  <<<
index matrix(x upper left : x lower right, y upper left : y lower right)
```

For illustration purposes, a 4 x 4 window size was specified, followed by the selection of three points in dn1swb -- two within the mass traversed by the biopsy needle and one in the upper right corner of the figure beyond the image boundary. The feedback after the instructional message was

```
upper left x =

191.9370
203.9487
558.2947

upper left y =

252.7463
233.2273
 81.5792
```

The feedback after assigning a 'malignant' truth to the regions of interest was

```
Invalid --> upper left x = 558.294721   upper left y = 81.579179


                          ROI COORDINATES SELECTED
                          ************************
upper left x  upper left y  lower right x  lower right y  truth
192           253           195            256            malignant
204           233           207            236            malignant



>>>  region of interest 1  <<<

2839     2839     2839     2880
2799     2961     2920     2839
2961     2961     2839     2839
2961     2880     2799     2920


>>>  region of interest 2  <<<

2636     2677     2717     2758
2717     2717     2636     2677
2677     2717     2677     2677
2799     2677     2799     2717
```

An attempt to **save ROIs and coordinate info** prior to executing **select ROIs** generates the error message

```
Please select ROIs prior to choosing this option.
```

If **select ROIs** was run, the regions of interest and their associated parameters are saved in a dot ROI file while the points defining the region of interest window(s) and their truth are stored in a dot coordinate file.

The roi filename extension is created by concatenating '.roi' with an integer counter called number or ASCII files which keeps track of how many times **save ROIs and coordinate info** is executed for the current loaded image. The complete roi filename is generated by concatenating image read and the extension.

The coordinate filename is created in a similar manner with '.coordinate' used in place of '.roi'. Thus when **save ROIs and coordinate info** is selected, the user receives the feedback

```
ROIs saved to roi filename
Coordinates saved to coordinate filename
```

As an example, if this option was chosen just after executing **select ROIs** for dn1swb, the feedback would be

```
ROIs saved to dn1swb.roi1
Coordinates saved to dn1swb.coordinate1
```

If **select ROIs** and **save ROIs and coordinate info** were repeated for dn1swb, the feedback would be

```
ROIs saved to dn1swb.roi2
Coordinates saved to dn1swb.coordinate2
```

The region of interest file consists of a rectangular array of data. The first four rows are, respectively, tissue, roi rows, roi columns, and number of roi matrices. The fifth row to the end of the file is roi matrices. MATLAB function *zeros* and roi number of columns are used to pad the first four rows so that a valid matrix is formed. The element in [row 1, column 1] of tissue is the ASCII equivalent of the first character in the truth string which can be 'malignant', 'benign', or 'normal'. The element in [row 2, column 1] of roi rows, [row 3, column 1] of roi columns, and [row 4, column 1] of number of roi matrices are, respectively, roi number of rows, roi number of columns, and index. Index is a counter of the number of rows in coordinate matrix. Starting with the first row of coordinate matrix, the four points are retrieved and assigned to x upper left, y upper left, x lower right, and y lower right. These four parameters are used as fields in index matrix to retrieve the pixel values for the region of interest. MATLAB function *eval* is used to assign the submatrix to roi matrix that has an integer -- equivalent to the row being processed -- concatenated to it. The sequence is repeated until all rows in coordinate matrix have been processed. Each roi matrix is then stored top-down in roi matrices using once again MATLAB function *eval*. MATLAB command *save* and function *eval* are used to write tissue, roi rows, roi columns, number of roi matrices, and roi matrices to an ASCII file. A copy of dn1swb.roi1 is shown in Figure 4.28

```
1.0900000e+02   0.0000000e+00   0.0000000e+00    0.0000000e+00
4.0000000e+00   0.0000000e+00   0.0000000e+00    0.0000000e+00
4.0000000e+00   0.0000000e+00   0.0000000e+00    0.0000000e+00
2.0000000e+00   0.0000000e+00   0.0000000e+00    0.0000000e+00
2.8390000e+03   2.8390000e+03   2.8390000e+03    2.8800000e+03
2.7990000e+03   2.9610000e+03   2.9200000e+03    2.8390000e+03
2.9610000e+03   2.9610000e+03   2.8390000e+03    2.8390000e+03
2.9610000e+03   2.8800000e+03   2.7990000e+03    2.9200000e+03
2.6360000e+03   2.6770000e+03   2.7170000e+03    2.7580000e+03
2.7170000e+03   2.7170000e+03   2.6360000e+03    2.6670000e+03
2.6770000e+03   2.7170000e+03   2.6770000e+03    2.6670000e+03
2.7990000e+03   2.6770000e+03   2.7990000e+03    2.7170000e+03
```

Figure 4.28 - Region of interest file dn1swb.roi1

The creation of the coordinate file is much simpler, requiring only the print out of image read, coordinate filename, coordinate matrix, and truth using MATLAB file I/O functions *fopen* and *fprintf*. A copy of dn1swb.coordinate1 is shown in Figure 4.29.

```
Input image filename --> dn1swb

Output ROI coordinates filename --> dn1swb.coordinate1

>>>>>>>>>>>>>>>>>>>>>>>>>> ROI COORDINATES <<<<<<<<<<<<<<<<<<<<<<<<<<
upper left x   upper left y   lower right x   lower right y  truth
192            253            195             256            malignant
204            233            207             236            malignant
```

Figure 4.29 - Coordinate file dn1swb.coordinate1

4.5.2 *Edge Extraction.*

Choosing **edge extraction** displays the submenu pictured in Figure 4.30.

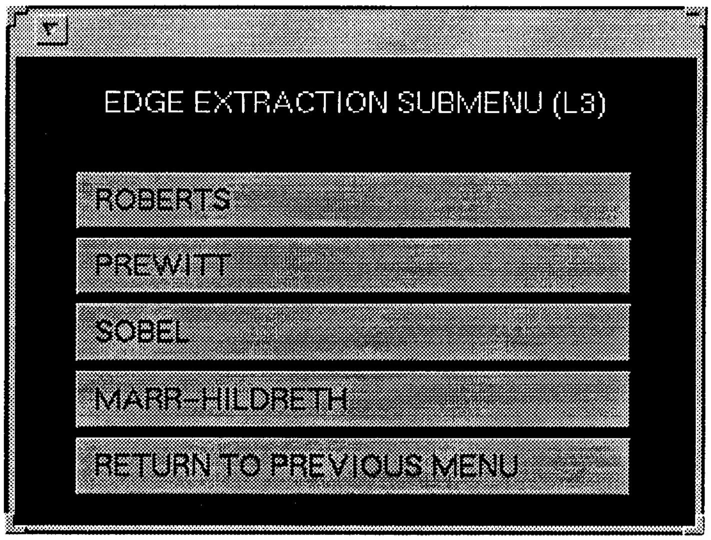


Figure 4.30 - Edge extraction submenu

When an edge selection is made, the name of the detector is assigned as a string to edge algorithm, followed by the prompt

```
Please enter the noise tolerance [0.01]:
```

All edges that do not exceed noise tolerance are disregarded. Selection of any one of the first three edge detectors displays a directionality factor options menu pictured in Figure 4.31. The options indicate the types of edges that should be searched for using a 1 x 2 directionality vector. Vertical, horizontal, and nondirectional edges are found by using, respectively, a directionality of [1 0], [0 1], and [1 1]. This menu is not displayed for the Marr-Hildreth algorithm since it does not use a directionality factor.

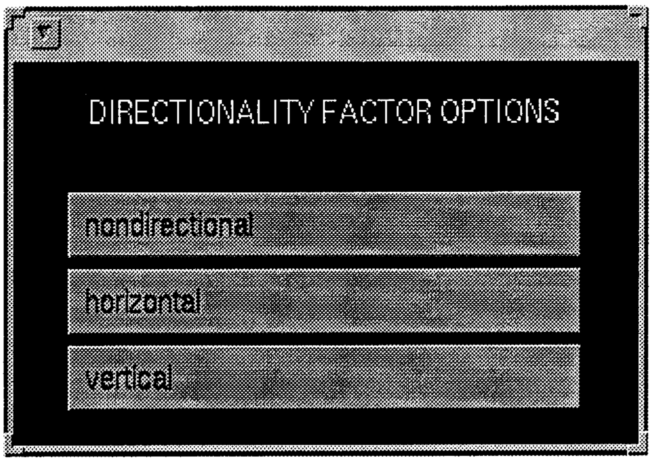


Figure 4.31 - Directionality factor options menu

The edge extraction is executed by calling MATLAB function *edge,* which uses intensity matrix, edge algorithm, noise tolerance, and directionality if applicable to return a binary image of the detected edges. Figure 4.32 shows a sample binary image created using the **Prewitt** edge algorithm, a 0.01 noise tolerance, and a nondirectional directionality.

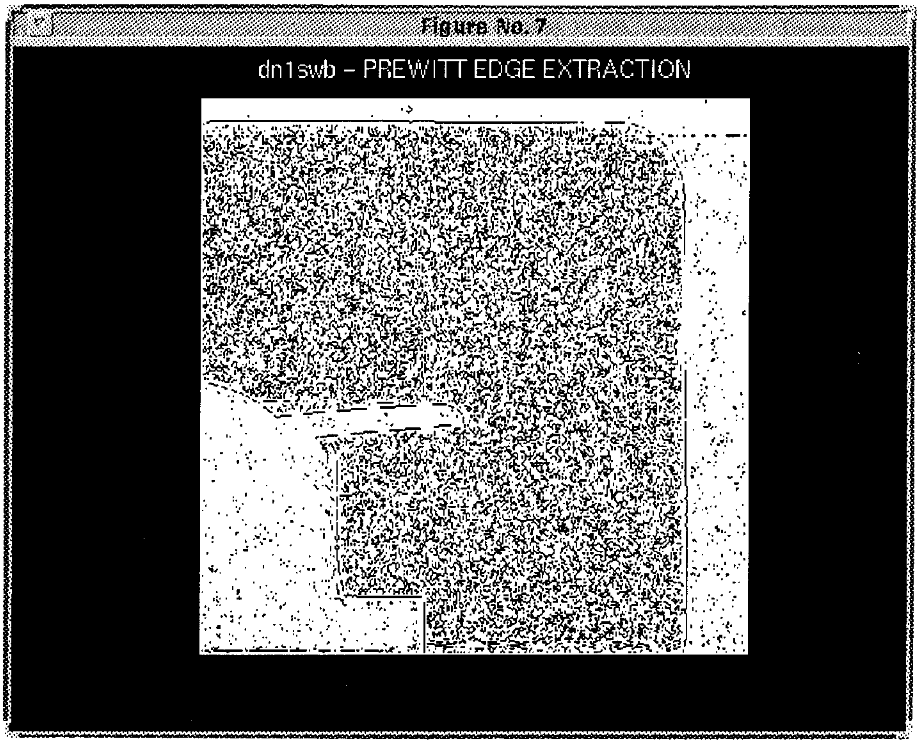


Figure 4.32 - Prewitt nondirectional edge extraction

The generation of the binary image is succeeded by the prompt

```
Complement the figure yes or no [default]?:
```

A complement answer other than a lowercase or uppercase 'n', 'no', 'y', or 'yes' will cause the user to be prompted again after the error message

```
Please reenter your choice.
```

An affirmative response complements the image as demonstrated in Figure 4.33.

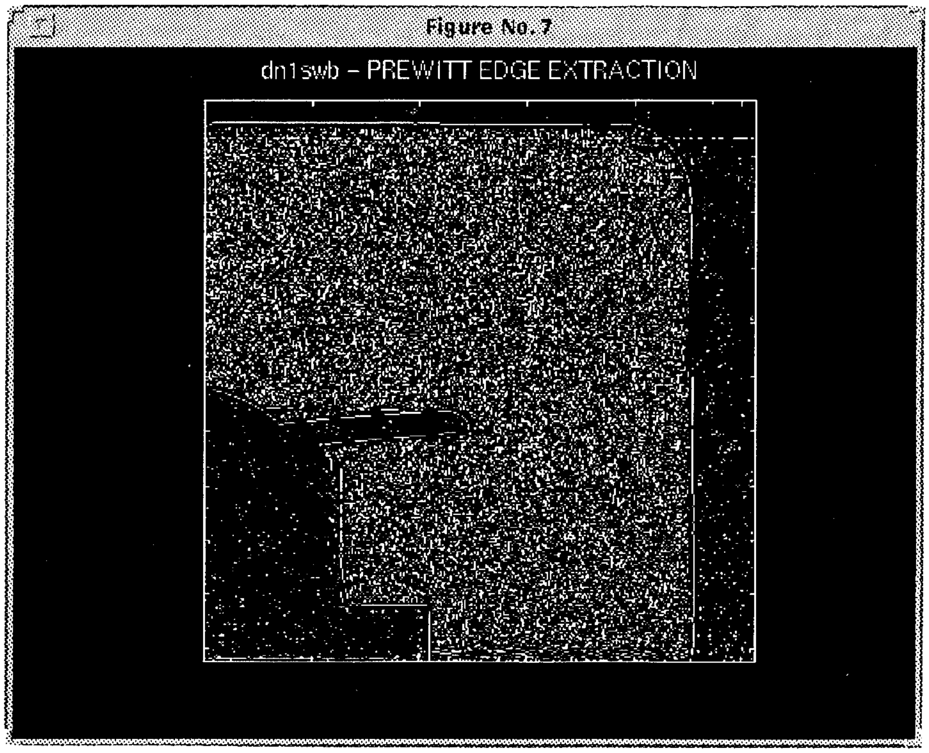


Figure 4.33 - Prewitt complemented edge extraction

All edge extractions are recorded in images read catalogue according to the method detailed for a histogram, with the exception being that 'edge extraction' is used in place of 'histogram'.

4.5.3 *Delete Image.* This utility was described earlier but included as an option so that the user would not have to return to the main menu to delete an image.

4.6 *Image Analysis*

Selection of this option displays the image analysis menu pictured in Figure 4.34. The image analysis hierarchy is where future feature extraction, segmentation, and classification routines should be added. **Segmentation** and **classification** are stubs that when selected print the message

```
Segmentation (Classification) routines to be added at a later date.
```

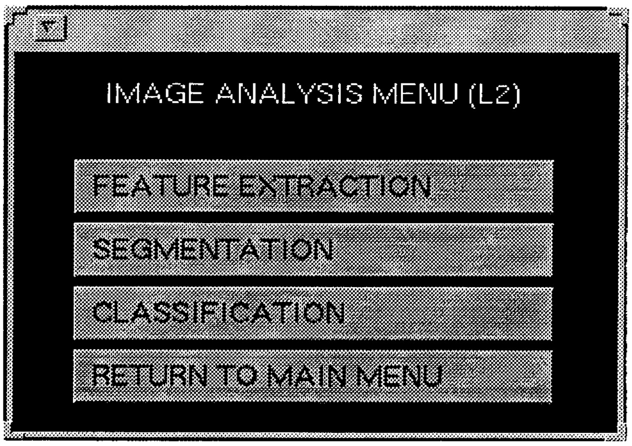


Figure 4.34 - Image analysis menu

Choosing **feature extraction** displays the submenu pictured in Figure 4.35. The feature extraction hierarchy is fairly well developed, allowing extracted region of interest files created in the image processing hierarchy to be processed by test stubs. These stubs serve as helpful guides for the addition of feature extraction routines in the future. The results from the stubs are printed in a LNKnet compatible input data file format.

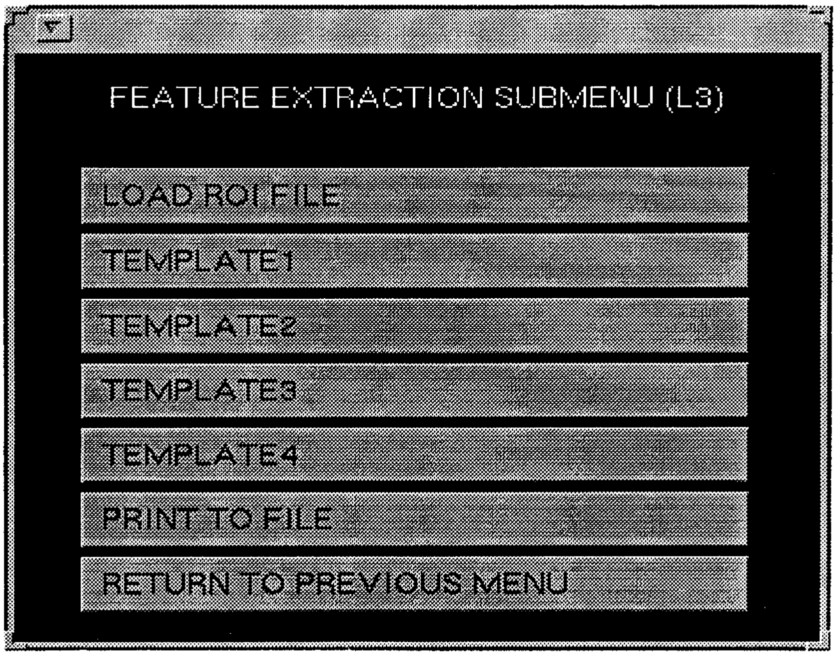


Figure 4.35 - Feature extraction submenu

4.6.1 *Load Region of Interest File.* Choosing **load ROI file** invokes the prompt

```
Please enter the name of the ROI file (e. g. dn1swb.roi1):
```

Leading and trailing blanks entered along with the filename are discarded. If an invalid filename is entered, the user is prompted again for the input file after the error message

```
File could not be located.
```

Once a valid input file is entered, MATLAB function *eval,* using arguments input file and MATLAB command *load,* reads the rectangular array of ASCII data into a matrix with the same name as the file, but not including the dot extension. The matrix is prepped for processing by repeating MATLAB function *eval,* this time using argument roi file, the filename without the dot extension. The element in [row 1, column 1] of the roi data matrix is assigned to tissue type text. If tissue type text is 'n' -- meaning normal, then tissue class is assigned a value of zero. If tissue type text is 'b' -- meaning benign, then tissue class is assigned a value of one. If tissue type text is neither of the first two characters, then it must be 'm' -- meaning malignant, and tissue class is therefore assigned a value of two. Tissue class is converted into string tissue value so as to be in a compatible format for the print to file utility. The final part of the algorithm completes the partition of roi data matrix, assigning element [row 2, column 1] to roi number of rows, element [row 3, column 1] to roi number of columns, element [row 4, column 1] to number of rois, and all elements from row 5 on to rois.

For illustration purposes, assume the default dn1swb.roi1 has been loaded. Then the variables discussed previously would have the following values:

```
input file >>> dn1swb.roi1
roi file >>> dn1swb
tissue type text >>> 109 {ASCII equivalent of 'm'}
tissue class >>> 2
tissue value >>> '2'
roi number of rows >>> 4
roi number of columns >>> 4
number of rois >>> 2
rois >>> 2839    2839    2839    2880
         2799    2961    2920    2839
         2961    2961    2839    2839
         2961    2880    2799    2920
         2636    2677    2717    2758
         2717    2717    2636    2677
         2677    2717    2677    2677
         2799    2677    2799    2717
```

4.6.2 *Templates.*

Attempting to execute a template prior to loading a ROI file generates the error message

```
Please load a ROI file prior to selecting this option.
```

The templates are functional test stubs. Thus if rois from dn1swb.roi1 is used, selection of **template1** sums the elements of both rois and returns

```
feature1_vector = 46076     43275
```

Template2 finds the maximum element for each ROI, **template3** finds the minimum element for each ROI, and **template4** calculates the mean value for each ROI. The selection of these templates returns

```
feature2_vector = 2961     2799
feature3_vector = 2799     2636
feature4_vector = 1.0e+03 *
                  2.8798   2.7047
```

The templates serve as a framework for the effortless addition of feature extraction routines. The only steps required to interface a feature extraction routine are to (1) rename the template call in feature_extraction.m to the routine filename without the dot extension, (2) update the menu call in feature_extraction.m, and (3) use the same arguments passed and returned to a template. These steps are included as comments in the appropriate files. For consistency purposes, the added routines should be able to handle multiple regions of interest. An alternate method for partitioning rois is included as a comment in each template.

4.6.3 *Print to File.*

For this research effort, the classifier that had to be supported was a package developed at the MIT Lincoln Laboratory called LNKnet. LNKnet can read files in ASCII format. Comments can be included in a file as long as they are preceded with a percent sign. The implementation chosen for this utility was to print the output of the test stubs in columnar format with each column representing a particular test stub and the last column reserved for which image, or more specifically which region of interest file was run through the various test stubs. A blank entry is printed for each test stub that is not run.

Selection of **print to file** prior to running a template produces the error message

```
Please run a feature extraction algorithm prior to selecting this
option.
```

If at least one template has been executed, then the print to file submenu pictured in Figure 4.36 is displayed.

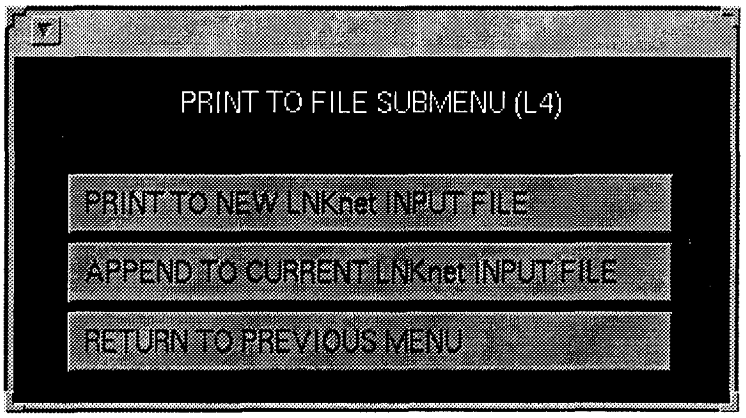


Figure 4.36 - Print to file submenu

When **print to new LNKnet input file** is chosen, the user is asked

```
Please enter the name of the LNKnet input file:
```

Feature1 vector, feature2 vector, feature3 vector, and feature4 vector are converted to strings vector1, vector2, vector3, and vector4 respectively. Tissue value, vector1, vector2, vector3, vector4, and input file are then printed to the specified LNKnet input file, followed by the feedback

```
Feature extraction results saved to LNKnet input file
```

Figure 4.37 is a copy of LNKnet input file 'LNKnet', showing the print out of these variables.

```
2    4.608e+04    2961    2799    2880    % dn1swb.roi1
2    4.328e+04    2799    2636    2705    % dn1swb.roi1
0    5929         406     325     370.6   % dn1swb.roi2
2    5.684e+04    2359    2142    2273    % dst1swb.roi1
```

Figure 4.37 - LNKnet file

The rest of the entries in 'LNKnet' were obtained by repeating the loading and test stub selection process as described previously for roi files dn1swb.roi2 and dst1swb.roi1. The print option used was **append to current LNKnet input file** which when executed provides the feedback

```
Feature extraction results appended to LNKnet input file
```

An attempt to append prior to specifying a LNKnet input file results in the error message

```
PRINT TO NEW LNKnet INPUT FILE must be selected first.
```

4.7 *Quit Program*

This option is a part of the script file and deletes any image prior to termination of the program. The algorithm calculates the number of records in images read catalogue, then reads the first entry for each record. If the retrieved string is not a blank string, then the corresponding image figure is deleted using the MATLAB command *delete* (h) where h is the handle of the image.

4.8 *Summary*

This chapter described the implementation of the program along with the functionality and operation of each utility. The discussion was tailored to simulate a work session, using photographs of function features, interactive dialogue displays, and generated output files. In the next, and final chapter, the program as a whole is evaluated using a quality model that provides an indication of its potential value as a support environment for research conducted in the field of computer-aided diagnosis.

V. Conclusion

In this final chapter, the *Mammogram Image Processing Program* is evaluated nonstatistically using McCall's *software quality factors*. In closing, a direction is suggested for future research conducted in the field of computer-aided diagnosis using this program.

5.1 *Assessment*

Ideally, the quality of the environment would be measured by a third party using a quantitative standard metric. Although a self-assessment using a model is likely to be skewed, in relative terms it is likely to be more objective than an evaluation conducted in an ad hoc manner that tends to highlight the strengths while glossing over or even omitting the weaknesses. Actually, an assessment conducted by a writer, who makes an honest effort to be objective, can be advantageous since no one is as familiar with the program. In regards to using a quantitative metric, the reader should note that quality is a relative, not an absolute quantity [Gillies, 1992]. Hence, even when metrics are used, quality is not measured, but rather some manifestation of quality [Pressman, 1987]. The complicating factor is in defining exact relationships between the measured parameters and the quality of the software [Pressman, 1987].

5.1.1 *Maintainability*. MATLAB is a high-level programming language with its own compiler and debugger. A number of methods as listed in [The MathWorks, Inc., 1992a] can be used in parallel with the compiler for quickly isolating faults including: (1) removing semicolons from the end of statements to display the value of variables during execution of an M-file, (2) debugging sections of code by cutting from a text file and pasting into the MATLAB workspace, (3) commenting out leading function declarations so that the M-file can be run as a script, (4) viewing commands in M-files during execution with the *echo* command, and (5) using the *keyboard* command to stop execution of an M-file and give control to the keyboard so as to examine and change variables. These techniques were used for this research effort, making use of the debugger unnecessary. Maintainability was designed into the program through the use of modularity, and code that is self-documenting and well-commented.

5.1.2 *Flexibility.* Any routine can be modified by simply using a text editor. In addition, the developer has access to the source code for many of the MATLAB routines and therefore can tailor them to suit the needs of the application. The command *which* can be used to display the full pathname for any function. As a suggestion, the filenames and functions should be renamed if they are copied.

5.1.3 *Testability.* Although debugging and testing are not synonymous, the debugging techniques described for the maintainability factor can be used to test the program. The program can be subjected to either white box or black box testing. For this research effort the test adequacy criteria employed was branch coverage where every branch in each function, that is M-file, was executed at least once. Upon completion of this unit level testing, routines that comprised a subsystem were subjected to bottom-up integration testing. Finally, the entire program, that is system, was tested by executing the functions in random sequences.

5.1.4 *Portability.* MATLAB version 4.2 (UNIX) and the MATLAB Image Processing Toolbox can run on the following platforms [The MathWorks, Inc., 1992a]:

- Sun-4/SPARC
  - Sun-4/SPARC-based workstation
  - SunOS 4.1 or above
  - OpenWindows version 3.0 or X Windows (X11R4 or X11R5)

- DEC RISC
  - DECstation 2100, 3100, 5000, or DECsystem 54xx, 58xx
  - ULTRIX operating system version 4.2 or above
  - DECwindows or X Windows (X11R4 or X11R5)
- HP 9000 Series 300 and 400
  - HP 9000 Series 300 or 400 workstation
  - 68881 or 68882 math coprocessor
  - HP-UX 8.0
  - X Windows (X11R4 or X11R5)
- HP 9000 Series 700
  - HP 9000 Series 700 workstation
  - HP-UX 8.0 or above
  - X Windows (X11R4 or X11R5)
- IBM RS/6000
  - IBM RS/6000 workstation
  - AIX 3.2 or above
  - X Windows (X11R4 or X11R5)

All platforms require 8 Mb of hard disk space and 8 Mb of random access memory, except the Sun-4/SPARC which requires 12 Mb of the latter. The program was developed on a Sun SPARC workstation and has not been tested on any other platform.

5.1.5 *Reusability.* The *Mammogram Image Processing Program* is actually a generic environment that can be used for other image processing applications. The program can load raw file formats of varying dimensions and gray levels, and if desired, can easily be tailored to read GIF, TIFF, HDF, BMP, XWD, PCX, and of course MAT-files [Thompson and Shure, 1993]. Special formats can be read by linking MATLAB to C or Fortran routines [Thompson and Shure, 1993].

Depending upon the application, the routines may have to be slightly adapted prior to reuse since they were written for a menu-based, mouse-driven, and keyboard interactive environment. The reuse of lower-level components as specific implementation techniques of a unit or higher-level components as the system level design is more realistic.

5.1.6 *Interoperability.* MATLAB can be interfaced with Mathematica, a high-level programming language containing extensive numerical, symbolic, and graphic capabilities [Kaplan and others, 1993] [Wolfram, 1991]. The interested reader should look at item numbers 0205-951, 0206-143, and 0206-200 located in the Enhancements/Interfacing/Matlab directory of the Internet site mathsource.wri.com maintained by the developers of Mathematica. An extensive number of application packages and notebooks are also available from this site. Obtain ItemLocations from the pub directory for a directory listing of these applications.

A significant feature of MATLAB is its ability to call, with MEX-files, C and Fortran routines [The MathWorks, Inc., 1993a]. This permits the reuse of existing image processing routines written in C (although the reuse of poorly documented, complex mathematical image processing algorithms is not recommended), and the development of algorithms that might otherwise be awkward or extremely difficult to implement in MATLAB. In addition to [The MathWorks, Inc., 1993a], C MEX-file function examples and information on how to write a C++ MEX-file function are available by downloading technical notes mxeng3.txt, mxeng4.txt, and mxeng5.txt located in the pub/tech-support/tech-notes directory of the ftp site ftp.mathworks.com. These technical notes are included in Appendix B as a convenience to the interested developer. Many C MEX-file examples can also be found in the $MATLAB/exter/src directory.

Although this factor was not completely investigated, the existence of other systems that can be interfaced with MATLAB is likely.

5.1.7 *Correctness.* The requirements were established in accordance that the environment would not require additions or modifications as physicians and engineers began to use it to research and develop detection and diagnostic algorithms. All of the requirements were met.

5.1.8 *Reliability.* As described for the usability factor, the program includes a number of error control mechanisms to prevent failure and thus the loss of a long and productive work session. These mechanisms include but are not limited to handling errors raised by attempting to load files, delete images, or select figures that do not exist; entering text for integer entries; typing leading and trailing blanks on text entries; trying to reload as the current image either loaded images that do not exist or unloaded images; turning on (off) the reference grid for a loaded image already in the on (off) state; and selecting regions of interest crossing or completely beyond the borders of the image.

5.1.9 *Efficiency.* The current size of the program is 180 Kb. Use of random access memory was minimized by modularizing the program into functions and by allowing only one image to be current at any given time. In the context of this research effort, the importance of speed is debatable considering the application is not time-critical, and the program itself is not intended for production and deployment. Optimization of detection and or diagnostic algorithms may be relevant if they have proven to be extremely accurate and are intended for use within the medical community, especially within understaffed radiology departments that must read a large volume of mammograms each day. To increase execution speed, [The MathWorks, Inc., 1992a] recommends vectorizing iterative loops or at least preallocating vectors that store output.

5.1.10 *Integrity.* This factor is not applicable considering the program was developed for research and not for commercial purposes. In fact, the source code is included in Appendix A. Hopefully, by making this relatively inexpensive environment that can be run on a number of common platforms available, other groups will join the effort to develop computerized schemes that will improve the detection and diagnosis of breast cancer, a major public health problem.

5.1.11 *Usability.* To make operation of the program as simple as possible, default values, error control mechanisms, action reversal, informative feedback, shortcut options, command consistency, and on-line help were designed into the program as detailed in the previous chapter. The program's usability was tested by a number of operators who provided constructive comments that were incorporated. This refinement process of testing and modification was repeated as many times as possible.

[The MathWorks, Inc., 1993b] describes how to build a graphical user interface for an application created within MATLAB. The types of controls that can be built include push buttons, check boxes, radio buttons, sliders, pop-up menus, static text, editable text, and frames. By providing a standard set of commands to create these controls and not requiring the developer to be familiar with Open Software Foundation (OSF) Motif programming in the X Window System environment, MATLAB has taken the complexity out of this otherwise most intimidating task. The reader should note that the use of a graphical user interface does not always equate to increased usability.

5.2 *Final Comments*

The *Mammogram Image Processing Program* provides the foundation for the effortless future addition of innovative, modified, and previously developed segmentation, feature extraction, and classification routines. Previously developed routines should be well-documented, and perhaps restricted to algorithms that have not been applied to mammography, or are being used as preprocessing or postprocessing steps. The performance of these computer-aided diagnosis schemes should be evaluated using receiver operating characteristic (ROC) analysis and a large, comprehensive, and preferably common mammogram database.

image_processing.m

```
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  image_processing.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Script file for Mammogram Image Processing Program that displays
%%                the main menu.
%%                Calls --> database_menu.m
%%                      --> delete_image.m
%%                      --> du_menu.m {display utilities}
%%                      --> ip_menu.m {image processing}
%%                      --> ia_menu.m {image analysis}
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%########################################################################################%
%#####################                                              #####################%
%#####################                                              #####################%
%#####################      MAMMOGRAM IMAGE PROCESSING PROGRAM      #####################%
%#####################                                              #####################%
%#####################                                              #####################%
%########################################################################################%


clear all
% clf reset

window_handle = gcf;
delete(window_handle);


fprintf('\n')
fprintf('\n')
fprintf('****  Welcome to the MAMMOGRAM IMAGE PROCESSING PROGRAM  ****\n')
fprintf('****                     Version 1.0                     ****\n')
fprintf('****            Written by 1Lt John L. Kelley            ****\n')
fprintf('\n')
fprintf('\n')


bye = 'no';
colormap_flag = 'unset';
num_loaded_images = 0;
num_unloaded_images = 0;
new_image_read = ' ';
LNKnet_file_loaded = 'false';
image_loaded = 'false';


while (strcmp(bye, 'no'))
  choice = menu('IMAGE PROCESSING MAIN MENU (L1)', ...
                'DATABASE', ...
                'DELETE IMAGE', ...
                'DISPLAY UTILITIES MENU', ...
                'IMAGE PROCESSING MENU', ...
                'IMAGE ANALYSIS MENU', ...
                'QUIT PROGRAM');
```

image_processing.m

```
%**************************************************************************************%
%**************************************************************************************%
%*********************                                           *********************%
%*********************              DATABASE                     *********************%
%*********************                                           *********************%
%**************************************************************************************%
%**************************************************************************************%


  if choice == 1
    [index_matrix, ...
     intensity_matrix, ...
     vertical_dimension, ...
     horizontal_dimension, ...
     gray_level, ...
     colormap_flag, ...
     new_map, ...
     image_read, ...
     images_read_catalogue, ...
     database_flag, ...
     num_loaded_images, ...
     image_loaded] = database_menu(images_read_catalogue, ...
                                   colormap_flag, ...
                                   new_map, ...
                                   num_loaded_images, ...
                                   image_loaded);

%**************************************************************************************%
%**************************************************************************************%
%*********************                                           *********************%
%*********************            DELETE IMAGE                   *********************%
%*********************                                           *********************%
%**************************************************************************************%
%**************************************************************************************%


  elseif choice == 2
    [images_read_catalogue, ...
     num_loaded_images, ...
     num_unloaded_images] = delete_image(images_read_catalogue, ...
                                         num_loaded_images, ...
                                         num_unloaded_images);

%**************************************************************************************%
%**************************************************************************************%
%*********************                                           *********************%
%*********************          DISPLAY UTILITIES                *********************%
%*********************                                           *********************%
%**************************************************************************************%
%**************************************************************************************%


  elseif choice == 3
    [images_read_catalogue, ...
     image_read, ...
     num_loaded_images, ...
     num_unloaded_images] = du_menu(index_matrix, ...
                                    intensity_matrix, ...
                                    vertical_dimension, ...
                                    horizontal_dimension, ...
                                    gray_level, ...
                                    new_map, ...
                                    image_read, ...
                                    images_read_catalogue, ...
                                    database_flag, ...
                                    num_loaded_images, ...
                                    num_unloaded_images, ...
                                    image_loaded);
```

```
%*****************************************************************************************%
%*****************************************************************************************%
%*********************                                              *********************%
%*********************              IMAGE PROCESSING                *********************%
%*********************                                              *********************%
%*****************************************************************************************%
%*****************************************************************************************%


  elseif choice == 4
    [images_read_catalogue, ...
     image_read, ...
     num_loaded_images, ...
     num_unloaded_images] = ip_menu(index_matrix, ...
                                    intensity_matrix, ...
                                    vertical_dimension, ...
                                    horizontal_dimension, ...
                                    gray_level, ...
                                    new_map, ...
                                    image_read, ...
                                    images_read_catalogue, ...
                                    database_flag, ...
                                    num_loaded_images, ...
                                    num_unloaded_images, ...
                                    image_loaded);

%*****************************************************************************************%
%*****************************************************************************************%
%*********************                                              *********************%
%*********************               IMAGE ANALYSIS                 *********************%
%*********************                                              *********************%
%*****************************************************************************************%
%*****************************************************************************************%


  elseif choice == 5
    [LNKnet_input_file, ...
     LNKnet_file_loaded, ...
     LNKnet_fid] = ia_menu(LNKnet_input_file, ...
                           LNKnet_file_loaded, ...
                           LNKnet_fid);

%*****************************************************************************************%
%*****************************************************************************************%
%*********************                                              *********************%
%*********************                QUIT PROGRAM                  *********************%
%*********************                                              *********************%
%*****************************************************************************************%
%*****************************************************************************************%


  elseif choice == 6

    %%>>>>>  Loop up to number of records in catalogue.

    for index = 1 : 1 : (size(images_read_catalogue, 1) / 7)
      offset = index + (6 * (index - 1));
      columns = size(images_read_catalogue, 2);

      %%>>>>>  Check to see if first entry in each record is a blank string.

      if ~strcmp(images_read_catalogue(offset, : ), blanks(columns))

        %%>>>>>  Delete corresponding image figure if first entry is not a blank string.

        delete(index);
      end

    end
    bye = 'yes';

  end
end

%##########################################################################################%
%######################                                              ######################%
%######################                                              ######################%
%######################                     END                      ######################%
%######################                                              ######################%
%######################                                              ######################%
%##########################################################################################%
```

database_menu.m

```
function [index_matrix, ...
         intensity_matrix, ...
         vertical_dimension, ...
         horizontal_dimension, ...
         gray_level, ...
         colormap_flag, ...
         new_map, ...
         image_read, ...
         images_read_catalogue, ...
         database_flag, ...
         num_loaded_images, ...
         image_loaded] = database_menu(images_read_catalogue, ...
                                       colormap_flag, ...
                                       new_map, ...
                                       num_loaded_images, ...
                                       image_loaded)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  database_menu.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Displays the database menu and sets according to the option selected
%%                a database flag that is used in load_image.m and reload_image.m.
%%                Calls --> load_image.m
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%**************************************************************************************%
%**************************************************************************************%
%*********************                                             *********************%
%*********************                 DATABASE                    *********************%
%*********************                                             *********************%
%**************************************************************************************%
%**************************************************************************************%


  quit_database = 'no';
  while (strcmp(quit_database, 'no'))
    database_option = menu('DATABASE MENU (L2)', ...
                           'LORAD DSM', ...
                           'CINCINNATI', ...
                           'RETURN TO MAIN MENU');
```

```
%>>>>>>>>>>                                                                 <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  LORAD DSM                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                 <<<<<<<<<<%


    if database_option == 1
      database_flag = 'lorad dsm';
      [index_matrix, ...
       intensity_matrix, ...
       vertical_dimension, ...
       horizontal_dimension, ...
       gray_level, ...
       colormap_flag, ...
       new_map, ...
       image_read, ...
       images_read_catalogue, ...
       num_loaded_images, ...
       image_loaded] = load_image(images_read_catalogue, ...
                                 colormap_flag, ...
                                 new_map, ...
                                 database_flag, ...
                                 num_loaded_images, ...
                                 image_loaded);


%>>>>>>>>>>                                                                 <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  CINCINNATI                 <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                 <<<<<<<<<<%


    elseif database_option == 2
      database_flag = 'cincinnati';
      [index_matrix, ...
       intensity_matrix, ...
       vertical_dimension, ...
       horizontal_dimension, ...
       gray_level, ...
       colormap_flag, ...
       new_map, ...
       image_read, ...
       images_read_catalogue, ...
       num_loaded_images, ...
       image_loaded] = load_image(images_read_catalogue, ...
                                 colormap_flag, ...
                                 new_map, ...
                                 database_flag, ...
                                 num_loaded_images, ...
                                 image_loaded);


%>>>>>>>>>>                                                                 <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             RETURN TO MAIN MENU             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                 <<<<<<<<<<%


    elseif database_option == 3
      quit_database = 'yes';
    end
  end

%*************************************************************************************%
%*************************************************************************************%
%********************                                            ********************%
%********************                     END                    ********************%
%********************                                            ********************%
%*************************************************************************************%
%*************************************************************************************%
```

load_image.m

```
function [index_matrix, ...
          intensity_matrix, ...
          vertical_dimension, ...
          horizontal_dimension, ...
          gray_level, ...
          colormap_flag, ...
          new_map, ...
          image_read, ...
          images_read_catalogue, ...
          num_loaded_images, ...
          image_loaded] = load_image(images_read_catalogue, ...
                                     colormap_flag, ...
                                     new_map, ...
                                     database_flag, ...
                                     num_loaded_images, ...
                                     image_loaded)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  load_image.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Loads binary mammographic images of varying dimensions, gray levels
%%                and bits/pixel.  The filename and axes with tick marks in increments
%%                of fifty are displayed along with the image.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                            <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>               LOAD IMAGE               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                            <<<<<<<<<<%


  %%>>>>>  Prompt until valid filename entered.

  fid = -1;
  while (fid == -1)
    image_read = input('Please enter the name of the image [dn1swb]: ', 's');
    if isempty(image_read)
      image_read = 'dn1swb';
    end
    image_read = (image_read(image_read ~= ' '));
    fid = fopen(image_read, 'r');
    if (fid == -1)
      fprintf('\n')
      fprintf('File could not be located.\n')
    end
  end
  vertical_dimension = ...
    input ('Please enter the vertical dimension (rows) of the image [512]: ');
  if isempty(vertical_dimension)
    vertical_dimension = 512;
  end
  horizontal_dimension = ...
    input ('Please enter the horizontal dimension (columns) of the image [512]: ');
  if isempty(horizontal_dimension)
    horizontal_dimension = 512;
  end
  gray_level = ...
    input ('Please enter the gray level of the image [4096]: ');
  if isempty(gray_level)
    gray_level = 4096;
  end
```

```
num_precision_choice = menu('NUMERIC PRECISION OPTIONS', ...
                            'char  character  8 bits', ...
                            'schar  signed character  8 bits', ...
                            'uchar  unsigned character  8 bits', ...
                            'short  integer  16 bits', ...
                            'ushort  unsigned integer  16 bits', ...
                            'int  integer  16 or 32 bits', ...
                            'uint  unsigned integer  16 or 32 bits', ...
                            'long  integer  32 bits', ...
                            'ulong  unsigned integer  32 bits', ...
                            'float  floating point value  32 bits', ...
                            'float32  floating point value  32 bits', ...
                            'float64  floating point value  64 bits', ...
                            'double  long floating point value  64 bits', ...
                            'intN  signed integer  N = 1..64 bits', ...
                            'uintN  unsigned integer  N = 1..64 bits');
if num_precision_choice == 1
  num_precision = 'char';
elseif num_precision_choice == 2
  num_precision = 'schar';
elseif num_precision_choice == 3
  num_precision = 'uchar';
elseif num_precision_choice == 4
  num_precision = 'short';
elseif num_precision_choice == 5
  num_precision = 'ushort';
elseif num_precision_choice == 6
  num_precision = 'int';
elseif num_precision_choice == 7
  num_precision = 'uint';
elseif num_precision_choice == 8
  num_precision = 'long';
elseif num_precision_choice == 9
  num_precision = 'ulong';
elseif num_precision_choice == 10
  num_precision = 'float';
elseif num_precision_choice == 11
  num_precision = 'float32';
elseif num_precision_choice == 12
  num_precision = 'float64';
elseif num_precision_choice == 13
  num_precision = 'double';
elseif num_precision_choice == 14
  num_precision = 'intN';
elseif num_precision_choice == 15
  num_precision = 'uintN';
end
```

load_image.m

```
%%>>>>>  Store entered parameters into a record.

image_read_record = str2mat(image_read);
vertical_size = int2str(vertical_dimension);
image_read_record = str2mat(image_read_record, vertical_size);
horizontal_size = int2str(horizontal_dimension);
image_read_record = str2mat(image_read_record, horizontal_size);
gray_scale = int2str(gray_level);
image_read_record = str2mat(image_read_record, gray_scale);
image_read_record = str2mat(image_read_record, num_precision);
image_read_record = str2mat(image_read_record, 'loaded');
image_read_record = str2mat(image_read_record, ' ');

%%>>>>>  Insert record into the catalogue.

if strcmp(image_loaded, 'false')

  %%>>>>>  If an image was not loaded previously, create a catalogue having the same dimensions
  %%>>>>>  as the record then assign the record to the catalogue.

  images_read_catalogue = str2mat(image_read_record);
  image_loaded = 'true';
else
  empty_slot = 'false';

  %%>>>>>  Calculate number of columns in the catalogue.

  columns = size(images_read_catalogue, 2);

  %%>>>>>  Calculate number of records in the catalogue.

  catalogue_size = size(images_read_catalogue, 1);

  %%>>>>>  Use linear search to check catalogue for a blank record.

  for index = 1 : 1 : catalogue_size / 7
    offset = index + (6 * (index - 1));

    %%>>>>>  Check first field in each record to see if it is blank.

    if strcmp(images_read_catalogue(offset, : ), blanks(columns))

      %%>>>>>  Insert record into catalogue.

      if offset == 1
        images_read_catalogue = str2mat(image_read_record, ...
          images_read_catalogue(8 : catalogue_size,  : ));
        empty_slot = 'true';
        break
      else
        images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
          image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
          empty_slot = 'true';
        break
      end
    end
  end
  if strcmp(empty_slot, 'false')

    %%>>>>>  Append record to catalogue.

    images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
  end
end
num_loaded_images = num_loaded_images + 1;
```

load_image.m

```
%%>>>>> Open image file for reading.

fid = fopen(image_read, 'r');

%%>>>>> Read image file binary data.

[raw_matrix, count] = ...
  fread(fid, [vertical_dimension, horizontal_dimension], num_precision);
if strcmp(database_flag, 'lorad dsm')

  %%>>>>> Raw data transpose.

  raw_matrix = raw_matrix';
end

intensity_matrix = mat2gray(raw_matrix);
index_matrix = gray2ind(intensity_matrix, gray_level);

if strcmp(colormap_flag, 'unset')
  colormap(gray(gray_level))
  if strcmp(database_flag, 'lorad dsm')

    %%>>>>> Color map reversal to match LORAD DSM display.

    map = colormap;
    [map_rows, map_columns] = size(map);
    new_map = map(map_rows : -1 : 1, :);
    colormap(new_map)
  else
    new_map = colormap;
    colormap(new_map)
  end

  %%>>>>> Display image.

  imshow(index_matrix)

  %%>>>>> Display filename.

  title(image_read);

  %%>>>>> Display axes with tick marks in increments of fifty.

  x_spacing_and_label = 50 : 50 : horizontal_dimension;
  y_spacing_and_label = 50 : 50 : vertical_dimension;
  set(gca, 'XTick', [x_spacing_and_label])
  set(gca, 'YTick', [y_spacing_and_label])
  set(gca, 'XTickLabels', [x_spacing_and_label])
  set(gca, 'YTickLabels', [y_spacing_and_label])
  colormap_flag = 'set';
else
  figure
  colormap(new_map)

  %%>>>>> Display loaded image.

  imshow(index_matrix)

  %%>>>>> Display filename.

  title(image_read);

  %%>>>>> Display axes with tick marks in increments of fifty.

  x_spacing_and_label = 50 : 50 : horizontal_dimension;
  y_spacing_and_label = 50 : 50 : vertical_dimension;
  set(gca, 'XTick', [x_spacing_and_label])
  set(gca, 'YTick', [y_spacing_and_label])
  set(gca, 'XTickLabels', [x_spacing_and_label])
  set(gca, 'YTickLabels', [y_spacing_and_label])
end

%>>>>>>>>>>                                                      <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                END               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                      <<<<<<<<<<%
```

delete_image.m

```
function [images_read_catalogue, ...
         num_loaded_images, ...
         num_unloaded_images] = delete_image(images_read_catalogue, ...
                                              num_loaded_images, ...
                                              num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  delete_image.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Deletes valid image figure specified by the user.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%**************************************************************************************%
%**************************************************************************************%
%*********************                                            *********************%
%*********************                DELETE IMAGE                *********************%
%*********************                                            *********************%
%**************************************************************************************%
%**************************************************************************************%


  %%>>>>>  Check to see if their is an image to delete.

  if (num_loaded_images == 0 & num_unloaded_images == 0)
    fprintf('\n')
    fprintf('No images are displayed.\n')
  else
    figure_number = input('What image figure number would you like to delete?: ');

    %%>>>>>  Calculate number of records in catalogue.

    max_figure_number = size(images_read_catalogue, 1) / 7;
    if (figure_number < 1 | figure_number > max_figure_number)
      fprintf('\n')
      fprintf('Invalid figure number entered.\n')
    else

      %%>>>>>  Calculate offset using entered figure number.

      offset = figure_number + (6 * (figure_number - 1));

      %%>>>>>  Retrieve image name from record.

      image_name = images_read_catalogue(offset, : );
      image_name = image_name(image_name ~= ' ');

      %%>>>>>  Retrieve image type from record.

      image_type = images_read_catalogue(offset + 5, : );
      image_type = image_type(image_type ~= ' ');
      columns = size(images_read_catalogue, 2);
```

```
%%>>>>> Check to see if first field in record is a blank.

blank_string = [];
if (strcmp(image_name, blank_string))
  fprintf('\n')
  fprintf('Invalid figure number entered.\n')
else

  %%>>>>> Revert action option.

  fprintf('\n')
  fprintf('Are you certain figure ')
  image_number = int2str(figure_number);
  fprintf(image_number)
  fprintf(' should be deleted?\n')

  %%>>>>> Prompt until valid response entered.

  correct_response = 'false';
  while (strcmp(correct_response, 'false'))
    answer = input('Enter yes or no [default]: ', 's');
    if isempty(answer)
      answer = 'no';
    end
    answer = (answer(answer ~= ' '));
    if (strcmp(answer, 'N') | strcmp(answer, 'n') | ...
        strcmp(answer, 'NO') | strcmp(answer, 'no') | ...
        strcmp(answer, 'Y') | strcmp(answer, 'y') | ...
        strcmp(answer, 'YES') | strcmp(answer, 'yes'))
      correct_response = 'true';
    else
      fprintf('\n')
      fprintf('Please reenter your choice.\n')
    end
  end
end
if (strcmp(answer, 'Y') | strcmp(answer, 'y') | ...
    strcmp(answer, 'YES') | strcmp(answer, 'yes'))
  delete(figure_number);

  %%>>>>> Overwrite fields in record with blank strings.

  for index = 0 : 1 : 6
    images_read_catalogue(offset + index, : ) = blanks(columns);
  end
  if (strcmp(image_type, 'loaded'))
    num_loaded_images = num_loaded_images - 1;
  else
    num_unloaded_images = num_unloaded_images - 1;
  end
end
  end
end

%**************************************************************************************%
%**************************************************************************************%
%*********************                                            *********************%
%*********************                   END                      *********************%
%*********************                                            *********************%
%**************************************************************************************%
%**************************************************************************************%
```

du_menu.m

```
function [images_read_catalogue, ...
         image_read, ...
         num_loaded_images, ...
         num_unloaded_images] = du_menu(index_matrix, ...
                                        intensity_matrix, ...
                                        vertical_dimension, ...
                                        horizontal_dimension, ...
                                        gray_level, ...
                                        new_map, ...
                                        image_read, ...
                                        images_read_catalogue, ...
                                        database_flag, ...
                                        num_loaded_images, ...
                                        num_unloaded_images, ...
                                        image_loaded)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  du_menu.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Executes a reload and if successful displays the display utilites menu.
%%                Calls --> reload_image.m
%%                      --> zoom.m
%%                      --> label_image.m
%%                      --> reference_grid.m
%%                      --> histogram.m
%%                      --> intensity_profile.m
%%                      --> density_slice.m
%%                      --> texture_map.m
%%                      --> delete_image.m
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%************************************************************************************%
%************************************************************************************%
%********************                                          ********************%
%********************              DISPLAY UTILITIES           ********************%
%********************                                          ********************%
%************************************************************************************%
%************************************************************************************%


  [index_matrix, ...
   intensity_matrix, ...
   vertical_dimension, ...
   horizontal_dimension, ...
   gray_level, ...
   image_read, ...
   figure_number, ...
   valid_choice] = reload_image(index_matrix, ...
                                intensity_matrix, ...
                                vertical_dimension, ...
                                horizontal_dimension, ...
                                gray_level, ...
                                images_read_catalogue, ...
                                new_map, ...
                                database_flag, ...
                                image_read, ...
                                image_loaded);
```

```
  if (strcmp(image_loaded, 'true') & strcmp(valid_choice, 'true'))
    quit_display = 'no';
    while (strcmp(quit_display, 'no'))
      display_option = menu('DISPLAY UTILITIES MENU (L2)', ...
                             'ZOOM', ...
                             'LABEL IMAGE', ...
                             'REFERENCE GRID', ...
                             'HISTOGRAM', ...
                             'INTENSITY PROFILE', ...
                             'DENSITY SLICE', ...
                             'TEXTURE MAP', ...
                             'DELETE IMAGE', ...
                             'RETURN TO MAIN MENU');


%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                   ZOOM                       <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%


      if display_option == 1
        zoom(images_read_catalogue, ...
             num_loaded_images, ...
             num_unloaded_images);


%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>               LABEL IMAGE                    <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%


      elseif display_option == 2
        label_image(images_read_catalogue, ...
                    num_loaded_images, ...
                    num_unloaded_images);


%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              REFERENCE GRID                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%


      elseif display_option == 3
        images_read_catalogue = reference_grid(vertical_dimension, ...
                                               horizontal_dimension, ...
                                               images_read_catalogue, ...
                                               num_loaded_images);


%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                HISTOGRAM                     <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%


      elseif display_option == 4
        [images_read_catalogue, ...
         num_unloaded_images] = histogram(index_matrix, ...
                                          gray_level, ...
                                          image_read, ...
                                          images_read_catalogue, ...
                                          num_unloaded_images);


%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>            INTENSITY PROFILE                 <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%


      elseif display_option == 5
        [images_read_catalogue, ...
         num_unloaded_images] = intensity_profile(index_matrix, ...
                                                  image_read, ...
                                                  images_read_catalogue, ...
                                                  num_loaded_images, ...
                                                  num_unloaded_images);
```

```
%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              DENSITY SLICE              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


      elseif display_option == 6
        [images_read_catalogue, ...
         num_unloaded_images] = density_slice(intensity_matrix, ...
                                     image_read, ...
                                     images_read_catalogue, ...
                                     num_unloaded_images);


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>               TEXTURE MAP               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


      elseif display_option == 7
        [images_read_catalogue, ...
         num_unloaded_images] = texture_map(intensity_matrix, ...
                                   gray_level, ...
                                   image_read, ...
                                   images_read_catalogue, ...
                                   num_unloaded_images);


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              DELETE IMAGE               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


      elseif display_option == 8
        [images_read_catalogue, ...
         num_loaded_images, ...
         num_unloaded_images] = delete_image(images_read_catalogue, ...
                                    num_loaded_images, ...
                                    num_unloaded_images);


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>          RETURN TO MAIN MENU            <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


      elseif display_option == 9
        quit_display = 'yes';
      end
    end
  end

%**************************************************************************************%
%**************************************************************************************%
%*********************                     END                    *********************%
%*********************                                            *********************%
%*********************                                            *********************%
%**************************************************************************************%
%**************************************************************************************%
```

reload_image.m

```
function [index_matrix, ...
        intensity_matrix, ...
        vertical_dimension, ...
        horizontal_dimension, ...
        gray_level, ...
        image_read, ...
        figure_number, ...
        valid_choice] = reload_image(index_matrix, ...
                                     intensity_matrix, ...
                                     vertical_dimension, ...
                                     horizontal_dimension, ...
                                     gray_level, ...
                                     images_read_catalogue, ...
                                     new_map, ...
                                     database_flag, ...
                                     image_read, ...
                                     image_loaded)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  reload_image.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Allows another loaded image to be selected and made current.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              RELOAD IMAGE               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


  if strcmp(image_loaded, 'false')
    fprintf('\n')
    fprintf('Please load an image before selecting this option.\n')
  else
    valid_choice = 'false';
    figure_number = input('What loaded image figure number would you like to make current?: ');

    %%>>>>> Calculate number of records in catalogue.

    max_figure_number = size(images_read_catalogue, 1) / 7;
    if (figure_number < 1 | figure_number > max_figure_number)
      fprintf('\n')
      fprintf('Invalid figure number entered.\n')
    else

      %%>>>>> Calculate offset using entered figure number.

      offset = figure_number + (6 * (figure_number - 1));

      %%>>>>> Retrieve image name from record.

      new_image_read = images_read_catalogue(offset, : );
      new_image_read = new_image_read(new_image_read ~= ' ');

      %%>>>>> Retrieve image type from record.

      image_type = images_read_catalogue(offset + 5, : );
      image_type = image_type(image_type ~= ' ');
```

reload_image.m

```
blank_string = [];

%%>>>>> Check to see if image type is unloaded.

if (strcmp(image_type, 'unloaded'))
  fprintf('\n')
  fprintf('An unloaded image figure number is not a valid entry.\n')

%%>>>>> Check to see if first field in record is a blank.

elseif (strcmp(new_image_read, blank_string))
  fprintf('\n')
  fprintf('No such figure exists.\n')

%%>>>>> Check to see if specified loaded image figure is the same as the most
%%>>>>> recently loaded image.

elseif (strcmp(new_image_read, image_read))
  valid_choice = 'true';
  fprintf('\n')
  fprintf(new_image_read)
  fprintf(' is the current image.\n')
else
  valid_choice = 'true';

  %%>>>>> Retrieve vertical dimension from record.

  vertical_dimension = images_read_catalogue(offset + 1, : );
  vertical_dimension = (vertical_dimension(vertical_dimension ~= ' '));
  vertical_dimension = str2num(vertical_dimension);

  %%>>>>> Retrieve horizontal dimension from record.

  horizontal_dimension = images_read_catalogue(offset + 2, : );
  horizontal_dimension = (horizontal_dimension(horizontal_dimension ~= ' '));
  horizontal_dimension = str2num(horizontal_dimension);

  %%>>>>> Retrieve gray level from record.

  gray_level = images_read_catalogue(offset + 3, : );
  gray_level = (gray_level(gray_level ~= ' '));
  gray_level = str2num(gray_level);

  %%>>>>> Retrieve numerical precision from record.

  num_precision = images_read_catalogue(offset + 4, : );
  num_precision = num_precision(num_precision ~= ' ');

  %%>>>>> Read image file binary data.

  fid = fopen(new_image_read, 'r');
  [raw_matrix, count] = ...
    fread(fid, [vertical_dimension, horizontal_dimension], num_precision);
  if strcmp(database_flag, 'lorad dsm')

    %%>>>>> Raw data transpose.

    raw_matrix = raw_matrix';
  end
  intensity_matrix = mat2gray(raw_matrix);
  index_matrix = gray2ind(intensity_matrix, gray_level);

  %%>>>>> Make specified loaded image figure current.

  figure(figure_number)

  colormap(new_map)

  %%>>>>> Display loaded image.

  imshow(index_matrix)

  %%>>>>> Display filename.

  title(new_image_read);
```

```
      %%>>>>> Display axes with tick marks in increments of fifty.

      x_spacing_and_label = 50 : 50 : horizontal_dimension;
      y_spacing_and_label = 50 : 50 : vertical_dimension;
      set(gca, 'XTick', [x_spacing_and_label])
      set(gca, 'YTick', [y_spacing_and_label])
      set(gca, 'XTickLabels', [x_spacing_and_label])
      set(gca, 'YTickLabels', [y_spacing_and_label])

      image_read = new_image_read;
      fprintf('\n')
      fprintf(new_image_read)
      fprintf(' is the current image.\n')
    end
  end
end

%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                       <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                 END                   <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                       <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%
```

zoom.m

```
function zoom(images_read_catalogue, ...
              num_loaded_images, ...
              num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  zoom.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) magnify/demagnify an image by x2 with each
%%                mouse click, (2) return image to original configuration with single
%%                mouse click, (3) turn zoom mode off, and (4) select an image figure
%%                to make it current.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                            <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  ZOOM                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                            <<<<<<<<<<%

  if (num_loaded_images == 0 & num_unloaded_images == 0)
    fprintf('\n')
    fprintf('An image must be displayed to select this option.\n')
  else
    quit_zoom = 'no';
    while (strcmp(quit_zoom, 'no'))
      zoom_option = menu('ZOOM SUBMENU (L3)', ...
                         'ZOOM ON', ...
                         'ZOOM OUT', ...
                         'ZOOM OFF', ...
                         'SELECT FIGURE', ...
                         'RETURN TO PREVIOUS MENU');


%/\/\/\/\/\/\/\/\/\/\                  ZOOM ON                 /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\%

      if zoom_option == 1
        fprintf('\n')
        fprintf('Use the left mouse button to zoom in and the right mouse\n')
        fprintf('button to zoom out.\n')

        %%>>>>> Magnify/demagnify an image by x2 with each mouse click.

        imzoom on


%/\/\/\/\/\/\/\/\/\/\                  ZOOM OUT                /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\%

      elseif zoom_option == 2

        %%>>>>> Return image to original configuration.

        imzoom out
```

```
%/\/\/\/\/\/\/\/\/\/\/\                ZOOM OFF                /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%

      elseif zoom_option == 3

        %%>>>>> Turn zoom mode off.

        imzoom off

%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\              SELECT FIGURE             /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%

      elseif zoom_option == 4
        figure_number = input('What image figure number would you like to select?: ');

        %%>>>>> Calculate number of records in catalogue.

        max_figure_number = size(images_read_catalogue, 1) / 7;
        if (figure_number < 1 | figure_number > max_figure_number)
          fprintf('\n')
          fprintf('Invalid figure number entered.\n')
        else

          %%>>>>> Calculate offset using entered figure number.

          offset = figure_number + (6 * (figure_number - 1));

          %%>>>>> Retrieve image name from record.

          new_image_read = images_read_catalogue(offset, : );
          new_image_read = new_image_read(new_image_read ~= ' ');

          %%>>>>> Check to see if first field in record is a blank.

          blank_string = [];
          if (strcmp(new_image_read, blank_string))
            fprintf('\n')
            fprintf('Invalid figure number entered.\n')
          else

            %%>>>>> Make specified image figure current.

            figure(figure_number);
          end
        end

%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\         RETURN TO PREVIOUS MENU        /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\%

      elseif zoom_option == 5
        quit_zoom = 'yes';
      end
    end
  end

%>>>>>>>>>>                                                                <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                            <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  END                       <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                            <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                <<<<<<<<<<%
```

label_image.m

```
function label_image(images_read_catalogue, ...
                     num_loaded_images, ...
                     num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  label_image.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) annotate an image with text and
%%                (2) select an image figure to make it current.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                            <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              LABEL IMAGE               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                            <<<<<<<<<<%

  if (num_loaded_images == 0 & num_unloaded_images == 0)
    fprintf('\n')
    fprintf('An image must be displayed to select this option.\n')
  else
    quit_label = 'no';
    while (strcmp(quit_label, 'no'))
      label_option = menu('LABEL SUBMENU (L3)', ...
                          'LABEL IMAGE', ...
                          'SELECT FIGURE', ...
                          'RETURN TO PREVIOUS MENU');


%/\/\/\/\/\/\/\/\/\/\                                      /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\             LABEL IMAGE              /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                      /\/\/\/\/\/\/\/\/\/\%


      if label_option == 1
        annotation = input('Enter the text: ', 's');
        fprintf('\n')
        fprintf('Place the mouse pointer over the image where the text\n')
        fprintf('is to be entered.  ')
        fprintf('Click any mouse button to enter\n')
        fprintf('the text on the image.\n')

        %%>>>>>  Place text on image.

        gtext(annotation)
```

```
%/\/\/\/\/\/\/\/\/\/\                    SELECT FIGURE                  /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                                   /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                                   /\/\/\/\/\/\/\/\/\/\%

      elseif label_option == 2
        figure_number = input('What image figure number would you like to select?: ');

        %%>>>>> Calculate number of records in catalogue.

        max_figure_number = size(images_read_catalogue, 1) / 7;
        if (figure_number < 1 | figure_number > max_figure_number)
          fprintf('\n')
          fprintf('Invalid figure number entered.\n')
        else

          %%>>>>> Calculate offset using entered figure number.

          offset = figure_number + (6 * (figure_number - 1));

          %%>>>>> Retrieve image name from record.

          new_image_read = images_read_catalogue(offset, : );
          new_image_read = new_image_read(new_image_read ~= ' ');

          %%>>>>> Check to see if first field in record is a blank.

          blank_string = [];
          if (strcmp(new_image_read, blank_string))
            fprintf('\n')
            fprintf('Invalid figure number entered.\n')
          else

            %%>>>>> Make specified image figure current.

            figure(figure_number);
          end
        end

%/\/\/\/\/\/\/\/\/\/\                                                   /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\             RETURN TO PREVIOUS MENU               /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                                   /\/\/\/\/\/\/\/\/\/\%

      elseif label_option == 3
        quit_label = 'yes';
      end
    end
  end

%>>>>>>>>>>                                                                      <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                     END                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                      <<<<<<<<<<%
```

reference_grid.m

```
function images_read_catalogue = reference_grid(vertical_dimension, ...
                                                horizontal_dimension, ...
                                                images_read_catalogue, ...
                                                num_loaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  reference_grid.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) turn on a reference grid; (2) change
%%                the grid block size, color, and line style; and (3) turn off
%%                the reference grid.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             REFERENCE GRID           <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


  exit_grid_menu = 'no';
  while (strcmp(exit_grid_menu, 'no'))
    grid_state = menu('GRID SUBMENU (L3)', ...
                      'TURN ON GRID', ...
                      'CHANGE GRID', ...
                      'TURN OFF GRID', ...
                      'RETURN TO PREVIOUS MENU');


%^^^^^^^^^^^^^^^^^^^^                                   ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^           TURN ON GRID            ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^                                   ^^^^^^^^^^^^^^^^^^^^%


    if grid_state == 1
      if num_loaded_images == 0
        fprintf('\n')
        fprintf('There must be a loaded image to use this option.\n')
      else
        figure_number = input('What loaded image figure would you like to turn the grid on for?: ');

        %%>>>>> Calculate number of records in catalogue.

        max_figure_number = size(images_read_catalogue, 1) / 7;
        if (figure_number < 1 | figure_number > max_figure_number)
          fprintf('\n')
          fprintf('Invalid figure number entered.\n')
        else

          %%>>>>> Calculate offset using entered figure number.

          offset = figure_number + (6 * (figure_number - 1));

          %%>>>>> Retrieve image name from record.

          new_image_read = images_read_catalogue(offset, : );
          new_image_read = new_image_read(new_image_read ~= ' ');

          %%>>>>> Retrieve image type from record.

          image_type = images_read_catalogue(offset + 5, : );
          image_type = image_type(image_type ~= ' ');

          %%>>>>> Retrieve grid state from record.

          grid_state = images_read_catalogue(offset + 6, : );
          grid_state = grid_state(grid_state ~= ' ');
```

```
    blank_string = [];

    %%>>>>> Check to see if image type is unloaded.

    if (strcmp(image_type, 'unloaded'))
      fprintf('\n')
      fprintf('Grid can not be turned on for an unloaded image.\n')

    %%>>>>> Check to see if first field in record is a blank.

    elseif (strcmp(new_image_read, blank_string))
      fprintf('\n')
      fprintf('No such figure exists.\n')

    %%>>>>> Check to see if grid is already turned on for specified loaded image figure.

    elseif (strcmp(grid_state, 'on'))
      fprintf('\n')
      fprintf('The grid is already turned on.\n')
    else

      %%>>>>> Make specified loaded image figure current.

      figure(figure_number);
      grid_on = 'true';
      grid_block_size = input('Please enter the grid block size [16]: ');
      if isempty(grid_block_size)
        grid_block_size = 16;
      end

      %%>>>>> Turn off current tick marks and tick labels.

      set(gca, 'XTick', [])
      set(gca, 'YTick', [])
      set(gca, 'XTickLabels', [])
      set(gca, 'YTickLabels', [])

      %%>>>>> Create vectors corresponding to x and y data values where tick marks
      %%>>>>> should be placed.

      horizontal_grid_spacing = grid_block_size : grid_block_size : horizontal_dimension;
      vertical_grid_spacing = grid_block_size : grid_block_size : vertical_dimension;

      %%>>>>> Add grid lines on current axes.

      grid on

      %%>>>>> Turn on new tick marks and tick labels.

      set(gca, 'XTick', [horizontal_grid_spacing])
      set(gca, 'YTick', [vertical_grid_spacing])
      set(gca, 'XLabel', text(0, 0, 'COLUMN BLOCKS'))
      set(gca, 'YLabel', text(0, 0, 'ROW BLOCKS'))

      %%>>>>> Calculate the number of columns in the catalogue.

      columns = size(images_read_catalogue, 2);

      %%>>>>> Pad 'on' with blanks equivalent to the columnar width of the catalogue
      %%>>>>> minus two.

      state = ['on' blanks(columns - 2)];

      %%>>>>> Insert padded 'on' in the seventh field of the record.

      images_read_catalogue(offset + 6, : ) = state;
    end
  end
end
```

```
%^^^^^^^^^^^^^^^^^^^^^^                    CHANGE GRID                ^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^                                               ^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^                                               ^^^^^^^^^^^^^^^^^^^^^^%


  elseif grid_state == 2
    if num_loaded_images == 0
      fprintf('\n')
      fprintf('There must be a loaded image to use this option.\n')
    else
      figure_number = input('What loaded image figure would you like to change the grid for?: ');

      %%>>>>> Calculate number of records in the catalogue.

      max_figure_number = size(images_read_catalogue, 1) / 7;
      if (figure_number < 1 | figure_number > max_figure_number)
        fprintf('\n')
        fprintf('Invalid figure number entered.\n')
      else

        %%>>>>> Calculate offset using entered figure number.

        offset = figure_number + (6 * (figure_number - 1));

        %%>>>>> Retrieve image name from record.

        new_image_read = images_read_catalogue(offset, : );
        new_image_read = new_image_read(new_image_read ~= ' ');

        %%>>>>> Retrieve image type from record.

        image_type = images_read_catalogue(offset + 5, : );
        image_type = image_type(image_type ~= ' ');

        %%>>>>> Retrieve grid state from record.

        grid_state = images_read_catalogue(offset + 6, : );
        grid_state = grid_state(grid_state ~= ' ');

        blank_string = [];

        %%>>>>> Check to see if image type is unloaded.

        if (strcmp(image_type, 'unloaded'))
          fprintf('\n')
          fprintf('Grid can not be changed for an unloaded image.\n')

        %%>>>>> Check to see if first field in record is blank.

        elseif (strcmp(new_image_read, blank_string))
          fprintf('\n')
          fprintf('No such figure exists.\n')

        %%>>>>> Check to see if grid is off for specified loaded image figure.

        elseif (strcmp(grid_state, 'off'))
          fprintf('\n')
          fprintf('The grid must first be turned on.\n')
        else

          %%>>>>> Make specified loaded image figure current.

          figure(figure_number);
```

reference_grid.m

```
%%>>>>> Prompt until valid response entered.

correct_response = 'false';
while (strcmp(correct_response, 'false'))
  change_block = input('Change the grid block size yes or no [default]?: ', 's');
  if isempty(change_block)
    change_block = 'no';
  end
  change_block = (change_block(change_block ~= ' '));
  if (strcmp(change_block, 'N') | strcmp(change_block, 'n') | ...
      strcmp(change_block, 'NO') | strcmp(change_block, 'no') | ...
      strcmp(change_block, 'Y') | strcmp(change_block, 'y') | ...
      strcmp(change_block, 'YES') | strcmp(change_block, 'yes'))
    correct_response = 'true';
  else
    fprintf('\n')
    fprintf('Please reenter your choice.\n')
  end
end

if (strcmp(change_block, 'Y') | strcmp(change_block, 'y') | ...
    strcmp(change_block, 'YES') | strcmp(change_block, 'yes'))
  size_entered = 'false';
  while (strcmp(size_entered, 'false'))
    grid_block_size = input('Please enter the new grid block size: ');
    if isempty(grid_block_size)
      ;
    else

      %%>>>>> Create vectors corresponding to x and y data values where tick marks
      %%>>>>> should be placed.

      horizontal_grid_spacing = grid_block_size : grid_block_size : horizontal_dimension;
      vertical_grid_spacing = grid_block_size : grid_block_size : vertical_dimension;

      %%>>>>> Update tick spacings.

      set(gca, 'XTick', [horizontal_grid_spacing])
      set(gca, 'YTick', [vertical_grid_spacing])

      size_entered = 'true';
    end
  end
end
```

reference_grid.m

```
%%>>>>> Prompt until valid response entered.

correct_response = 'false';
while (strcmp(correct_response, 'false'))
  change_color = input('Change the grid color yes or no [default]?: ', 's');
  if isempty(change_color)
    change_color = 'no';
  end
  change_color = (change_color(change_color ~= ' '));
  if (strcmp(change_color, 'N') | strcmp(change_color, 'n') | ...
      strcmp(change_color, 'NO') | strcmp(change_color, 'no') | ...
      strcmp(change_color, 'Y') | strcmp(change_color, 'y') | ...
      strcmp(change_color, 'YES') | strcmp(change_color, 'yes'))
    correct_response = 'true';
  else
    fprintf('\n')
    fprintf('Please reenter your choice.\n')
  end
end

if (strcmp(change_color, 'Y') | strcmp(change_color, 'y') | ...
    strcmp(change_color, 'YES') | strcmp(change_color, 'yes'))
  exit_color_menu = 'no';
  while (strcmp(exit_color_menu, 'no'))
    color = menu('COLOR SUBMENU (L4)', ...
                 'black', ...
                 'white', ...
                 'red', ...
                 'green', ...
                 'blue', ...
                 'yellow', ...
                 'magenta', ...
                 'cyan', ...
                 'RETURN TO PREVIOUS MENU');
    if color == 1
      set(gca, 'XColor','black')
      set(gca, 'YColor','black')
    elseif color == 2
      set(gca, 'XColor','white')
      set(gca, 'YColor','white')
    elseif color == 3
      set(gca, 'XColor','red')
      set(gca, 'YColor','red')
    elseif color == 4
      set(gca, 'XColor','green')
      set(gca, 'YColor','green')
    elseif color == 5
      set(gca, 'XColor','blue')
      set(gca, 'YColor','blue')
    elseif color == 6
      set(gca, 'XColor','yellow')
      set(gca, 'YColor','yellow')
    elseif color == 7
      set(gca, 'XColor','magenta')
      set(gca, 'YColor','magenta')
    elseif color == 8
      set(gca, 'XColor','cyan')
      set(gca, 'YColor','cyan')
    elseif color == 9
      exit_color_menu = 'yes';
    end
  end
end
```

```
      %%>>>>> Prompt until valid response entered.

      correct_response = 'false';
      while (strcmp(correct_response, 'false'))
        change_line_style = ...
          input('Change the grid line style yes or no [default]?: ', 's');
        if isempty(change_line_style)
          change_line_style = 'no';
        end
        change_line_style = (change_line_style(change_line_style ~= ' '));
        if (strcmp(change_line_style, 'N') | strcmp(change_line_style, 'n') | ...
            strcmp(change_line_style, 'NO') | strcmp(change_line_style, 'no') | ...
            strcmp(change_line_style, 'Y') | strcmp(change_line_style, 'y') | ...
            strcmp(change_line_style, 'YES') | strcmp(change_line_style, 'yes'))
          correct_response = 'true';
        else
          fprintf('\n')
          fprintf('Please reenter your choice.\n')
        end
      end

      if (strcmp(change_line_style, 'Y') | strcmp(change_line_style, 'y') | ...
          strcmp(change_line_style, 'YES') | strcmp(change_line_style, 'yes'))
        exit_line_style_menu = 'no';
        while (strcmp(exit_line_style_menu, 'no'))
          line_style = menu('LINE STYLE SUBMENU (L4)', ...
                            'solid', ...
                            'dash', ...
                            'dot', ...
                            'dashdot', ...
                            'RETURN TO PREVIOUS MENU');
          if line_style == 1
            set(gca, 'GridLineStyle', '-')
          elseif line_style == 2
            set(gca, 'GridLineStyle', '--')
          elseif line_style == 3
            set(gca, 'GridLineStyle', ':')
          elseif line_style == 4
            set(gca, 'GridLineStyle', '-.')
          elseif line_style == 5
            exit_line_style_menu = 'yes';
          end
        end
      end
    end
  end
end
```

reference_grid.m

```
%/\/\/\/\/\/\/\/\/\/\/\/\            TURN OFF GRID            /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\                                     /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\                                     /\/\/\/\/\/\/\/\/\/\/\/\%

  elseif grid_state == 3
    if num_loaded_images == 0
      fprintf('\n')
      fprintf('There must be a loaded image to use this option.\n')
    else
      figure_number = input('What loaded image figure would you like to turn the grid off for?: ');

      %%>>>>>  Calculate number of records in catalogue.

      max_figure_number = size(images_read_catalogue, 1) / 7;
      if (figure_number < 1 | figure_number > max_figure_number)
        fprintf('\n')
        fprintf('Invalid figure number entered.\n')
      else

        %%>>>>>  Calculate offset using entered figure number.

        offset = figure_number + (6 * (figure_number - 1));

        %%>>>>>  Retrieve image name from record.

        new_image_read = images_read_catalogue(offset, : );
        new_image_read = new_image_read(new_image_read ~= ' ');

        %%>>>>>  Retrieve image type from record.

        image_type = images_read_catalogue(offset + 5, : );
        image_type = image_type(image_type ~= ' ');

        %%>>>>>  Retrieve grid state from record.

        grid_state = images_read_catalogue(offset + 6, : );
        grid_state = grid_state(grid_state ~= ' ');

        blank_string = [];

        %%>>>>>  Check to see if image type is unloaded.

        if (strcmp(image_type, 'unloaded'))
          fprintf('\n')
          fprintf('Grid can not be turned off for an unloaded image.\n')

        %%>>>>>  Check to see if first field in record is blank.

        elseif (strcmp(new_image_read, blank_string))
          fprintf('\n')
          fprintf('No such figure exists.\n')

        %%>>>>>  Check to see if grid is already turned off for specified loaded image figure.

        elseif (strcmp(grid_state, 'off'))
          fprintf('\n')
          fprintf('The grid is already turned off.\n')
        else

          %%>>>>>  Make specified loaded image figure current.

          figure(figure_number);

          %%>>>>>  Remove grid lines on current axes.

          grid off

          %%>>>>>  Return loaded image to its original display format.

          x_spacing_and_label = 50 : 50 : horizontal_dimension;
          y_spacing_and_label = 50 : 50 : vertical_dimension;
          set(gca, 'XTick', [x_spacing_and_label])
          set(gca, 'YTick', [y_spacing_and_label])
          set(gca, 'XTickLabels', [x_spacing_and_label])
          set(gca, 'YTickLabels', [y_spacing_and_label])
          set(gca, 'XLabel', text(0, 0, ''))
          set(gca, 'YLabel', text(0, 0, ''))
          set(gca, 'XColor','white')
          set(gca, 'YColor','white')
```

```
          %%>>>>>  Calculate the number of columns in the catalogue.

          columns = size(images_read_catalogue, 2);

          %%>>>>>  Pad 'off' with blanks equivalent to the columnar width of the catalogue
          %%>>>>>  minus three.

          state = ['off' blanks(columns - 3)];

          %%>>>>>  Insert padded 'off' in the seventh field of the record.

          images_read_catalogue(offset + 6, : ) = state;
        end
      end
    end

%^^^^^^^^^^^^^^^^^^^^^^         RETURN TO PREVIOUS MENU          ^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^                                          ^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^                                          ^^^^^^^^^^^^^^^^^^^^^^%

    elseif grid_state ==4
      exit_grid_menu = 'yes';
    end
  end

%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  END                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%
```

histogram.m

```
function [images_read_catalogue, ...
          num_unloaded_images] = histogram(index_matrix, ...
                                           gray_level, ...
                                           image_read, ...
                                           images_read_catalogue, ...
                                           num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  histogram.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Plots a histogram of the current loaded image.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                            <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              HISTOGRAM                 <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                            <<<<<<<<<<%


  quit_histogram = 'no';
  while (strcmp(quit_histogram, 'no'))
    histogram_option = menu('HISTOGRAM SUBMENU (L3)', ...
                            'DISPLAY HISTOGRAM', ...
                            'RETURN TO PREVIOUS MENU');


%/\/\/\/\/\/\/\/\/\/\                                /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\         DISPLAY HISTOGRAM      /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                /\/\/\/\/\/\/\/\/\/\%

    if histogram_option == 1
      figure
      colormap(gray(gray_level));
      map = colormap;

      %%>>>>>  Plot histogram.

      imhist(index_matrix, map)
      title_string = [image_read ' HISTOGRAM'];
      hold on
      title(title_string)
```

```
%%>>>>> Create record for unloaded image.

image_read_record = str2mat('histogram');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, 'unloaded');
image_read_record = str2mat(image_read_record, ' ');

empty_slot = 'false';

%%>>>>> Calculate number of columns in catalogue.

columns = size(images_read_catalogue, 2);

%%>>>>> Calculate number of records in catalogue.

catalogue_size = size(images_read_catalogue, 1);

%%>>>>> Use linear search to check catalogue for a blank record.

for index = 1 : 1 : catalogue_size / 7
  offset = index + (6 * (index - 1));

  %%>>>>> Check first field in each record to see if it is blank.

  if strcmp(images_read_catalogue(offset, : ), blanks(columns))

    %%>>>>> Insert record into catalogue.

    if offset == 1
      images_read_catalogue = str2mat(image_read_record, ...
        images_read_catalogue(8 : catalogue_size,  : ));
      empty_slot = 'true';
      break
    else
      images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
        image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
      empty_slot = 'true';
      break
    end
  end
end
if strcmp(empty_slot, 'false')

  %%>>>>> Append record to catalogue.

  images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
end
num_unloaded_images = num_unloaded_images + 1;

%^^^^^^^^^^^                RETURN TO PREVIOUS MENU          ^^^^^^^^^^^%
%^^^^^^^^^^^                                                 ^^^^^^^^^^^%
%^^^^^^^^^^^                                                 ^^^^^^^^^^^%

    elseif histogram_option == 2
      quit_histogram = 'yes';
    end
  end

%>>>>>>>>>>                                                 <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>            END               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                 <<<<<<<<<<%
```

intensity_profile.m

```
function [images_read_catalogue, ...
        num_unloaded_images] = intensity_profile(index_matrix, ...
                                                  image_read, ...
                                                  images_read_catalogue, ...
                                                  num_loaded_images, ...
                                                  num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  intensity_profile.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) get pixel coordinates from an image
%%                and (2) plot an intensity profile for a single path traversing
%%                the current loaded image.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>          INTENSITY PROFILE           <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


  quit_profile = 'no';
  while (strcmp(quit_profile, 'no'))
    profile_option = menu('INTENSITY PROFILE SUBMENU (L3)', ...
                          'GET PIXEL COORDINATES', ...
                          'DISPLAY INTENSITY PROFILE', ...
                          'RETURN TO PREVIOUS MENU');
```

intensity_profile.m

```
%/\/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\          GET PIXEL COORDINATES         /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\                                        /\/\/\/\/\/\/\/\/\/\/\/\%

  if profile_option == 1
    if (num_loaded_images == 0 & num_unloaded_images == 0)
      fprintf('\n')
      fprintf('An image must be displayed to select this option.\n')
    else
      figure_number = input('What image figure number would you like to select?: ');

      %%>>>>> Calculate number of records in catalogue.

      max_figure_number = size(images_read_catalogue, 1) / 7;
      if (figure_number < 1 | figure_number > max_figure_number)
        fprintf('\n')
        fprintf('Invalid figure number entered.\n')
      else

        %%>>>>> Calculate offset using entered figure number.

        offset = figure_number + (6 * (figure_number - 1));

        %%>>>>> Retrieve image name from record.

        new_image_read = images_read_catalogue(offset, : );
        new_image_read = new_image_read(new_image_read ~= ' ');

        %%>>>>> Check to see if first field in record is blank.

        blank_string = [];
        if (strcmp(new_image_read, blank_string))
          fprintf('\n')
          fprintf('Invalid figure number entered.\n')
        else

          %%>>>>> Make specified image figure current.

          figure(figure_number);
          fprintf('\n')
          fprintf('Use any mouse button to get the pixel coordinates\n')

          %%>>>>> Determine float coordinates of the image point defined by the mouse cursor.

          [x, y] = ginput(1);

          %%>>>>> Round float coordinates and convert to strings for display.

          x = round(x);
          y = round(y);
          x_string = num2str(x);
          y_string = num2str(y);

          fprintf('\n')
          fprintf('The x coordinate is --> ')
          fprintf(x_string)
          fprintf('\n\n')
          fprintf('The y coordinate is --> ')
          fprintf(y_string)
          fprintf('\n')
        end
      end
    end
```

```
%^^^^^^^^^^^^^                                          ^^^^^^^^^^^^^%
%^^^^^^^^^^^^^         DISPLAY INTENSITY PROFILE        ^^^^^^^^^^^^^%
%^^^^^^^^^^^^^                                          ^^^^^^^^^^^^^%

  elseif profile_option == 2
    fprintf('\n')
    x1 = input('Please enter x1: ');
    y1 = input('Please enter y1: ');
    x2 = input('Please enter x2: ');
    y2 = input('Please enter y2: ');
    figure

    %%>>>>>  Plot bilinear interpolated data using entered endpoints.

    improfile(index_matrix, [x1 x2], [y1, y2])
    title_string = [image_read ' INTENSITY PROFILE'];
    hold on
    title(title_string)

    %%>>>>>  Create record for unloaded image.

    image_read_record = str2mat('intensity profile');
    image_read_record = str2mat(image_read_record, ' ');
    image_read_record = str2mat(image_read_record, ' ');
    image_read_record = str2mat(image_read_record, ' ');
    image_read_record = str2mat(image_read_record, ' ');
    image_read_record = str2mat(image_read_record, 'unloaded');
    image_read_record = str2mat(image_read_record, ' ');

    empty_slot = 'false';

    %%>>>>>  Calculate number of columns in catalogue.

    columns = size(images_read_catalogue, 2);

    %%>>>>>  Calculate number of records in catalogue.

    catalogue_size = size(images_read_catalogue, 1);

    %%>>>>>  Use linear search to check catalogue for a blank record.

    for index = 1 : 1 : catalogue_size / 7
      offset = index + (6 * (index - 1));

      %%>>>>>  Check first field in each record to see if it is blank.

      if strcmp(images_read_catalogue(offset, : ), blanks(columns))

        %%>>>>>  Insert record into catalogue.

        if offset == 1
          images_read_catalogue = str2mat(image_read_record, ...
            images_read_catalogue(8 : catalogue_size,  : ));
          empty_slot = 'true';
          break
        else
          images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
            image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
          empty_slot = 'true';
          break
        end
      end
    end
    if strcmp(empty_slot, 'false')

      %%>>>>>  Append record to catalogue.

      images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
    end
    num_unloaded_images = num_unloaded_images + 1;
```

```
%/\/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\            RETURN TO PREVIOUS MENU       /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                          /\/\/\/\/\/\/\/\/\/\/\%

    elseif profile_option == 3
      quit_profile = 'yes';
    end
  end

%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                    END                       <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%
```

density_slice.m

```
function [images_read_catalogue, ...
         num_unloaded_images] = density_slice(intensity_matrix, ...
                                             image_read, ...
                                             images_read_catalogue, ...
                                             num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  density_slice.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) run a demonstration of color maps
%%                supported by MATLAB and (2) map the current loaded image to
%%                a specified color map.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                       <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>         DENSITY SLICE             <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                       <<<<<<<<<<%


  quit_density_slice = 'no';
  while (strcmp(quit_density_slice, 'no'))
    density_slice_option = menu('DENSITY SLICE SUBMENU (L3)', ...
                                'COLOR MAP DEMO', ...
                                'DISPLAY DENSITY SLICE', ...
                                'RETURN TO PREVIOUS MENU');
```

density_slice.m

```
%/\/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\          COLOR MAP DEMO            /\/\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\/\/\%

  if density_slice_option == 1
    exit_demo = 'no';
    space = ' ';
    figure
    while (strcmp(exit_demo, 'no'))
      color_map_choice = menu('COLOR MAP OPTIONS', ...
                              'bone', ...
                              'cool', ...
                              'copper', ...
                              'flag', ...
                              'gray', ...
                              'hot', ...
                              'hsv', ...
                              'jet', ...
                              'pink', ...
                              'EXIT');

      if color_map_choice == 1
        color_map = 'bone';
      elseif color_map_choice == 2
        color_map = 'cool';
      elseif color_map_choice == 3
        color_map = 'copper';
      elseif color_map_choice == 4
        color_map = 'flag';
      elseif color_map_choice == 5
        color_map = 'gray';
      elseif color_map_choice == 6
        color_map = 'hot';
      elseif color_map_choice == 7
        color_map = 'hsv';
      elseif color_map_choice == 8
        color_map = 'jet';
      elseif color_map_choice == 9
        color_map = 'pink';
      elseif color_map_choice == 10
        color_map_demo = gcf;
        delete(color_map_demo);
        break;
      end

      color_map_levels = input ('Please enter the number of color map levels [64]: ');
      if isempty(color_map_levels)
        color_map_levels = 64;
      end
      map = eval(color_map);
      colormap(map);
      clf

      %%>>>>> Draw specified number of rectangular cells of the selected color map.

      pcolor([1 : color_map_levels + 1; 1 : color_map_levels + 1]')
      set(gca, 'XTickLabels', space)
      hold on

      %%>>>>> Create then display title.

      color_map_uppercase_string = upper(color_map);
      title_string = [color_map_uppercase_string, ' COLOR MAP'];
      title(title_string)

      ylabel('LEVELS')
    end
```

```
%^^^^^^^^^^^^                DISPLAY DENSITY SLICE           ^^^^^^^^^^^^%
%^^^^^^^^^^^^                                                ^^^^^^^^^^^^%
%^^^^^^^^^^^^                                                ^^^^^^^^^^^^%

    elseif density_slice_option == 2
      quit_density_slice_display = 'no';
      space = ' ';
      while (strcmp(quit_density_slice_display, 'no'))
        color_map_choice = menu('COLOR MAP OPTIONS', ...
                                'bone', ...
                                'cool', ...
                                'copper', ...
                                'flag', ...
                                'gray', ...
                                'hot', ...
                                'hsv', ...
                                'jet', ...
                                'pink', ...
                                'QUIT');

        if color_map_choice == 1
          color_map = 'bone';
        elseif color_map_choice == 2
          color_map = 'cool';
        elseif color_map_choice == 3
          color_map = 'copper';
        elseif color_map_choice == 4
          color_map = 'flag';
        elseif color_map_choice == 5
          color_map = 'gray';
        elseif color_map_choice == 6
          color_map = 'hot';
        elseif color_map_choice == 7
          color_map = 'hsv';
        elseif color_map_choice == 8
          color_map = 'jet';
        elseif color_map_choice == 9
          color_map = 'pink';
        elseif color_map_choice == 10
          break;
        end

        num_slicing_planes = input ('Please enter the number of slicing planes [20]: ');
        if isempty(num_slicing_planes)
          num_slicing_planes = 20;
        end
        figure
        map = eval([color_map '(' int2str(num_slicing_planes) ')']);

        %%>>>>> Map intensity matrix to selected color map.

        grayslice_handle = grayslice(intensity_matrix, num_slicing_planes);

        %%>>>>> Display density slice.

        imshow(grayslice_handle, map);
        sub_title_string1 = [image_read ' DENSITY SLICE - '];
        hold on

        %%>>>>> Display color bar to right of density slice.

        colorbar

        %%>>>>> Create then display title.

        color_map_uppercase_string = upper(color_map);
        num_slicing_planes_string = num2str(num_slicing_planes);
        sub_title_string2 = [sub_title_string1 num_slicing_planes_string ' '];
        color_map_uppercase_string = upper(color_map);
        title_string = [sub_title_string2 color_map_uppercase_string ' SLICING PLANES'];
        title(title_string)
```

```
        %%>>>>> Create record for unloaded image.

        image_read_record = str2mat('density slice');
        image_read_record = str2mat(image_read_record, ' ');
        image_read_record = str2mat(image_read_record, ' ');
        image_read_record = str2mat(image_read_record, ' ');
        image_read_record = str2mat(image_read_record, ' ');
        image_read_record = str2mat(image_read_record, 'unloaded');
        image_read_record = str2mat(image_read_record, ' ');

        empty_slot = 'false';

        %%>>>>> Calculate number of columns in catalogue.

        columns = size(images_read_catalogue, 2);

        %%>>>>> Calculate number of records in catalogue.

        catalogue_size = size(images_read_catalogue, 1);

        %%>>>>> Use linear search to check catalogue for a blank record.

        for index = 1 : 1 : catalogue_size / 7
          offset = index + (6 * (index - 1));

          %%>>>>> Check first field in each record to see if it is blank.

          if strcmp(images_read_catalogue(offset, : ), blanks(columns))

            %%>>>>> Insert record into catalogue.

            if offset == 1
              images_read_catalogue = str2mat(image_read_record, ...
                images_read_catalogue(8 : catalogue_size,  : ));
              empty_slot = 'true';
              break
            else
              images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
                image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
              empty_slot = 'true';
              break
            end
          end
        end
        if strcmp(empty_slot, 'false')

          %%>>>>> Append record to catalogue.

          images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
        end
          num_unloaded_images = num_unloaded_images + 1;
      end

%/\/\/\/\/\/\/\/\/\/\/\             RETURN TO PREVIOUS MENU          /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                              /\/\/\/\/\/\/\/\/\/\/\%

    elseif density_slice_option == 3
      quit_density_slice = 'yes';
    end
  end

%>>>>>>>>>>                                                                     <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                                 <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                      END                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                                 <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                     <<<<<<<<<<%
```

texture_map.m

```
function [images_read_catalogue, ...
         num_unloaded_images] = texture_map(intensity_matrix, ...
                                           gray_level, ...
                                           image_read, ...
                                           images_read_catalogue, ...
                                           num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  texture_map.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  Image Processing Toolbox
%%                     Clay M. Thompson & Loren Shure
%%                     The MathWorks, Inc.
%%                     1993
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Generates a texture map of the current loaded image.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>            TEXTURE MAP               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


  quit_texture_map = 'no';
  while (strcmp(quit_texture_map, 'no'))
    texture_map_option = menu('TEXTURE MAP SUBMENU (L3)', ...
                              'DISPLAY TEXTURE MAP', ...
                              'RETURN TO PREVIOUS MENU');
```

texture_map.m

```
%^^^^^^^^^^^^^^^^^^^^^          DISPLAY TEXTURE MAP          ^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^                                       ^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^                                       ^^^^^^^^^^^^^^^^^^^^^%

  if texture_map_option == 1
    scale_factor = input ('Please enter the scale factor to be used to change the image size [0.1]: ');
    if isempty(scale_factor)
      scale_factor = 0.1;
    end

    interpolation_choice = menu('INTERPOLATION OPTIONS', ...
                                'nearest', ...
                                'bilinear', ...
                                'bicubic');

    if interpolation_choice == 1
      interpolation = 'nearest';
    elseif interpolation_choice == 2
      interpolation= 'bilinear';
    elseif interpolation_choice == 3
      interpolation = 'bicubic';
    end

    figure

    %%>>>>>  Change size of intensity matrix.

    resized_matrix = imresize(intensity_matrix, scale_factor, interpolation);
    [rows, columns] = size(resized_matrix);

    %%>>>>>  Transform scaled matrix dimensions to x and y arrays used for 3D plots.

    [x_axis, y_axis] = meshgrid(1 : 1 : rows, 1 : 1 : columns);

    %%>>>>>  Generate 3D intensity mesh plot.

    mesh_surface_handle = mesh(flipud(resized_matrix));

    %%>>>>>  Color mesh white.

    set(mesh_surface_handle, 'Edgecolor', 'white');
    title_string = [image_read ' TEXTURE MAP'];
    hold on
    title(title_string)
    xlabel('MATRIX COLUMN (RESIZED)')
    ylabel('MATRIX ROW (RESIZED)')
    zlabel('INTENSITY')

    %%>>>>>  Place image below mesh plot.

    warp(x_axis, y_axis, -1 * ones(size(x_axis)), intensity_matrix, gray_level);
```

```
      %%>>>>>  Create record for unloaded image.

      image_read_record = str2mat('texture map');
      image_read_record = str2mat(image_read_record, ' ');
      image_read_record = str2mat(image_read_record, ' ');
      image_read_record = str2mat(image_read_record, ' ');
      image_read_record = str2mat(image_read_record, ' ');
      image_read_record = str2mat(image_read_record, 'unloaded');
      image_read_record = str2mat(image_read_record, ' ');

      empty_slot = 'false';

      %%>>>>>  Calculate number of columns in catalogue.

      columns = size(images_read_catalogue, 2);

      %%>>>>>  Calculate number of records in catalogue.

      catalogue_size = size(images_read_catalogue, 1);

      %%>>>>>  Use linear search to check catalogue for a blank record.

      for index = 1 : 1 : catalogue_size / 7
        offset = index + (6 * (index - 1));

        %%>>>>>  Check first field in each record to see if it is blank.

        if strcmp(images_read_catalogue(offset, : ), blanks(columns))

          %%>>>>>  Insert record into catalogue.

          if offset == 1
            images_read_catalogue = str2mat(image_read_record, ...
              images_read_catalogue(8 : catalogue_size,  : ));
            empty_slot = 'true';
            break
          else
            images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
              image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
            empty_slot = 'true';
            break
          end
        end
      end
      if strcmp(empty_slot, 'false')

        %%>>>>>  Append record to catalogue.

        images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
      end
      num_unloaded_images = num_unloaded_images + 1;

%/\/\/\/\/\/\/\/\/\/\/\               RETURN TO PREVIOUS MENU              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                                    /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                                    /\/\/\/\/\/\/\/\/\/\/\%

    elseif texture_map_option == 2
      quit_texture_map = 'yes';
    end
  end

%>>>>>>>>>>                                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                      END                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                  <<<<<<<<<<%
```

ip_menu.m

```
function [images_read_catalogue, ...
        image_read, ...
        num_loaded_images, ...
        num_unloaded_images] = ip_menu(index_matrix, ...
                                        intensity_matrix, ...
                                        vertical_dimension, ...
                                        horizontal_dimension, ...
                                        gray_level, ...
                                        new_map, ...
                                        image_read, ...
                                        images_read_catalogue, ...
                                        database_flag, ...
                                        num_loaded_images, ...
                                        num_unloaded_images, ...
                                        image_loaded)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  ip_menu.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Executes a reload and if successful displays the image processing menu.
%%                Calls --> reload_image.m
%%                      --> region_of_interest.m
%%                      --> edge_extraction.m
%%                      --> delete_image.m
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%*************************************************************************************%
%*************************************************************************************%
%********************                                            ********************%
%********************               IMAGE PROCESSING             ********************%
%********************                                            ********************%
%*************************************************************************************%
%*************************************************************************************%


 [index_matrix, ...
  intensity_matrix, ...
  vertical_dimension, ...
  horizontal_dimension, ...
  gray_level, ...
  image_read, ...
  figure_number, ...
  valid_choice] = reload_image(index_matrix, ...
                               intensity_matrix, ...
                               vertical_dimension, ...
                               horizontal_dimension, ...
                               gray_level, ...
                               images_read_catalogue, ...
                               new_map, ...
                               database_flag, ...
                               image_read, ...
                               image_loaded);

 if (strcmp(image_loaded, 'true') & strcmp(valid_choice, 'true'))
   quit_image_processing = 'no';
   while (strcmp(quit_image_processing, 'no'))
     image_processing_option = menu('IMAGE PROCESSING MENU (L2)', ...
                                    'REGION OF INTEREST', ...
                                    'EDGE EXTRACTION', ...
                                    'DELETE IMAGE', ...
                                    'RETURN TO MAIN MENU');
```

ip_menu.m

```
%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>            REGION OF INTEREST        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


      if image_processing_option == 1
        region_of_interest(index_matrix, ...
                           vertical_dimension, ...
                           horizontal_dimension, ...
                           image_read, ...
                           figure_number);


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             EDGE EXTRACTION          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


      elseif image_processing_option == 2
        [images_read_catalogue, ...
         num_unloaded_images] = edge_extraction(intensity_matrix, ...
                                                image_read, ...
                                                images_read_catalogue, ...
                                                num_unloaded_images);


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              DELETE IMAGE            <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


      elseif image_processing_option == 3
        [images_read_catalogue, ...
         num_loaded_images, ...
         num_unloaded_images] = delete_image(images_read_catalogue, ...
                                             num_loaded_images, ...
                                             num_unloaded_images);


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>           RETURN TO MAIN MENU        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%

      elseif image_processing_option == 4
        quit_image_processing = 'yes';
      end
    end
  end

%******************************************************************************%
%******************************************************************************%
%*********************                                     *********************%
%*********************                 END                 *********************%
%*********************                                     *********************%
%******************************************************************************%
%******************************************************************************%
```

```
function region_of_interest(index_matrix, ...
                            vertical_dimension, ...
                            horizontal_dimension, ...
                            image_read, ...
                            figure_number)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  region_of_interest.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Provides options to (1) extract single or multiple variable
%%                sized, rectangular regions of interest from the current
%%                loaded image and (2) save as text files the coordinates defining
%%                the region of interest window(s) and the pixel values within.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                            <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>           REGION OF INTEREST           <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                        <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                            <<<<<<<<<<%


  quit_roi = 'no';
  roi_selected = 'false';
  num_ASCII_files = 1;
  while (strcmp(quit_roi, 'no'))
    roi_option = menu('REGION OF INTEREST SUBMENU (L3)', ...
                      'SELECT ROIs', ...
                      'SAVE ROIs AND COORDINATE INFO', ...
                      'RETURN TO PREVIOUS MENU');


%/\/\/\/\/\/\/\/\/\/\                                      /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\              SELECT ROIS             /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                      /\/\/\/\/\/\/\/\/\/\%

    if roi_option == 1

      %%>>>>>  Make most recently loaded image as current figure.

      figure(figure_number);
      roi_num_rows = input ('Please enter the vertical dimension (rows) of the ROI [16]: ');
      if isempty(roi_num_rows)
        roi_num_rows = 16;
      end
      roi_num_columns = input ('Please enter the horizontal dimension (columns) of the ROI [16]: ');
      if isempty(roi_num_columns)
        roi_num_columns = 16;
      end

      fprintf('\n')
      fprintf('Use any mouse button to select the upper left \n')
      fprintf('x and y coordinates of the regions of interest.  The \n')
      fprintf('lower right x and y coordinates for each region of \n')
      fprintf('interest will be calculated by the program.\n')
      fprintf('After the coordinates are selected, keep the mouse \n')
      fprintf('pointer in the image window and press return.\n')
```

region_of_interest.m

```
%%>>>>> Get float coordinate(s) of image point(s) defined by the mouse cursor and
%%>>>>> store into two column vectors.

[upper_left_x, upper_left_y] = ginput

%%>>>>> Store two vectors column wise into a matrix.

coordinates = [];
coordinates = [upper_left_x, upper_left_y];

%%>>>>> Determine number of points selected.

num_coordinates = size([upper_left_x, upper_left_y], 1);

%%>>>>> Prompt until valid response entered.

correct_response = 'false';
while (strcmp(correct_response, 'false'))
  truth = input('Are the regions normal (N, n), benign (B, b) or malignant (M, m) [default]?: ', 's');
  if isempty(truth)
    truth = 'malignant';
  end
  truth = (truth(truth ~= ' '));
  if (strcmp(truth, 'N') | strcmp(truth, 'n') | ...
      strcmp(truth, 'NORMAL') | strcmp(truth, 'normal'))
    truth = 'normal';
    correct_response = 'true';
  elseif (strcmp(truth, 'B') | strcmp(truth, 'b') | ...
          strcmp(truth, 'BENIGN') | strcmp(truth, 'benign'))
    truth = 'benign';
    correct_response = 'true';
  elseif (strcmp(truth, 'M') | strcmp(truth, 'm') | ...
          strcmp(truth, 'MALIGNANT') | strcmp(truth, 'malignant'))
    truth = 'malignant';
    correct_response = 'true';
  else
    fprintf('\n')
    fprintf('Please reenter your choice.\n')
  end
end
roi_selected = 'true';

coordinate_matrix = [];
counter = 1;

for index = 1 : 1 : num_coordinates

  %%>>>>> Check to see if any selected point is beyond the image boundary
  %%>>>>> or if any roi window crosses the right or bottom edge of the image.

  if (coordinates(index : index, 1 : 1) < 0 | ...
      coordinates(index : index, 1 : 1) > horizontal_dimension | ...
      coordinates(index : index, 2 : 2) < 0 | ...
      coordinates(index : index, 2 : 2) > vertical_dimension | ...
      round(coordinates(index : index, 1 : 1)) + roi_num_rows > horizontal_dimension | ...
      round(coordinates(index : index, 2 : 2)) + roi_num_columns > vertical_dimension)

    %%>>>>> Print to screen invalid selected points.

    fprintf('\n')
    fprintf('Invalid --> upper_left_x = %f  upper_left_y = %f \n', ...
           coordinates(index : index, 1 : 1), ...
           coordinates(index : index, 2 : 2))
  else

    %%>>>>> Use upper left coordinate and entered roi dimensions to calculate
    %%>>>>> the lower right coordinate.  Save both coordinates into a matrix.

    x_upper_left = round(coordinates(index : index, 1 : 1));
    y_upper_left = round(coordinates(index : index, 2 : 2));
    x_lower_right = x_upper_left + roi_num_rows - 1;
    y_lower_right = y_upper_left + roi_num_columns - 1;
    coordinate_matrix(counter : counter, 1 : 1) = x_upper_left;
    coordinate_matrix(counter : counter, 2 : 2) = y_upper_left;
    coordinate_matrix(counter : counter, 3 : 3) = x_lower_right;
    coordinate_matrix(counter : counter, 4 : 4) = y_lower_right;
    counter = counter + 1;
  end
end
```

region_of_interest.m

```
%%>>>>> Print to screen the coordinates of the roi window(s) and their truth.

fprintf('\n\n\n')
fprintf('                   ROI COORDINATES SELECTED                    \n')
fprintf('                   ************************                    \n\n')
fprintf('upper left x   upper left y   lower right x   lower right y')
fprintf('  truth\n');
num_valid_coordinates = size(coordinate_matrix, 1);
for index = 1 : 1 : num_valid_coordinates
  fprintf('%3.0f           %3.0f            %3.0f             %3.0f', ...
         coordinate_matrix(index : index, 1 : 1), ...
         coordinate_matrix(index : index, 2 : 2), ...
         coordinate_matrix(index : index, 3 : 3), ...
         coordinate_matrix(index : index, 4 : 4));
  fprintf(blanks(12));
  fprintf(truth);
  fprintf('\n');
end

%%>>>>> Print to screen pixel values within roi window(s).

fprintf('\n\n\n')
for index = 1 : 1 : num_valid_coordinates
  fprintf('>>> region of interest ')
  fprintf('%3.0f', index)
  fprintf(' <<<')
  fprintf('\n')
  fprintf('\n')
  x_upper_left = coordinate_matrix(index : index, 1 : 1);
  y_upper_left = coordinate_matrix(index : index, 2 : 2);
  x_lower_right = coordinate_matrix(index : index, 3 : 3);
  y_lower_right = coordinate_matrix(index : index, 4 : 4);
  disp(index_matrix(x_upper_left : x_lower_right, y_upper_left : y_lower_right))
  fprintf('\n\n')
end
```

```
%^^^^^^^^^^^^       SAVE ROIs AND COORDINATE INFO       ^^^^^^^^^^^^%
%^^^^^^^^^^^^                                           ^^^^^^^^^^^^%
%^^^^^^^^^^^^                                           ^^^^^^^^^^^^%

  elseif roi_option == 2
    if strcmp(roi_selected, 'false')
      fprintf('\n')
      fprintf('Please select ROIs prior to choosing this option.\n')
    else

      %%>>>>> Create roi filename extension.

      extension1 = ['.roi', int2str(num_ASCII_files)];

      %%>>>>> Create complete roi filename.

      roi_filename = [image_read, extension1];
      output_file1 = eval('roi_filename');

      %%>>>>> Extract each roi from the loaded image and store it in a matrix.

      for index = 1 : 1 : num_valid_coordinates
        x_upper_left = coordinate_matrix(index : index, 1 : 1);
        y_upper_left = coordinate_matrix(index : index, 2 : 2);
        x_lower_right = coordinate_matrix(index : index, 3 : 3);
        y_lower_right = coordinate_matrix(index : index, 4 : 4);
        eval(['roi_matrix' int2str(index) ...
        '= index_matrix(x_upper_left : x_lower_right, y_upper_left : y_lower_right);']);
      end

      %%>>>>> Store each matrix top-down into one large matrix.

      roi_matrices = [];
      for index = 1 : 1 : num_valid_coordinates
        eval(['temp = roi_matrix' int2str(index) ';']);
        roi_matrices = [roi_matrices; temp];
      end

      %%>>>>> Pad truth, roi dimensions, and number of rois with zeros equivalent to the
      %%>>>>> columnar width of the roi(s)

      tissue = zeros(1, roi_num_columns);
      if strcmp(truth, 'normal')
        tissue(1, 1) = 'n';
      elseif strcmp(truth, 'benign')
        tissue(1, 1) = 'b';
      else
        tissue(1, 1) = 'm';
      end

      roi_rows = zeros(1, roi_num_columns);
      roi_rows(1, 1) = roi_num_rows;

      roi_columns = zeros(1, roi_num_columns);
      roi_columns(1, 1) = roi_num_columns;

      num_roi_matrices = zeros(1, roi_num_columns);
      num_roi_matrices(1, 1) = index;

      %%>>>>> Save truth, roi dimensions, number of rois, and rois in ASCII format to roi file.

      eval(['save ' output_file1 ' tissue roi_rows roi_columns num_roi_matrices roi_matrices -ascii'])
      fprintf('\n')
      fprintf('ROIs saved to ')
      fprintf(roi_filename)
      fprintf('\n')

      %%>>>>> Create coordinate filename extension.

      extension2 = ['.coordinate', int2str(num_ASCII_files)];

      %%>>>>> Create complete coordinate filename.

      coordinate_filename = [image_read, extension2];
      output_file2 = eval('coordinate_filename');
      num_ASCII_files = num_ASCII_files + 1;
```

```
      %%>>>>>  Open coordinate file and write filename rois extracted from,
      %%>>>>>  coordinate filename, coordinates defining roi window(s), and assigned
      %%>>>>>  truth of rois.

      fid = fopen(output_file2, 'w');
      fprintf(fid, 'Input image filename --> ');
      fprintf(fid, image_read);
      fprintf(fid, '\n\n');
      fprintf(fid, 'Output ROI coordinates filename --> ');
      fprintf(fid, coordinate_filename);
      fprintf(fid, '\n\n\n');
      fprintf(fid, '>>>>>>>>>>>>>>>>>>>>>>>>>  ROI COORDINATES  ');
      fprintf(fid, '<<<<<<<<<<<<<<<<<<<<<<<<\n');
      fprintf(fid, 'upper left x   upper left y   lower right x   lower right y');
      fprintf(fid, '  truth\n');
      for index = 1 : 1 : num_valid_coordinates
        fprintf(fid, '%3.0f           %3.0f           %3.0f              %3.0f', ...
                coordinate_matrix(index : index, 1 : 1), ...
                coordinate_matrix(index : index, 2 : 2), ...
                coordinate_matrix(index : index, 3 : 3), ...
                coordinate_matrix(index : index, 4 : 4));
        fprintf(fid, blanks(12));
        fprintf(fid, truth);
        fprintf(fid, '\n');
      end
      fprintf('\n')
      fprintf('Coordinates saved to ')
      fprintf(coordinate_filename)
      fprintf('\n')
    end

%^^^^^^^^^^^^^^^^^^^^^                                        ^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^         RETURN TO PREVIOUS MENU        ^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^                                        ^^^^^^^^^^^^^^^^^^^^^%

    elseif roi_option == 3
      quit_roi = 'yes';
    end
  end

%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                 END                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%
```

edge_extraction.m

```
function [images_read_catalogue, ...
         num_unloaded_images] = edge_extraction(intensity_matrix, ...
                                                image_read, ...
                                                images_read_catalogue, ...
                                                num_unloaded_images)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  edge_extraction.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION: Displays a binary image of edges detected in an intensity image
%%               using the Roberts, Prewitt, Sobel, or Marr-Hildreth algorithms.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>            EDGE EXTRACTION               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%


  quit_edge_extraction = 'no';
  while (strcmp(quit_edge_extraction, 'no'))
    edge_option = menu('EDGE EXTRACTION SUBMENU (L3)', ...
                       'ROBERTS', ...
                       'PREWITT',...
                       'SOBEL', ...
                       'MARR-HILDRETH', ...
                       'RETURN TO PREVIOUS MENU');

    if edge_option == 1
      edge_algorithm = 'roberts';
    elseif edge_option == 2
      edge_algorithm = 'prewitt';
    elseif edge_option == 3
      edge_algorithm = 'sobel';
    elseif edge_option == 4
      edge_algorithm = 'marr';
    elseif edge_option == 5
      break;
    end

    noise_tolerance = input ('Please enter the noise tolerance [0.01]: ');
    if isempty(noise_tolerance)
      noise_tolerance = 0.01;
    end
```

edge_extraction.m

```
if (strcmp(edge_algorithm, 'roberts') | strcmp(edge_algorithm, 'prewitt') | ...
    strcmp(edge_algorithm, 'sobel'))
  directionality_option = menu('DIRECTIONALITY FACTOR OPTIONS', ...
                               'nondirectional', ...
                               'horizontal', ...
                               'vertical');
  if directionality_option == 1

    %%>>>>> Nondirectional.

    directionality = [1 1];
  elseif directionality_option == 2

    %%>>>>> Horizontal.

    directionality = [0 1];
  elseif directionality_option == 3

    %%>>>>> Vertical.

    directionality = [1 0];
  end
  figure

  %%>>>>> Display Roberts, Prewitt, or Sobel binary image.

  imshow(~edge(intensity_matrix, noise_tolerance, edge_algorithm, directionality), 2)
  if strcmp(edge_algorithm, 'roberts')
    title_string = [image_read ' - ROBERTS EDGE EXTRACTION'];
  elseif strcmp(edge_algorithm, 'prewitt')
    title_string = [image_read ' - PREWITT EDGE EXTRACTION'];
  elseif strcmp(edge_algorithm, 'sobel')
    title_string = [image_read ' - SOBEL EDGE EXTRACTION'];
  end
  hold on
  title(title_string)
else
  figure

  %%>>>>> Display Marr-Hildreth binary image.

  imshow(~edge(intensity_matrix, noise_tolerance, edge_algorithm), 2)
  title_string = [image_read ' - MARR-HILDRETH EDGE EXTRACTION'];
  hold on
  title(title_string)
end
```

```
%%>>>>> Create record for unloaded image.

image_read_record = str2mat('edge extraction');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, ' ');
image_read_record = str2mat(image_read_record, 'unloaded');
image_read_record = str2mat(image_read_record, ' ');

empty_slot = 'false';

%%>>>>> Calculate number of columns in catalogue.

columns = size(images_read_catalogue, 2);

%%>>>>> Calculate number of records in catalogue.

catalogue_size = size(images_read_catalogue, 1);

%%>>>>> Use linear search to check catalogue for a blank record.

for index = 1 : 1 : catalogue_size / 7
  offset = index + (6 * (index - 1));

  %%>>>>> Check first field in each record to see if it is blank.

  if strcmp(images_read_catalogue(offset, : ), blanks(columns))

    %%>>>>> Insert record into catalogue.

    if offset == 1
      images_read_catalogue = str2mat(image_read_record, ...
        images_read_catalogue(8 : catalogue_size,  : ));
      empty_slot = 'true';
      break
    else
      images_read_catalogue = str2mat(images_read_catalogue(1 : 7 * (index - 1),  : ), ...
        image_read_record, images_read_catalogue((7 * index) + 1 : catalogue_size,  : ));
      empty_slot = 'true';
      break
    end
  end
end
if strcmp(empty_slot, 'false')

  %%>>>>> Append record to catalogue.

  images_read_catalogue = str2mat(images_read_catalogue, image_read_record);
end
num_unloaded_images = num_unloaded_images + 1;
```

```
%%>>>>>  Prompt until valid response entered.

correct_response = 'false';
while (strcmp(correct_response, 'false'))
  complement = input('Complement the figure yes or no [default]?: ', 's');
  if isempty(complement)
    complement = 'no';
  end
  complement = (complement(complement ~= ' '));
  if (strcmp(complement, 'N') | strcmp(complement, 'n') | ...
      strcmp(complement, 'NO') | strcmp(complement, 'no') | ...
      strcmp(complement, 'Y') | strcmp(complement, 'y') | ...
      strcmp(complement, 'YES') | strcmp(complement, 'yes'))
    correct_response = 'true';
  else
    fprintf('\n')
    fprintf('Please reenter your choice.\n')
  end
end
if (strcmp(complement, 'Y') | strcmp(complement, 'y') | ...
    strcmp(complement, 'YES') | strcmp(complement, 'yes'))
  if (strcmp(edge_algorithm, 'roberts') | strcmp(edge_algorithm, 'prewitt') | ...
      strcmp(edge_algorithm, 'sobel'))

    %%>>>>>  Display complemented Roberts, Prewitt, or Sobel binary image.

    imshow(edge(intensity_matrix, noise_tolerance, edge_algorithm, directionality), 2)
  else

    %%>>>>>  Display complemented Marr-Hildreth binary image.

    imshow(edge(intensity_matrix, noise_tolerance, edge_algorithm), 2)
  end
end
end

%>>>>>>>>>>                                                   <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             END               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                   <<<<<<<<<<%
```

ia_menu.m

```
function [LNKnet_input_file, ...
          LNKnet_file_loaded, ...
          LNKnet_fid] = ia_menu(LNKnet_input_file, ...
                                LNKnet_file_loaded, ...
                                LNKnet_fid)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  ia_menu.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Displays image analysis menu.
%%                Calls --> feature_extraction.m
%%                      --> segmentation.m
%%                      --> classification.m
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%**************************************************************************************%
%**************************************************************************************%
%********************                                              ********************%
%********************               IMAGE ANALYSIS                 ********************%
%********************                                              ********************%
%**************************************************************************************%
%**************************************************************************************%


  quit_image_analysis = 'no';
  while (strcmp(quit_image_analysis, 'no'))
    image_analysis_option = menu('IMAGE ANALYSIS MENU (L2)', ...
                                 'FEATURE EXTRACTION', ...
                                 'SEGMENTATION', ...
                                 'CLASSIFICATION', ...
                                 'RETURN TO MAIN MENU');


%>>>>>>>>>>                                                                   <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>           FEATURE EXTRACTION                  <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                   <<<<<<<<<<%


    if image_analysis_option == 1
      [LNKnet_input_file, ...
       LNKnet_file_loaded, ...
       LNKnet_fid] = feature_extraction(LNKnet_input_file, ...
                                        LNKnet_file_loaded, ...
                                        LNKnet_fid);


%>>>>>>>>>>                                                                   <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              SEGMENTATION                     <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                                   <<<<<<<<<<%


    elseif image_analysis_option == 2
      segmentation;
```

```
%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              CLASSIFICATION              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%


    elseif image_analysis_option == 3
      classification;


%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>           RETURN TO MAIN MENU            <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%


    elseif image_analysis_option == 4
      quit_image_analysis = 'yes';
    end
  end

%***************************************************************************************%
%***************************************************************************************%
%********************                                               ********************%
%********************                      END                      ********************%
%********************                                               ********************%
%***************************************************************************************%
%***************************************************************************************%
```

feature_extraction.m

```
function [LNKnet_input_file, ...
          LNKnet_file_loaded, ...
          LNKnet_fid] = feature_extraction(LNKnet_input_file, ...
                                           LNKnet_file_loaded, ...
                                           LNKnet_fid)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  feature_extraction.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Displays feature extraction menu.
%%                Calls --> load_roi_file.m
%%                      --> template1.m
%%                      --> template2.m
%%                      --> template3.m
%%                      --> template4.m
%%                      --> print_to_file.m
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                          <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>          FEATURE EXTRACTION          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                      <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                          <<<<<<<<<<%


    fprintf('\n')
    fprintf('NOTE:  DO NOT SELECT THE PRINT OPTION UNTIL ALL\n')
    fprintf('       DESIRED FEATURE EXTRACTIONS HAVE BEEN RUN\n')
    fprintf('       FOR THE LOADED ROI FILE. \n')
    quit_feature = 'no';
    input_file_loaded = 'false';
    feature_called_flag = 'false';
    while (strcmp(quit_feature, 'no'))
      feature_option = menu('FEATURE EXTRACTION SUBMENU (L3)', ...
                            'LOAD ROI FILE', ...
                            'TEMPLATE1', ...
                            'TEMPLATE2', ...
                            'TEMPLATE3', ...
                            'TEMPLATE4', ...
                            'PRINT TO FILE', ...
                            'RETURN TO PREVIOUS MENU');


%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\             LOAD ROI FILE         /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%


      if feature_option == 1
        [input_file, ...
         input_file_loaded, ...
         tissue_value, ...
         roi_num_rows, ...
         roi_num_columns, ...
         num_rois, ...
         rois, ...
         feature1_flag, ...
         feature2_flag, ...
         feature3_flag, ...
         feature4_flag] = load_roi_file;
```

```
%^^^^^^^^^^^             TEMPLATE1              ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%

   elseif feature_option == 2
     if strcmp(input_file_loaded, 'false')
       fprintf('\n')
       fprintf('Please load a ROI file prior to selecting this option.\n')
     else

       %%>>>>>  Rename template1 to the name of feature extraction routine,
       %%>>>>>  then update the menu.

       feature1_vector = template1(roi_num_rows, ...
                                   roi_num_columns, ...
                                   num_rois, ...
                                   rois)
       feature1_flag = 'true';
       feature_called_flag = 'true';
     end

%^^^^^^^^^^^             TEMPLATE2              ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%

   elseif feature_option == 3
     if strcmp(input_file_loaded, 'false')
       fprintf('\n')
       fprintf('Please load a ROI file prior to selecting this option.\n')
     else

       %%>>>>>  Rename template2 to the name of feature extraction routine,
       %%>>>>>  then update the menu.

       feature2_vector = template2(roi_num_rows, ...
                                   roi_num_columns, ...
                                   num_rois, ...
                                   rois)
       feature2_flag = 'true';
       feature_called_flag = 'true';
     end

%^^^^^^^^^^^             TEMPLATE3              ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%
%^^^^^^^^^^^                                    ^^^^^^^^^^^%

   elseif feature_option == 4
     if strcmp(input_file_loaded, 'false')
       fprintf('\n')
       fprintf('Please load a ROI file prior to selecting this option.\n')
     else

       %%>>>>>  Rename template3 to the name of feature extraction routine,
       %%>>>>>  then update the menu.

       feature3_vector = template3(roi_num_rows, ...
                                   roi_num_columns, ...
                                   num_rois, ...
                                   rois)
       feature3_flag = 'true';
       feature_called_flag = 'true';
     end
```

```
%^^^^^^^^^^^                    TEMPLATE4                   ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%

      elseif feature_option == 5
        if strcmp(input_file_loaded, 'false')
          fprintf('\n')
          fprintf('Please load a ROI file prior to selecting this option.\n')
        else

          %%>>>>> Rename template4 to the name of feature extraction routine,
          %%>>>>> then update the menu.

          feature4_vector = template4(roi_num_rows, ...
                                      roi_num_columns, ...
                                      num_rois, ...
                                      rois)
          feature4_flag = 'true';
          feature_called_flag = 'true';
        end

%^^^^^^^^^^^                  PRINT TO FILE                 ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%

      elseif feature_option == 6
        [LNKnet_input_file, ...
         LNKnet_file_loaded, ...
         LNKnet_fid] = print_to_file(input_file, ...
                                     LNKnet_input_file, ...
                                     LNKnet_file_loaded, ...
                                     LNKnet_fid, ...
                                     tissue_value, ...
                                     num_rois, ...
                                     feature_called_flag, ...
                                     feature1_flag, ...
                                     feature2_flag, ...
                                     feature3_flag, ...
                                     feature4_flag, ...
                                     feature1_vector, ...
                                     feature2_vector, ...
                                     feature3_vector, ...
                                     feature4_vector);

%^^^^^^^^^^^              RETURN TO PREVIOUS MENU           ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%
%^^^^^^^^^^^                                                ^^^^^^^^^^^%

      elseif feature_option == 7
        quit_feature = 'yes';
      end
    end

%>>>>>>>>>>                                                  <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             END              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                              <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                  <<<<<<<<<<%
```

load_roi_file.m

```
function [input_file, ...
          input_file_loaded, ...
          tissue_value, ...
          roi_num_rows, ...
          roi_num_columns, ...
          num_rois, ...
          rois, ...
          feature1_flag, ...
          feature2_flag, ...
          feature3_flag, ...
          feature4_flag] = load_roi_file
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  load_roi_file.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Load ASCII roi files created by region of interest routine.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\           LOAD ROI FILE         /\/\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\/\/\%


  %%>>>>>  Prompt until valid filename entered.

  roi_fid = -1;
  while (roi_fid == -1)
    input_file = ...
      input('Please enter the name of the ROI file (e.g. dn1swb.roi1): ', 's');
    if isempty(input_file)
      input_file = 'dn1swb.roi1';
    end
    input_file = (input_file(input_file ~= ' '));
    roi_fid = fopen(input_file, 'r');
    if (roi_fid == -1)
      fprintf('\n')
      fprintf('File could not be located.\n')
    end
  end

  %%>>>>>  Convert input file string to ASCII numeric values.

  string_vector = abs(input_file);
  index = 1;
  roi_file = [];

  %%>>>>>  Truncate dot extension from input file, then assign input file to roi file.

  while (string_vector(index) ~= 46 & index <= length(input_file))
    roi_file(index) = input_file(index);
    index = index + 1;
  end
```

load_roi_file.m

```
%%>>>>>  Read rectangular array of ASCII data in input file into a matrix
%%>>>>>  with same name as the input file, but without the dot extension.

eval(['load ' input_file]);

%%>>>>>  Prep matrix for processing.

roi_data = eval(roi_file);

%%>>>>>  Partition matrix.

tissue_type_ascii = roi_data(1 : 1, 1 : 1);
tissue_type_text = setstr(tissue_type_ascii);
if strcmp(tissue_type_text, 'n')

  %%>>>>>  Normal tissue.

  tissue_class = 0;
elseif strcmp(tissue_type_text, 'b')

  %%>>>>>  Benign tissue.

  tissue_class = 1;
else

  %%>>>>>  Malignant tissue.

  tissue_class = 2;
end
tissue_value = num2str(tissue_class);
roi_num_rows = roi_data(2 : 2, 1 : 1);
roi_num_columns = roi_data(3 : 3, 1 : 1);
num_rois = roi_data(4 : 4, 1 : 1);
rois = roi_data(5 : (4 + (roi_num_rows * num_rois)), 1 : roi_num_columns);

input_file_loaded = 'true';
feature1_flag = 'false';
feature2_flag = 'false';
feature3_flag = 'false';
feature4_flag = 'false';

%/\/\/\/\/\/\/\/\/\/\/\                END                /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%
```

template1.m

```
function answer_vector = template1(roi_num_rows, ...
                                  roi_num_columns, ...
                                  num_rois, ...
                                  rois)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  template1.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Feature extraction template that sums each ROI.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                    Modification bullet
%%                                    Modification bullet
%%                         DD Mon YY - Author
%%                                    Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\/\                              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\          TEMPLATE1           /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                              /\/\/\/\/\/\/\/\/\/\/\%

  increment = 0;
  answer_vector = zeros(1,num_rois);
  for index = 1 : 1 : num_rois
    eval(['roi' int2str(index) ...
      '= rois(1 + increment : roi_num_rows + increment, 1 : roi_num_columns);']);

    %%>>>>> Alternative partition of rois.
    % roi_matrix = eval(['(roi' int2str(index) ')']);
    % answer_vector(index) = sum(sum(roi_matrix));

    answer_vector(index) = eval(['sum(sum(roi' int2str(index) '))']);
    increment = increment + roi_num_rows;
  end

%/\/\/\/\/\/\/\/\/\/\/\                              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\             END              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                              /\/\/\/\/\/\/\/\/\/\/\%
```

template2.m

```
function answer_vector = template2(roi_num_rows, ...
                                   roi_num_columns, ...
                                   num_rois, ...
                                   rois)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  template2.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Feature extraction template that finds the maximum element
%%                for each ROI.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\              TEMPLATE2             /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\%

  increment = 0;
  answer_vector = zeros(1,num_rois);
  for index = 1 : 1 : num_rois
    eval(['roi' int2str(index) ...
      '= rois(1 + increment : roi_num_rows + increment, 1 : roi_num_columns);']);

    %%>>>>> Alternative partition of rois.
    % roi_matrix = eval(['(roi' int2str(index) ')']);
    % answer_vector(index) = sum(sum(roi_matrix));

    answer_vector(index) = eval(['max(max(roi' int2str(index) '))']);
    increment = increment + roi_num_rows;
  end

%/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                END                 /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                    /\/\/\/\/\/\/\/\/\/\/\%
```

template3.m

```
function answer_vector = template3(roi_num_rows, ...
                                  roi_num_columns, ...
                                  num_rois, ...
                                  rois)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  template3.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Feature extraction template that finds the minimum element
%%                for each ROI.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\              TEMPLATE3          /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\%

  increment = 0;
  answer_vector = zeros(1,num_rois);
  for index = 1 : 1 : num_rois
    eval(['roi' int2str(index) ...
      '= rois(1 + increment : roi_num_rows + increment, 1 : roi_num_columns);']);

    %%>>>>> Alternative partition of rois.
    % roi_matrix = eval(['(roi' int2str(index) ')']);
    % answer_vector(index) = sum(sum(roi_matrix));

    answer_vector(index) = eval(['min(min(roi' int2str(index) '))']);
    increment = increment + roi_num_rows;
  end

%/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                 END             /\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\                                 /\/\/\/\/\/\/\/\/\/\%
```

```
function answer_vector = template4(roi_num_rows, ...
                                   roi_num_columns, ...
                                   num_rois, ...
                                   rois)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  template4.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Feature extraction template that calculates the mean value for
%%                each ROI.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\/\                                  /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\              TEMPLATE4           /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                  /\/\/\/\/\/\/\/\/\/\/\%


  increment = 0;
  answer_vector = zeros(1,num_rois);
  for index = 1 : 1 : num_rois
    eval(['roi' int2str(index) ...
      '= rois(1 + increment : roi_num_rows + increment, 1 : roi_num_columns);']);

    %%>>>>>  Alternative partition of rois.
    % roi_matrix = eval(['(roi' int2str(index) ')']);
    % answer_vector(index) = sum(sum(roi_matrix));

    answer_vector(index) = eval(['mean(mean(roi' int2str(index) '))']);
    increment = increment + roi_num_rows;
  end

%/\/\/\/\/\/\/\/\/\/\/\                                  /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                 END              /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                  /\/\/\/\/\/\/\/\/\/\/\%
```

print_to_file.m

```
function [LNKnet_input_file, ...
         LNKnet_file_loaded, ...
         LNKnet_fid] = print_to_file(input_file, ...
                                     LNKnet_input_file, ...
                                     LNKnet_file_loaded, ...
                                     LNKnet_fid, ...
                                     tissue_value, ...
                                     num_rois, ...
                                     feature_called_flag, ...
                                     feature1_flag, ...
                                     feature2_flag, ...
                                     feature3_flag, ...
                                     feature4_flag, ...
                                     feature1_vector, ...
                                     feature2_vector, ...
                                     feature3_vector, ...
                                     feature4_vector)
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  print_to_file.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Prints feature extraction results in a LNKnet compatible input
%%                data file format.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\           PRINT TO FILE           /\/\/\/\/\/\/\/\/\/\/\%
%/\/\/\/\/\/\/\/\/\/\/\                                   /\/\/\/\/\/\/\/\/\/\/\%

  if strcmp(feature_called_flag, 'false')
    fprintf('\n')
    fprintf('Please run a feature extraction algorithm prior to selecting this option.\n')
  else
    quit_print = 'no';
    percent = '%';
    while(strcmp(quit_print, 'no'))
      print_option = menu('PRINT TO FILE SUBMENU (L4)', ...
                          'PRINT TO NEW LNKnet INPUT FILE', ...
                          'APPEND TO CURRENT LNKnet INPUT FILE', ...
                          'RETURN TO PREVIOUS MENU');
```

print_to_file.m

```
%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^      PRINT TO NEW LNKnet INPUT FILE      ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%


     if print_option == 1
       LNKnet_input_file = ...
         input('Please enter the name of the LNKnet input file: ', 's');
       LNKnet_input_file = (LNKnet_input_file(LNKnet_input_file ~= ' '));
       LNKnet_file_loaded = 'true';
       LNKnet_fid = fopen(LNKnet_input_file, 'w');
       for index = 1 : 1 : num_rois
         if strcmp(feature1_flag, 'true')
           vector1 = num2str(feature1_vector(index));
         else
           vector1 = '';
         end
         if strcmp(feature2_flag, 'true')
           vector2 = num2str(feature2_vector(index));
         else
           vector2 = '';
         end
         if strcmp(feature3_flag, 'true')
           vector3 = num2str(feature3_vector(index));
         else
           vector3 = '';
         end
         if strcmp(feature4_flag, 'true')
           vector4 = num2str(feature4_vector(index));
         else
           vector4 = '';
         end
         fprintf(LNKnet_fid, '%s     %s          %s          %s          %s', ...
                 tissue_value, vector1, vector2, vector3, vector4);
         fprintf(LNKnet_fid, blanks(10));
         fprintf(LNKnet_fid, '%s', percent);
         fprintf(LNKnet_fid, blanks(1));
         fprintf(LNKnet_fid, input_file);
         fprintf(LNKnet_fid, '\n');
       end
       fprintf('\n')
       fprintf('Feature extraction results saved to ')
       fprintf(LNKnet_input_file)
       fprintf('\n')
```

```
%^^^^^^^^^^^^^^^^^^^^                                   ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^  APPEND TO CURRENT LNKnet INPUT FILE  ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^                                   ^^^^^^^^^^^^^^^^^^^^%


      elseif print_option == 2
        if ~strcmp(LNKnet_file_loaded, 'true')
          fprintf('\n')
          fprintf('PRINT TO NEW LNKnet INPUT FILE must be selected first.\n')
        else
          for index = 1 : 1 : num_rois
            if strcmp(feature1_flag, 'true')
              vector1 = num2str(feature1_vector(index));
            else
              vector1 = '';
            end
            if strcmp(feature2_flag, 'true')
              vector2 = num2str(feature2_vector(index));
            else
              vector2 = '';
            end
            if strcmp(feature3_flag, 'true')
              vector3 = num2str(feature3_vector(index));
            else
              vector3 = '';
            end
            if strcmp(feature4_flag, 'true')
              vector4 = num2str(feature4_vector(index));
            else
              vector4 = '';
            end
            fprintf(LNKnet_fid, '%s     %s          %s          %s          %s', ...
                    tissue_value, vector1, vector2, vector3, vector4);
            fprintf(LNKnet_fid, blanks(10));
            fprintf(LNKnet_fid, '%s', percent);
            fprintf(LNKnet_fid, blanks(1));
            fprintf(LNKnet_fid, input_file);
            fprintf(LNKnet_fid, '\n');
          end
          fprintf('\n')
          fprintf('Feature extraction results appended to ')
          fprintf(LNKnet_input_file)
          fprintf('\n')
        end
```

```
%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^        RETURN TO PREVIOUS MENU     ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%


     elseif print_option == 3
       quit_print = 'yes';
     end
   end
 end


%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^               END                  ^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^                                    ^^^^^^^^^^^^^^^^^^^^%
```

segmentation.m

```
function segmentation
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  segmentation.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Menu for segmentation routines added in the future.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>              SEGMENTATION               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%


  fprintf('\n')
  fprintf('Segmentation routines to be added at a later date.\n')


%>>>>>>>>>>                                                             <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  END                    <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                         <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                             <<<<<<<<<<%
```

classification.m

```
function classification
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%
%%  FILENAME:  classification.m
%%
%%  CREATION DATE:  09 Nov 94
%%
%%  AUTHOR:  1Lt John L. Kelley
%%
%%  DESIGN REFERENCE:  None
%%
%%  MATLAB VERSION:  4.2 (UNIX)
%%
%%  TOOLBOXES:  IMAGE PROCESSING
%%
%%  DESCRIPTION:  Menu for classification routines added in the future.
%%
%%  MODIFICATION HISTORY:  DD Mon YY - Author
%%                                     Modification bullet
%%                                     Modification bullet
%%                         DD Mon YY - Author
%%                                     Modification bullet
%%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%


%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>             CLASSIFICATION               <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%

  fprintf('\n')
  fprintf('Classification routines to be added at a later date.\n')


%>>>>>>>>>>                                                              <<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                  END                     <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>>>>>>>>>>>                                          <<<<<<<<<<<<<<<<<<<<%
%>>>>>>>>>>                                                              <<<<<<<<<<%
```

```
MATLAB Technical Note

Introduction to CMEX, Part I
----------------------------

Many people have requested a simple example on how to create a C MEX-
file.  In response to this request, the following C MEX-file, named
MEXAMPLE, is provided as an introduction to CMEX programming.  MEXAMPLE
is a commented program which describes how to use the following MEX-
functions:

   mexErrMsgTxt
   mxCreateFull
   mxGetM
   mxGetN
   mxGetPr
   mxIsComplex
   mxIsSparse
   mxIsString

In MATLAB, MEXAMPLE accepts two inputs and returns one output. The inputs
are a 2x2 matrix denoted as MATRIX_IN and a  2x1 vector denoted as
VECTOR_IN.  The function calculates the  determinant of MATRIX_IN,
multiplies each element of VECTOR_IN by the determinant, and returns this
as the output, denoted by  VECTOR_OUT.  All inputs and outputs to this
function are assumed to be real (not complex).

------------------------------------------------------------------------

/*      First, include some basic header files.  The header file "mex.h" is
        required for a MEX-file.  Add any other header files that your
        function may need here. */

#include "mex.h"

/*      A C MEX-file generally consists of two sections.  The first
        section is a function or set of functions which performs the
        actual mathematical calculation that the MEX-function is
        to carry out.  In this example, the function is called
        workFcn().  The second section is a gateway between MATLAB
        and the first section, and consists of a function called
        mexFunction.  The gateway is responsible for several tasks,
        including:

                  I)  error checking,
                 II)  allocating memory for return arguments,
                III)  converting data from MATLAB into a format that the
                      workFcn function can use, and vice versa.

        The first function to be written in this example, then, is workFcn:

        Since C and MATLAB handle two-dimensional arrays differently, we
        will explicitly declare the dimension of the variable theMatrix.  The
```



```
        variables, theVector and theResult, are both one-dimensional
        arrays, and therefore do not need such rigid typing.

        The #ifdef clause is intended so that both ANSI and non-ANSI
        compilers can compile this MEX-file. */

#ifdef __STDC__
void workFcn(
       double theMatrix[2][2],  /* matrix with data from MATRIX_IN */
       double theVector[],      /* vector with data from VECTOR_IN */
       double theResult[]       /* vector with data for VECTOR_OUT */
       )
#else
void workFcn(theMatrix,theVector,theResult)
       double theMatrix[2][2];  /* matrix with data from MATRIX_IN */
       double theVector[];      /* vector with data from VECTOR_IN */
       double theResult[];      /* vector with data for VECTOR_OUT */
#endif                          /* __STDC__ */
{
  double determinant;           /* determinant of theMatrix */

  determinant = theMatrix[0][0]*theMatrix[1][1]
                  -theMatrix[0][1]*theMatrix[1][0];

  theResult[0] = theVector[0]*determinant;
  theResult[1] = theVector[1]*determinant;
}

/*      Now, define the gateway function, i.e., mexFunction.  Below is the
        standard, predeclared header to mexFunction.  nlhs and nrhs are the
        number of left-hand and right-hand side arguments that MEXAMPLE
        was called with from within MATLAB.  In this example, nlhs equals 1
        and nrhs should equal 2.  If not, then the user has called MEXAMPLE
        the wrong way and should be informed of this.  plhs and prhs are
        arrays which contain the pointers to the MATLAB matrices, which
        are stored in a C struct called a Matrix.  prhs is an array of length
        nrhs, and its pointers point to valid input data.  plhs is an array of
        length nlhs, and its pointers point to invalid data (i.e., garbage).
        It is the job of mexFunction to fill plhs with valid data.

        First, define the following values.  This makes it much easier to
        change the order of inputs to MEXAMPLE, should we want to change
        the function later.  In addition, it makes the code easier to read. */

#define MATRIX_IN prhs[0]
#define VECTOR_IN prhs[1]
#define VECTOR_OUT plhs[0]

#ifdef __STDC__
void mexFunction(
        int     nlhs,
        Matrix  *plhs[],
        int     nrhs,
        Matrix  *prhs[]
        )
#else
```

```
mexFunction(nlhs, plhs, nrhs, prhs)
int nlhs, nrhs;
Matrix *plhs[], *prhs[];
#endif
{
double twoDarray[2][2];/* 2 dimensional C array to pass to workFcn() */
int    row,col;         /* loop indices */
int    m,n;                     /* temporary matrix size holders */

/*      Step 1: Error Checking
        Step 1a: is nlhs 1?  If not, generate an error message and exit
        MEXAMPLE (mexErrMsgTxt does this for us!) */
  if (nlhs!=1)
    mexErrMsgTxt("MEXAMPLE requires one output argument.");

/*      Step 1b: is nrhs 2? */
  if (nrhs!=2)
    mexErrMsgTxt("MEXAMPLE requires two input arguments, MATRIX_IN and \
VECTOR_IN.");

/*      Step 1c: Is MATRIX_IN a 2x2 numeric (nonstring), full (non-sparse),
        real (noncomplex) matrix? */
  if (mxGetM(MATRIX_IN)!=2 || mxGetN(MATRIX_IN)!=2 ||
      mxIsString(MATRIX_IN) || mxIsSparse(MATRIX_IN) ||
      mxIsComplex(MATRIX_IN))
      mexErrMsgTxt("First argument to MEXAMPLE must be a 2x2, full, \
numeric, real-valued matrix.");

/*      Step 1d: Is VECTOR_IN a 2x1 or 1x2 numeric, full, real-valued
        vector? */
  m=mxGetM(VECTOR_IN);  /* Assigning result of mxGetM and mxGetN to
                                variables */
  n=mxGetN(VECTOR_IN);  /* with simpler names makes for easier code
                                reading.  */
  if (!((m==2 && n==1) || (m==1 && n==2)) ||
      mxIsString(VECTOR_IN) || mxIsSparse(VECTOR_IN)  ||
      mxIsComplex(VECTOR_IN))
      mexErrMsgTxt("Second argument to MEXAMPLE must be 2 element, \
full, numeric, real-valued vector.");

/*      Step 2:  Allocate memory for return argument(s) */
  VECTOR_OUT = mxCreateFull(2, 1, REAL);        /* create a 2x1 full,
                                                    numeric, real-va…
…lued matrix
                                                    */

/*      Step 3:  Convert MATRIX_IN to a 2x2 C array

        MATLAB stores a two-dimensional matrix in memory as a one-
        dimensional array.  If the matrix is size MxN, then the first M
        elements of the one-dimensional array correspond to the first
        column of the matrix, and the next M elements correspond to the
        second column, etc. The following loop converts from MATLAB
        format to C format: */

  for (col=0; col < mxGetN(MATRIX_IN); col++)
     for (row=0; row < mxGetM(MATRIX_IN); row++)
```



```
        twoDarray[row][col] =
          (mxGetPr(MATRIX_IN))[row+col*mxGetM(MATRIX_IN)];
/*      mxGetPr returns a pointer to the real part of the matrix MATRIX_IN.
        In the line above, it is treated as the one-dimensional array
        mentioned in the previous comment.  */

/*      Step 4:  Call workFcn function */
  workFcn(twoDarray,mxGetPr(VECTOR_IN), mxGetPr(VECTOR_OUT));

/*      workFcn will fill VECTOR_OUT with the return values for MEXAMPLE,
        so the MEX-function is done!  To use it, compile it according to the
        instructions in the MATLAB External Interface Guide. Then, in
        MATLAB, type

        c=mexample([1 2; 3 4],[1 2])

        This will return the same as

        c=det([1 2; 3 4]) * [1 2]

*/

}
```

MATLAB Technical Note

Introduction to CMEX, Part II

The following MEX-file is part two in a series of tutorials intended to demonstrate how to write C MEX-files by way of extended examples. When run in MATLAB this function, MEXAMPL2, accepts one input, a (possibly complex) vector, and if the user requests it, returns one output scalar. The function performs the following actions:

1. creates a new figure
2. breaks the figure into two subplots
3. plots the magnitude and phase of the the vector in the two subplots
4. changes the title of the figure to the name of the variable being plotted.

If requested, the function returns the figure handle of the figure created in step one. The user would request this by supplying a return variable when calling MEXAMPL2 from within MATLAB (e.g., fig_handle=mexampl2(...);).

MEXAMPL2 calls the following MEX functions, and discusses their use in detail:

    mexCallMATLAB
    mexEvalString
    mxGetPi
    mxFreeMatrix
    mxCreateString
    mxGetName.

MEXAMPL2 also calls the following MEX functions, which are described more fully in MEXAMPLE, the first C MEX-files tutorial:

    mexErrMsgTxt
    mxGetM, mxGetN
    mxIsString
    mxIsSparse
    mxCreateFull
    mxGetPr

In addition, certain basic aspects of MATLAB Handle Graphics are briefly covered in this example.

```
/*      Include basic header files: */
#include <math.h>  /* for sqrt, fabs, atan, etc. */
#include <stdio.h> /* for NULL */
#include <mex.h>

/*      First, create two utility functions.  These functions accept two
```



```
        arrays of doubles (one representing the real part of a vector and the
        other representing the imaginary part of the vector) and return
        the magnitude and phase of these arrays.  The output data is stored
        in a third input array (whose memory has been preallocated), while a
        fourth input argument represents the length of the arrays. */
#ifdef __STDC__
void magnitude(
    double *realData,    /* real part of the input vector */
    double *imagData,    /* imaginary part of the input vector */
    double *magData,     /* magnitude of the input vector (return argument)
                                */
    int vectorSize       /* size of the input vector */
    )
#else
magnitude(realData,imagData,magData,vectorSize)
double *realData;        /* real part of the input vector */
double *imagData;        /* imaginary part of the input vector */
double *magData;         /* magnitude of the input vector (return argument)
                            */
int vectorSize;          /* size of the input vector */
#endif
{
int element;             /* loop counter to cycle through elements of the
                            input vector */

/*      if imagData is not NULL, loop through each element of the input
        vector.  For each element, store in magData the square root of the
        sum of the squares of the real and imaginary parts. */
    if (imagData != NULL)
        for (element=0;element<vectorSize;element++)
            magData[element] = sqrt(realData[element]*realData[element]+
                                    imagData[element]*imagData[element]);
    else                 /* otherwise imaginary part is zero, take a short
                            cut: */
        for (element=0;element<vectorSize;element++)
            magData[element] = fabs(realData[element]);

}                        /* END magnitude() */

#define PI 3.14159265358979344408920985006 26
#ifdef __STDC__
void phase(
    double *realData,    /* real part of the input vector */
    double *imagData,    /* imaginary part of the input vector */
    double *phaseData,   /* phase of the input vector (return argument) */
    int vectorSize       /* size of the input vector */
    )
#else
phase(realData,imagData,phaseData,vectorSize)
double *realData;        /* real part of the input vector */
double *imagData;        /* imaginary part of the input vector */
double *phaseData;       /* phase of the input vector (return argument) */
int vectorSize;          /* size of the input vector */
#endif
{
int element;             /* loop counter to cycle through elements of the
```

```
                                input vector */

/*      if imagData is not NULL, loop through each element of the input
        vector.  For each element, store in phaseData the inverse tangent
        of the imaginary part of the vector divided by the real part of the
        vector. */
  if (imagData != NULL)
    for (element=0;element<vectorSize;element++)
      phaseData[element] = atan(imagData[element]/realData[element]);
  else                  /* otherwise imaginary part is zero, take a short
                           cut: */
    for (element=0;element<vectorSize;element++)
      phaseData[element] = (realData[element] > 0 ? 0 : PI);
}                       /* END phase() */

/*      Now that the utility functions are defined, we can work on
        mexFunction itself.  But first, some definitions: */

#define VECTOR_IN prhs[0]
#define SCALAR_OUT plhs[0]

#ifdef __STDC__
void mexFunction(
    int         nlhs,
    Matrix      *plhs[],
    int         nrhs,
    Matrix      *prhs[]
    )
#else
mexFunction(nlhs, plhs, nrhs, prhs)
int nlhs, nrhs;
Matrix *plhs[], *prhs[];
#endif
{
/*      We will be making several calls to the function mexCallMATLAB.
        Two of that function's inputs are arrays of Matrix pointers.  The next
        two variables will be used when we make these calls.  None of the
        mexCallMATLAB calls use more than five input variables or result in
        more than 1 output variable, so MCM_inputs is declared as a 5-
        element array, and MCM_outputs is declared as a 1 element array.
        Optionally, if the maximum number of input or output variables were
        not known ahead of time, we could declare MCM_inputs and
        MCM_outputs to be of type Matrix **, and then allocate memory for
        the array using mxCalloc. */
Matrix * MCM_inputs[5]; /* Matrix pointer array for mexCallMATLAB
                           function */
Matrix * MCM_outputs[1];/* Matrix pointer array for mexCallMATLAB
                           function */
Matrix * figure_handle; /* handle to figure that MEXAMPL2 creates */
int      size_VECTOR_IN;/* number of elements in VECTOR_IN */

/*      Step 1: Input checking.
        Step 1a: Check nlhs.  Should be 0 or 1. */
  if (nlhs > 1)
    mexErrMsgTxt("MEXAMPL2 only returns up to 1 argument.");
```



```
/*      Step 1b: Check nrhs.  Should be 1. */
  if (nrhs != 1)
    mexErrMsgTxt("Requires 1 input argument.");

/*      Step 1c: check VECTOR_IN (prhs[0]).  Should be one-dimensional, full,
        and numeric.  Should not be sparse or a string. */
  if ( ((mxGetM(VECTOR_IN) != 1) && (mxGetN(VECTOR_IN) != 1)) ||
    mxIsString(VECTOR_IN) || mxIsSparse(VECTOR_IN) )
    mexErrMsgTxt("Input argument should be Nx1 or 1xN, \
full, and numeric.");

/*      For ease later on, we store the size of VECTOR_IN in the
        variable size_VECTOR_IN. */
size_VECTOR_IN = mxGetM(VECTOR_IN) * mxGetN(VECTOR_IN);

/*      Step 2: create figure and plot the plots.
        Step 2a: create the figure. In MATLAB, the figure is created with
        the figure command.  To do it here, we will use mexCallMATLAB to
        call the figure function.  The arguments to mexCallMATLAB are
        similar to those for mexFunction itself.  There is nlhs (the number
        of output arguments we are requesting), plhs (an array of pointers
        to the output arguments), nrhs (the number of input arguments), and
        prhs (an array of pointers to the input arguments).  In addition,
        there is a string which contains the name of the function we are
        going to call (in this case, figure).  The mexCallMATLAB call
        below, then, mimics the MATLAB command "figure_handle =
        figure;".  Note that in this case there are zero inputs to figure, so
        MCM_inputs does not need to be initialized. */
mexCallMATLAB(1,MCM_outputs,0,MCM_inputs,"figure");

/*      Save result (the figure handle) to SCALAR_OUT.  Since
        mexCallMATLAB reallocates memory for its second argument,
        plhs, each time it is called, we don't need to worry about a future
        mexCallMATLAB call writing over figure_handle.*/
figure_handle = MCM_outputs[0];

/*      Step 2b: Create each subplot, title it, and plot the corresponding
        data. This consists of calling mexEvalString to call subplot
        and title, and mexCallMATLAB to call plot.  We repeat this
        twice to plot the magnitude and phase.  mexEvalString is a simple
        function which passes its one input, a string, to MATLAB for
        parsing.  Thus it is similar to the eval command in MATLAB. */
mexEvalString("subplot(2,1,1);");       /* causes MATLAB to execute
                                           subplot(2,1,1); */

/*      Create a matrix to pass to plot as the data to be plotted.  */
MCM_inputs[0] = mxCreateFull(size_VECTOR_IN,1,REAL);

/*      Load this matrix with the magnitude of VECTOR_IN: */
magnitude(mxGetPr(VECTOR_IN),mxGetPi(VECTOR_IN),
mxGetPr(MCM_inputs[0]),size_VECTOR_IN);

/*      Call mexCallMATLAB: */
mexCallMATLAB(0,MCM_outputs,1,MCM_inputs,"plot");

/*      Title the plot: */
```

```
mexEvalString("title('Magnitude');");   /* Causes MATLAB to execute
                                           title('Magnitude'); */

/*      Do it again for imaginary part: First, the subplot command: */
mexEvalString("subplot(2,1,2);");

/*      Load the matrix with the phase of VECTOR_IN: */
phase(mxGetPr(VECTOR_IN),mxGetPi(VECTOR_IN),
mxGetPr(MCM_inputs[0]),size_VECTOR_IN);

/*      Call mexCallMATLAB: */
mexCallMATLAB(0,MCM_outputs,1,MCM_inputs,"plot");

/*      Title the plot: */
mexEvalString("title('Phase');");

/*      Free the matrix pointed to by MCM_inputs[0]. */
mxFreeMatrix(MCM_inputs[0]);

/*      For our second to last task, let's make the title of the figure be the
        name of the variable being plotted.  To do this, we need to set the
        figure's 'NumberTitle' property to 'off' and its 'Name' property to
        the name of the variable.  We do this with another call to
        mexCallMATLAB, which call the MATLAB set command.  To
        create the MATLAB strings 'NumberTitle', 'off', and 'Name', we
        use the function mxCreateString.  This function accepts a string
        input and returns a MATLAB string Matrix containing that string.
        Note that since mxGetName returns a string with the name of
        VECTOR_IN, we can pass its output directly to mxCreateString to
        create a MATLAB string containing the name of VECTOR_IN. */
MCM_inputs[0] = figure_handle;
MCM_inputs[1] = mxCreateString("NumberTitle");
MCM_inputs[2] = mxCreateString("off");
MCM_inputs[3] = mxCreateString("Name");
MCM_inputs[4] = mxCreateString(mxGetName(VECTOR_IN));

    mexCallMATLAB(0,MCM_outputs,5,MCM_inputs,"set");

/*      The above mexCallMATLAB call mimics the MATLAB command

        set(figure_handle,'NumberTitle','off', ...
        'Name',<name of VECTOR_IN>);

        Note, however, that using MEX-files allows users access to the name
        of the input variable, something which no ordinary M-file could do!
*/

/*      Finally, we need to determine what the value of SCALAR_OUT should
        be.  This depends on nlhs.  If nlhs equals 1, then the user
        requested we return the figure handle, and so SCALAR_OUT should be
        set to figure_handle.  If nlhs equals 0, no such request was made,
        and SCALAR_OUT should be NULL. */

if (nlhs)
        SCALAR_OUT = figure_handle;
else
```



```
        SCALAR_OUT = NULL;

}       /* END mexFunction() */
```

```
MATLAB Technical Note


Creating C++ MEX-files
----------------------

/*  The following example illustrates how to use C++ code with your
    C language MEX-file for MATLAB v4.x.  It makes use of member
    functions, constructors, destructors, and the iostream.

    The file is called example.cxx.  The routine simply defines
    a class, constructs a simple object, and displays the initial
    values of the internal variables.  It then sets the data members
    of the object based on the input given to the MEX-file and displays
    the changed values.

    The calling syntax is:    example( num1, num2 )

    The mexFunction definition, math.h, and mex.h are wrapped with extern
    "C" so that it has "C linkage" for the benefit of MATLAB.  Within the
    body of the mexFunction definition, full C++ syntax is allowed.

    At runtime you have to tell the  executable (in this case the output
    of cmex) where to find the libraries that resolve the C++ symbols.
    It needs to know this in order to go look for them at load time.
    Compile your MEX-file like this:

         g++ -c -I<directory containing mex.h> example.cxx
         cmex example.o -L/hub/lib/sun4 -lg++ -lgcc


    Jeffery Faneuff     April 1, 1994   <jeff@mathworks.com>
    Copyright (c) 1994 The Mathworks, Inc.

*/

#include <iostream.h>

extern "C" {
#include <math.h>
#include "mex.h"
}

/****************************/
class MyData {

public:
        void display();
        void set_data(double v1, double v2);
        MyData(double v1 = 0, double v2 = 0);
        ~MyData() { }
private:
        double val1, val2;
};

MyData::MyData(double v1, double v2) { val1 = v1; val2 = v2; }
```



```
void MyData::display() {
        cout << "Value1 = " << val1 << "\n";
        cout << "Value2 = " << val2 << "\n\n";
 }

void MyData::set_data(double v1, double v2) { val1 = v1; val2 = v2; }

/*********************/

static
  void example(  double num1, double num2 )
{
        cout << "\nThe initialized data in object:\n";
        MyData *d = new MyData; // Create a  MyData object
        d->display();           // It should be initialized to zeros
        d->set_data(num1,num2); // Set data members to incoming values
        cout << "After setting the object's data to your input:\n";
        d->display();           // Make sure the set_data() worked
        delete(d);
        return;
}

extern "C" {
  void mexFunction(
                  int          nlhs,
                  Matrix       *plhs[],
                  int          nrhs,
                  Matrix       *prhs[]
                  )
    {
     double      *vin1, *vin2;

      /* Check for proper number of arguments */

      if (nrhs != 2) {
        mexErrMsgTxt("EXAMPLE requires two input arguments.");
      } else if (nlhs >= 1) {
        mexErrMsgTxt("EXAMPLE requires no output argument.");
      }

      vin1 = mxGetPr(prhs[0]);
      vin2 = mxGetPr(prhs[1]);

      example(*vin1, *vin2);
      return;
    }
}                          /* end of extern "C" */
```

Appendix C
Lorad Digital Spot Mammogram Specifications

A. Digital Image Receptor

1. Size:	14" x 10" x 6"
2. Active Image Area:	50mm x 50mm
3. Image Device:	1024 x 1024 picture element (pixel) CCD, configurable as 512 x 512 array for increased sensitivity
4. Maximum Resolution:	20 pixels/mm in X and Y axis
5. Exposure Range:	0.1 to 5.0 seconds
6. Synchronization:	interlocked with x-ray control signal

B. Digital Spot Mammogram Computer

1. Microprocessor:	80486 @ 50 MHz
2. RAM:	16 Mb
3. Drive Controller:	32 bits
4. Fixed Disk Drive:	240Mb
5. Floppy Disk Drive:	1.44 Mb 3.5"
6. Optical Disk Drive:	600 Mb 5.25"

C. Digital Spot Mammogram Monitor

1. Monitor:	21" high resolution, monochrome
2. Display Area:	16" max horizontal, 12" max vertical
3. Horizontal Scan Rate:	48 to 108 kHz (automatic adjustment)
4. Vertical Scan Rate:	60 to 80 Hz (automatic adjustment)
5. Video Bandwidth:	200 MHz
6. Horizontal Resolution:	1024 to 2048
7. Vertical Resolution:	768 to 1536

D. System Features

1. 5cm x 5cm tissue image area
2. Operator control of display contrast and luminance window
3. Choice of 512 x 512 or 1024 x 1024 bit resolution display
4. Menu control of acquisition, display, storage retrieval, and processing functions
5. Large 21" monitor screen
6. Near real time display, 3 seconds from exposure to display in 512 mode (approximate)
7. RG50 BNC connector on back of Digital Spot Mammogram Computer
8. MP400 Series III Camera

Appendix D
Annotated Readings

[Anderson and Williams, 1992]
Anderson, Steven A. and Carla S. Williams. "Implementation of Neural Algorithms Within the Khoros System," *Proceedings of SPIE, Volume 1721, Third Workshop on Neural Networks: Academic/Industrial/NASA/Defense*, 417-424. Auburn University AL : The International Society for Optical Engineering, February 1992.

The paper provides an overview of how neural network programs developed in Fortran, back in 1990, were implemented in Khoros. This paper is tailored for a beginner: (1) introduces Khoros, (2) furnishes a "big picture" perspective of how algorithms are implemented in Khoros, (3) clarifies the role of the various utilities such as *composer*, *preview*, and *ghostwriter* in the implementation process, and (4) shows the *cantata* workspace and subform pane for the Zimmerman statistical clustering training algorithm.

[Astley and others, 1993]
Astley, Sue, Ian Hutt, Stephen Adamson, Peter Miller, Peter Rose, Caroline Boggis, Chris Taylor, Tim Valentine, Jack Davies, and Janette Armstrong. "Automation in Mammography: Computer Vision and Human Perception," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 716-730. San Jose CA : The International Society for Optical Engineering, February 1993.

This superb paper should be required reading for those creating a mammogram image processing environment. The paper includes a synopsis and critique of the research being conducted by the authors and a number of other groups in the detection and diagnosis of mammographic abnormalities, a description of an idealized and a realistic computer-based mammographic interpretation environment, an examination of a study to determine how certain image characteristics affect the ability of a radiologist to detect microcalcifications, a presentation on a study investigating the effects of prompting on detection performance, and a description of a new methodology that uses multiple x-ray views of biopsy specimens -- specifically microcalcification clusters -- to extract quantitative descriptions of mammographic abnormalities.

[Doi and others, 1990]
Doi, Kunio, Heber MacMahon, Shigehiko Katsuragawa, Heang-Ping Chan, Maryellen L. Giger, Kenneth R. Hoffmann, Nobuyuki Nakamori, Charles E. Metz, Hiroshi Fujita, Laura E. Fencil, and Carl J. Vyborny. "Utilization of Digital Image Data for Computer-Aided Diagnosis," *Proceedings of the First International Conference on Image Management and Communication in Patient Care*, 128-129. Washington D.C. : IEEE Computer Society, June 1989.

A paper paralleling [Doi and others, 1991] that discusses a little more in-depth six of the nine computer-aided diagnostic schemes being developed to identify and assess abnormalities in chest radiographs, mammograms, and angiograms.

[Doi, 1991a]
Doi, Kunio. "Technology Requirements for Computer-Aided Diagnosis," *Proceedings, Technology Requirements for Biomedical Imaging - Strategic Defense Initiative Technology Applications Symposium*, 127-130. Los Alamitos CA : IEEE Computer Society, May 1991.

A paper similar to [Doi and others, 1991], but with the scope of the discussion constrained to the computer-aided diagnostic schemes being developed in chest radiography. The technical limitations that must be overcome if computer-aided diagnosis is to one day become a "second opinion" tool of the radiologist are touched upon.

[Doi, 1991b]
Doi, Kunio. "Computer-Aided Diagnosis (CAD) Schemes Based on Quantitative Analysis of Digital Radiographic Images," *ICRU News*, 4-9 (June 1991).

A paper resembling [Doi and others, 1991], but includes a brief discussion of the development of computer-aided diagnosis.

[Doi and others, 1991]
Doi, K., H. MacMahon, S. Katsuragawa, M. L. Giger, K. R. Hoffmann, C. E. Metz, R. M. Nishikawa, H. Yoshimura, S. Sanada, X. Chen, R. A. Schmidt, S. M. Montner, T. Matsumoto, K. G. Chua, and C. J. Vyborny. "Computer-Aided Diagnosis: Present and Future," *Proceedings of the Tutorial Session in the World Congress on Medical Physics and Biological Engineering*, 59-66. Kyoto Japan : Elsevier Science Publishers B. V., July 1991.

The Kurt Rossmann Laboratories for Radiologic Image Research is developing computer-aided diagnostic schemes to identify and assess abnormalities in chest radiographs, mammograms, and angiograms. In mammography, the detection of clustered microcalcifications and masses is being studied. The techniques used for the detection of microcalcifications include linear or nonlinear processing and gray level thresholding. This method achieves an 87% true-positive detection rate with an average of four false-positive detections per image. Nonlinear bilateral subtraction and feature extraction techniques are used to identify masses. This scheme achieves a 95% true-positive detection rate with an average of three false-positive detections per image. Very little information will be sacrificed if this paper is read in place of [Doi, 1991a], [Doi, 1991b], [Doi and others, 1990], and [Doi and others, 1993b].

[Doi and others, 1993a]
Doi, K., M. L. Giger, R. M. Nishikawa, K. R. Hoffmann, H. MacMahon, R. A. Schmidt, and K. G. Chua. "Digital Radiography," *Acta Radiologica*, 34 : 426-439 (1993).

The best of Doi's papers that not only includes an in-depth discussion of six of the nine computerized techniques being developed to identify and assess abnormalities in chest radiographs, mammograms, and angiograms, but includes (1) a brief discussion on digital radiography, differential diagnosis of breast lesions, and Kundel's research identifying three of the components responsible for false-negative radiologic perceptual errors; (2) a background in mammography in regards to epidemiology, radiographic anatomy of microcalcifications and the technique used in their identification by a radiologist, and statistics on missed detections and possible causes; and (3) an impressive bibliography with 137 references.

[Doi and others, 1993b]
Doi, K., H. MacMahon, S. Katsuragawa, M. Giger, K. Hoffmann, R. Nishikawa, S. Sanada, R. Schmidt, F. Behlen, and D. Sluis. "Development of Digital Processing Techniques for Computer-Aided Diagnosis in Radiographic Images," *Proceedings of the Third International Conference on Image Management and Communication in Patient Care: Requirements, Implementation, and Assessment of Digital Imaging in Medicine*, 110-115. Berlin Germany : IEEE Computer Society, June 1993.

A paper similar to [Doi and others, 1991], but includes in tabular format the performance levels of the computer-aided diagnostic schemes in chest radiography and mammography, and a discussion of the lab's six-monitor viewstation being used to display computer-aided diagnostic results.

[Donohoe, 1993]
Donohoe, Gregory W. *Image Processing with Khoros - Introduction.* Albuquerque: Khoral Research, Inc., 1993.

Exceptional, must have, Khoros short course notes in slide format organized into the following categories: introduction, data representation, image generation and display, basic image processing, spatial filters, transform filters, image analysis, program control, writing Vroutines, labs, Khoros 2.0 technical overview, and frequently asked questions and answers. Highly recommended the novice user step through the labs to gain a better understanding and appreciation of the capabilities of this scientific visualization environment.

[Dougherty and Giardina, 1987]
Dougherty, Edward R. and Charles R. Giardina *Matrix Structured Image Processing.* Englewood Cliffs NJ : Prentice-Hall, Inc., 1987.

The ideal book for anybody writing image processing algorithms for the first time, especially in MATLAB. The book is more application oriented, filled with illustrative examples rather than mathematical proofs. The topics covered include image processing operations, gray level processing, edge detection, morphology, topological operations, transform techniques, and image architecture.

[Dronkers, 1992]
Dronkers, Daniel J. "Stereotaxic Core Biopsy of Breast Lesions," *Radiology*, 183 : 631-634 (June 1992).

A study was conducted to assess the accuracy of using stereotaxic core biopsy to obtain breast tissue specimens. The one-year research was conducted using 70 selected patients with suspect breast lesions at mammography. The results, which include a comparison between this technique and surgical biopsy, along with the procedure used to obtain the breast tissue specimens are examined. From the results, a conclusion is drawn that stereotaxic core biopsy is an alternative to surgical biopsy.

[Etter, 1993]
Etter, D. M. *Engineering Problem Solving with MATLAB*. Englewood Cliffs NJ : Prentice-Hall, Inc., 1993.

A book that should be in the library of any engineer that frequently uses MATLAB. The book is organized into three main sections : (1) The *Introduction* provides an introduction to problem solving and to MATLAB. (2) *Fundamental Engineering Computations* covers scalar and array computations, control flow, statistical measurements, matrix computations, and plotting. (3) Finally, *Numerical Techniques* examines solutions to systems of linear equations, interpolation and curve fitting, polynomial analysis, numerical integration and differentiation, ordinary differential equations, matrix decomposition and factorization, signal processing, and control systems. Also included as appendices in the book is a quick reference to MATLAB functions, references cited in the text, solutions to selected problems, a discussion on reading and writing MAT files, and an explanation of using MATLAB on MS-DOS personal computers and Macintosh computers. The book comes with a 3.5" MS-DOS format floppy diskette, containing all of the text examples and data files.

[Gale and others, 1993]
Gale, A. G., A. R. M. Wilson, and E. J. Roebuck. "Mammographic Screening: Radiological Performance as a Precursor to Image Processing," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 458-464. San Jose CA : The International Society for Optical Engineering, February 1993.

This paper begins with an overview of England's breast screening system, followed by an examination of the recently established national self-assessment program for mammographic screening radiologists. This important program not only provides screening radiologists feedback on their performance -- individual and in comparison with their peers, but also significant factors that need to be accounted for in the implementation of any mammographic, computer-aided diagnostic system: "(1) the overall reader variability in mammographic screening, (2) the difficulty in identifying key mammographic features, and (3) the identification of particular cases which radiologists find difficult to interpret."

[Giger, 1993a]
Giger, Maryellen L. "Computer-Aided Diagnosis in Digital Mammography," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 46-47. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

A paper tailored for the individual desiring a nontechnical synopsis on computer-aided diagnosis in mammography. The paper touches upon the purpose of computer-aided diagnosis, computerized detection and classification methods, and areas warranting further work.

[Giger, 1993b]
Giger, Maryellen L. "Computer-Aided Diagnosis," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

The first three papers that should be read for an individual just entering this field are this one, [Giger, 1993a], and [Astley and others, 1993]. This paper is extremely helpful since it provides a synopsis of the research being conducted by a number of groups, including the Kurt Rossmann Laboratories for Radiologic Image Research, in the development of (1) computerized methods to detect microcalcifications, masses, and parenchymal patterns in mammograms, and (2) classification methods for integrating the computer or human extracted features into the probability of malignancy.

[Giger and others, 1990]
Giger, M., F. Yin, and K. Doi. "Image Features of Mammographic Masses Used in the Development of Computerized Schemes," *Proceedings of the 10th Conference on Computer Applications in Radiology and 4th Conference on Computer Assisted Radiology - S/CAR 90 Computer Applications to Assist Radiology*, 471-473. Carlsbad CA : Symposia Foundation Publishers, 1990.

Nonlinear bilateral subtraction and degree of spiculation are two methods being investigated at the Kurt Rossmann Laboratories for Radiologic Image Research to detect masses in mammograms. A 95% true-positive detection rate with an average of three false-positive detections per image is given for the nonlinear bilateral subtraction technique. No hard figures are given for the degree of spiculation method.

[Giger and others, 1993a]
Giger, M. L., K. Doi, F. F. Yin, H. MacMahon, C. E. Metz, R. A. Schmidt, S. M. Montner, H. Yoshimura, and C. J. Vyborny. "Computer-Vision Schemes for Lung and Breast Cancer Detection," *Proceedings of the 2nd International Conference on Visual Search*, 225-230. Taylor and Francis Ltd, 1993.

At the Kurt Rossmann Laboratories for Radiologic Image Research, computer-aided diagnostic schemes are being developed for the detection of masses in mammograms and pulmonary nodules in chest radiographs. One of the techniques used for the detection of masses includes non-linear subtraction (gray level thresholding, run length analysis) followed by feature extraction (morphological filtering, size and border testing). This method achieves a 95% true-positive detection rate with an average of three false-positive detections per image. Degree of spiculation (region growing, margin extraction and smoothing, standard deviation of fluctuations, morphological opening) is another technique used to differentiate malignant from benign masses. This scheme achieves a 97% malignant classification sensitivity with a 79% false-positive rate. The introductory paragraph captivates the reader with a listing of why diagnostic errors are made by radiologists.

[Giger and Vyborny, 1993]
Giger, Maryellen L. and Carl J. Vyborny. "Computers Aid Diagnosis of Breast Abnormalities," *Diagnostic Imaging*, 98-102 (June 1993).

As this paper highlights, mammography is an inexact science needing much improvement in the area of diagnostic accuracy. The work being conducted at the Kurt Rossmann Laboratories for Radiologic Image Research and by other researchers in the field of computer-aided diagnosis in mammography is the focus of this paper. The detection schemes being investigated by various researchers, including the results, are quite diverse. The paper provides a brief historical perspective of computer-aided diagnosis in mammography, along with a discussion of its introduction into radiology, and its role in the future.

[Giles and others, 1993]
Giles, A., A. R. Cowen, and G. J. S. Parkin. "A Clinical Workstation for Digital Mammography," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 806-817. San Jose CA : The International Society for Optical Engineering, February 1993.

FAXIL was tasked with the design and development of a prototype clinical workstation for digital mammography following the submission of the Forrest Report (see [Gale and others, 1993]) in 1986 on breast cancer screening in England. For those planning to design a clinical workstation or even just the software environment, this paper may provide helpful design tips since it includes a schematic of the main components of the workstation along with a discussion of the system architecture.

[Hendrick and Parker, 1993]
Hendrick, R. Edward and Steve H. Parker. "Stereotaxic Imaging," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

If a solid introduction to the stereotaxic core biopsy imaging system is desired, this paper along with [Parker and others, 1990] should be read. The paper starts off with a stimulating discussion on the development of stereoscopic acquisition, how it was held in disfavor in the early years to how interest has been rejuvenated with its use as an expedient, cost-effective, and dependable biopsy alternative that does not cause scarring and minimizes the potential for complications and disfigurement. The paper then goes on to detail, with helpful figures, the principles of stereotaxic localization and core needle biopsy, and the step-by-step method used to obtain a core specimen of breast tissue. The discussion does include mathematical equations; fortunately, only on a geometric/trigonometric level. The paper concludes with an examination of errors that can cause damage to the system or lesions to be missed at biopsy, two crucial quality control tests that need to be performed prior to using the system, and associated technological developments.

[Jordan and others, 1993]
Jordan, Ramiro and The Khoros Group. "Khoros - A Software Development Environment for Data Processing," *DSP Applications*, 2 (3) : 16-23 (March 1993).

This paper presents an overview of Khoros, discussing its six principal subsystems and use in a variety of applications. The Khoros infrastructure consists of (1) an extensible visual programming language, (2) a user interface and software development system, (3) an interoperable data and visualization exchange format, (4) application specific data processing libraries, (5) meta-system calls used for distributed computing, and (6) interactive visualization tools tailorable to fit specific applications. The applications examined include segmentation of ultrasound images, spatial analysis of vegetation, Magnetic Resonance Spectroscopy, ion beam dwell time calculation, and surface rendering of terrain. The images and *cantata* workspace (partially covered) for each application are included. Highly recommend this paper be read prior to or in conjunction with [Donohoe, 1993].

[Kaplan and others, 1993]
Kaplan, Gadi, Kenneth R. Foster, Ken Kornbluh, and Michael J. Riezenman. "Focus Report: Engineering Software," *IEEE Spectrum*, 30 (11) : 40-87 (November 1993).

A paper actually composed of five articles and three long tables that will help to clarify what software environments could or should be used in the areas of symbolic and numerical math, data acquisition, and data analysis and visualization. The software packages from over 110 companies are detailed in three tables that include the following columns: package and price, platform, requirements, brief product description, recent enhancements, and comments and features. A few, but extremely useful Internet sites are included such as (1) csi.jpl.nasa.gov that contains a frequently asked questions file on MATLAB in the pub/matlab directory, and (2) ftp.mathworks.com that permits sharing of files and access to MathWorks product information.

[Kegelmeyer, 1992]
Kegelmeyer, Jr., W. Philip. "Computer Detection of Stellate Lesions in Mammograms," *Proceedings of SPIE, Volume 1660, Biomedical Image Processing and Three-Dimensional Microscopy*, 446-454. San Jose CA : The International Society for Optical Engineering, February 1992.

The detection of stellate lesions in digitized mammograms using a binary decision tree, spatial filtering, and the features Analysis of Local Oriented Edges (ALOE) and Laws texture energy measures is the focus of this paper. The paper mentions Khoros as one of two image processing environments used in the research, but does not explicitly state that the glyph *vtexture* can be used to compute the twenty five Laws texture energy measures. The database for this study included the use of only ten test images -- half of them used as the training set -- with each containing at least one stellate lesion. The results yielded a sensitivity of 83% with 0.6 false positives per image. An overview of the entire algorithmic process as well as for the ALOE is provided.

[Kegelmeyer, 1993a]
Kegelmeyer, Jr., W. Philip. "Evaluation of Stellate Lesion Detection in a Standard Mammogram Data Set," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 787-798. San Jose CA : The International Society for Optical Engineering, February 1993.

The study conducted previously in [Kegelmeyer, 1992] to detect stellate lesions in digitized mammograms using a computerized diagnostic scheme was expanded by using sixty two mammograms -- including twelve with stellate lesions, and a revised spatial filtering method. A receiver operating characteristic analysis of the results yields a 92% probability of detection at a false alarm rate of 0 per image and a 100% probability of detection at a false alarm rate of 0.27 per image. The mammogram database, consisting of one hundred 800 x 1040 8-bit images, is available to the public courtesy of the University of South Florida and can be accessed on the Internet at figment.csee.usf.edu.

[Kegelmeyer, 1993b]
Kegelmeyer, Jr., W. Philip. "Evaluation of Stellate Lesion Detection in a Standard Mammogram Data Set," *International Journal of Pattern Recognition and Artificial Intelligence*, 7 (6) : 1477-1492 (December 1993).

A composite of [Kegelmeyer, 1992] and [Kegelmeyer, 1993a] with a couple of additional figures and slightly expanded discussion of some of the sections.

[Kegelmeyer and others, 1994]
Kegelmeyer, Jr., W. Philip, Joe M. Pruneda, Philip D. Bourland, Argye Hillis, Mark W. Riggs, and Michael L. Nipper. "Computer-Aided Mammographic Screening for Spiculated Lesions," *Radiology*, 191 (2) : 331-337 (May 1994).

The study conducted previously in [Kegelmeyer, 1992] and [Kegelmeyer, 1993a] to detect stellate lesions (now called spiculated lesions in this paper) in digitized mammograms using a computerized diagnostic scheme was expanded by using eighty five cases of four-view mammograms -- thirty six cases with confirmed spiculated lesions, four radiologists, a much more comprehensive screening study and statistical analysis, and slightly refined computer reports. The results yielded a per-case sensitivity of 100%, with 0.28 false positives per image. Use of the computer reports increased the average radiologist sensitivity by 9.7%, with no decrease in average specificity.

[Kelly and White, 1993]
Kelly, Patrick M. and James M. White. "Manipulating Remotely-Sensed Data with Khoros," *Proceedings of the 2nd International AVS User Group Conference, Volume 2*, 81-87. Lake Buena Vista FL : May 1993.

Khoros is being used in two remote-sensing applications: (1) Identification of urban areas from Landsat Thematic Mapper data using Laws texture energy measures and a one-class classifier in conjunction with an adaptive algorithm. (2) Clustering of Landsat Thematic Mapper data, followed by classification and analysis with the SPECTRUM program -- a Khoros toolbox. The one-class classifier, which uses hyperellipsoidal decision surfaces, warrants closer scrutiny since according to the authors it ". . . can produce classification systems that are more robust than traditional neural network techniques, . . ." and ". . . always generates closed decision boundaries." The adaptive algorithm, called Learning Vector Quantization - Mahalanobis Metric, minimizes the classification error for training data and also warrants closer examination. Although not mentioned in the paper, SPECTRUM can be downloaded from the pub/khoros/toolbox/classify directory of the Khoros ftp site ftp.eece.unm.edu.

[Koechner and others, 1990]
Koechner, Donna, John Rasure, Richard Griffey, and Tom Sauer. "Clustering and Classification of Multispectral Magnetic Resonance Images," *Proceedings of the Third Annual IEEE Symposium on Computer-Based Medical Systems*, 32-37. Chapel Hill NC : IEEE Computer Society, 1990.

A clustering and classification system is being researched to automatically identify different metabolic states in chemical shift images of human tissue. Good results were obtained when the classification system was tested on phantom images, potentially leading to its use in the automatic identification of pathological conditions. The majority of this system -- consisting of image acquisition, preprocessing, feature extraction, and unsupervised and supervised classification stages -- was developed using Khoros.

[Krupinski and Roehrig, 1994]
Krupinski, Elizabeth and Hans Roehrig. "Comparison of Digital X-ray Cameras for Stereotactic Breast Needle Biopsy: An Observer Performance Study," *Proceedings of the Second International Workshop on Digital Mammography*. York England, July 1994.

Observer detection performance using Lorad's lens-coupled digital mammography system and Fischer's fiberoptic-coupled digital mammography system were compared using a CDMAM-Phantom; various kV and mAs levels; different plexiglass plate thickness; randomized image, condition, and system orders; and six observers. Based on this study, the Lorad system outperformed the Fischer system. The report also touches upon some of the hardware aspects of both systems and lists the advantages of digital imaging systems versus conventional film radiology.

[Kubica and others, 1993]
Kubica, Steven, Benjamin S. Brann IV, John Rasure, and Tom Sauer. "An Automatic Segmentation System for Ultrasound Images Developed Using the Khoros Environment," *Proceedings of the 2nd International AVS User Group Conference, Volume 2*, 81-87. Lake Buena Vista FL : May 1993.

Ultra is a software application that calculates from a radial series of ultrasound images, the ventricular volume in preterm infants suffering from an accumulation of cerebrospinal fluid. A significant portion of this application, namely the segmentation routines and the user interface, are built from Khoros. The paper includes a flow diagram for the segmentation routine, with each process discussed in brief. Ultra is currently being used at the University of New Mexico Hospital, Department of Pediatrics, but further refinements are needed in the segmentation routine.

[Liberman and others, 1994]
Liberman, Laura, W. Phil Evans III, D. David Dershaw, Lucy E. Hann, Beth M. Deutch, Andrea F. Abramson, and Paul Peter Rosen. "Radiography of Microcalcifications in Stereotaxic Mammary Core Biopsy Specimens," *Radiology*, 190 : 223-225 (January 1994).

A study was conducted to assess specimen radiography in terms of its ability to detect and diagnose microcalcifications in stereotaxically guided core biopsies of breast tissue. The eight-month research was conducted using 361 breast core biopsies of 72 nonpalpable, microcalcification containing lesions from 65 patients. After radiography, the breast tissue specimen slides were sent to pathology. The histologic detection rate is given for those specimens in which microcalcifications were detected in radiography and for those specimens in which microcalcifications were not detected in radiography. Interestingly, the histologic detection rate without the use of specimen radiography is not addressed.

[Masters, 1994]
Masters, Timothy. *Signal and Image Processing with Neural Networks - A C++ Sourcebook.* New York NY : John Wiley & Sons, Inc., 1994.

This text covers (1) the design and training of complex domain, multiple layer feedforward networks, and (2) some of the traditional signal and image processing algorithms, especially those suited for integration with these networks. The book comes with a 3.5" MS-DOS format floppy diskette, containing commented C++ source code for a complex domain, multiple layer feedforward network program; data and sample validation command files to test the program; and moment, tone/texture, Fourier, Morlet wavelet, and Gabor transform subroutines. The examples in the text are also written in C++ and include comments. This text was published and obtained in the late phase of this thesis, but is worthy of close scrutiny by researchers in this field. Recapitulating the author, complex domain networks versus real domain networks generalize better, almost always train faster, are more resistant to becoming trapped in local minima, and have significantly lower error. The text is, according to the author once again, "... the only detailed description available in print on standard multiple layer feedforward networks generalized to the complex domain."

[Miasek and Lin, 1992]
Miasek, William and Frank Lin. "Diagnosis of HIV Infection by Neural Network: A Prototype," *Proceedings of SPIE, Volume 1721, Third Workshop on Neural Networks: Academic/Industrial/NASA/Defense*, 311-316. Auburn University AL : The International Society for Optical Engineering, February 1992.

Used a two hidden layer neural network with twenty four input nodes corresponding to clinical symptoms and 15 output nodes corresponding to various diseases -- the program output provides, with a probability of certainty, the disease diagnosis based upon the given symptoms. Authors discuss their experimental approach, as well as that of other notable researchers in the field of neural networks, in using known design principles to select the optimal neural network configuration for training.

[Miller and Astley, 1992]
Miller, Peter and Sue Astley. "Classification of Breast Tissue by Texture Analysis," *Image and Vision Computing*, 10 (5) : 277-282 (June 1992).

Laws texture energy measures, granulometric analysis, and associated algorithms are being used to classify glandular and fatty tissue in mammograms. The technique achieves an 80% correct discrimination rate, with Laws texture energy measures outperforming granulometric analysis. This is a unique paper considering the majority of writings on computer-aided diagnosis in mammography tend to focus on the detection, localization, and quantification of either masses or microcalcifications. Texture analysis has two significant potential applications: (1) The technique may allow for the early detection of breast cancer in asymptomatic women when used in combination with risk factors, like family history. (2) The risk for developing cancer from mammography is linearly influenced by the x-ray dose to composition of glandular tissue within the breast; therefore, the technique when used in conjunction with other variables, like compressed breast thickness, may provide risk estimates that could dictate the frequency and type of screening procedures performed.

[Moler, 1992]
Moler, Cleve. *The Student Edition of MATLAB - Student User Guide*. Natick MA : The MathWorks, Inc., 1992.

A handy guide for an inexperienced MATLAB user to have as a supplement to the professional edition. The book is organized into five main sections : (1) *Getting Started* includes notes to the student and about the guide, quick start instructions, and using MATLAB on MS-DOS personal computers and Macintosh computers. (2) The *Tutorial* covers fundamental, matrix, and array operations; vectors and matrix manipulation; data analysis; matrix and function functions; polynomials and signal processing; graphing; control flow; and M and disk files. (3) *Signals and Systems* explains a number of functions available in the Signal Processing Toolbox and Control System Toolbox. (4) The *Reference* includes boxes containing functions within the same subject area and an alphabetized listing of the functions, including a description for each. (5) *Appendices* consists of a listing of the references cited in the book, and discussions on using MATLAB on the Macintosh, reading and writing MAT files, and printing on the personal computer with professional MATLAB.

[Myler and Weeks, 1993]
Myler, Harley R. and Arthur R. Weeks. *Computer Imaging Recipes in C.* Englewood Cliffs NJ : P T R Prentice-Hall, Inc., 1993.

A good primer for those desiring an introduction to basic digital image processing algorithms and techniques. The topics covered include image characterization, storage, and display; image algebra and geometry; histogram techniques; image measurement; color; spatial and adaptive filters; morphological and spatial frequency filtering; and the UCFImage computer imaging software system. The book comes with a 3.5" MS-DOS format floppy diskette, containing the menu-based UCFImage program that performs file manipulations, algebraic and logical operations, geometric transformations, grayscale profile and measurements, object identification, histogram equalization, spatial and frequency filtering, display palette modification, and noise generation. The program is plain, needing improvement especially in the area of user-friendliness. One of the appendices contains a simple discussion on implementing in C a Laplacian edge detector according to a predefined programming strategy. This strategy accounts for the input and storage, representation, and output/display of the image.

[Parker and others, 1990]
Parker, Steve H., Jeffrey D. Lovin, William E. Jobe, Brian J. Burke, Kenneth D. Hopper, James M. Luethke, and Wayne F. Yakes. "Stereotactic Breast Biopsy with a Biopsy Gun," *Radiology*, 176 : 741-747 (September 1990).

A study was conducted to compare stereotactic core biopsies with corresponding surgical biopsies using 103 patients with nonpalpable, mammographically suspicious lesions, two different stereotactic mammography systems, and needles of various sizes. The histology of the stereotactic core and surgical biopsy specimens yielded agreement in 89 of the 102 cases (the stereotactic core specimen from one patient was lost), including 14 of 16 malignancies. This paper is a standout, containing a concise description of the procedure used to biopsy a breast lesion with a Fischer Mammotest Stereotactic System, a justification for the disagreement of the histology in fourteen of the cases, and a stimulating detailed discussion on the virtues of stereotactic core biopsies and the shortcomings of fine needle aspiration biopsy and surgical biopsy.

[Parker and others, 1991]
Parker, Steve H., Jeffrey D. Lovin, William E. Jobe, Brian J. Burke, Kenneth D. Hopper, and Wayne F. Yakes. "Nonpalpable Breast Lesions: Stereotactic Automated Large-Core Biopsies," *Radiology*, 180 : 403-407 (August 1991).

The study conducted previously in [Parker and others, 1990] to compare stereotactic core biopsies with corresponding surgical biopsies was expanded by using an additional 102 patients with nonpalpable, mammographically suspicious lesions and only 14-gauge needles. The histology of the stereotactic core and surgical biopsy specimens yielded agreement in 98 of the 102 cases, including 22 of 23 malignancies. From the results, a conclusion is drawn that for women with mammographically suspicious lesions, stereotactic core biopsy can be an alternative to surgical biopsy. A discussion is included covering such topics as the shortcomings of the previous study, two important considerations when using stereotactic core biopsy, the adequacy of specimen sampling in fine needle aspiration biopsy versus stereotactic core biopsy, and the advantages of stereotactic core biopsy. Interestingly, a comment is made that specimen radiography to detect microcalcifications in stereotactic core specimens -- as studied in [Liberman and others, 1994] -- is an unnecessary expenditure of time and effort. The basis for this comment is provided.

[Pietikainen and others, 1982]
Pietikainen, Matti, Azriel Rosenfeld, and Larry S. Davis. "Texture Classification Using Averages of Local Pattern Matches," *Proceedings of the 6th International Conference on Pattern Recognition, Volume 1*, 301-303. Munich Germany : IEEE Computer Society, October 1982.

A three part experiment was conducted to essentially test the texture classification ability of Laws masks. In the first part, each of four standard texture feature masks and different 3x3 and 5x5 Laws masks were convolved with twenty eight images, followed by a summation of the absolute values for each image. Laws features correctly classified anywhere from 67-90% of the images, while the standard features faired much more poorly. In the second part, the first experiment was repeated, but this time using modified 5x5 masks. A conclusion is drawn from the results that simple concentric spot masks and symmetric, centrally weighted edge masks may provide comparable results to Laws "standard" masks. In the final part, the first experiment was repeated, but this time using three different 5x5 Laws masks and local nonmaximum suppression of the convolutions. A conclusion is drawn from the results that the key variable for texture classification may be the maxima of the response.

[Rao, 1990]
Rao, A. Ravishankar. *A Taxonomy for Texture Description and Identification.* New York NY : Springer-Verlag, 1990.

For those planning to develop an expert learning system to analyze digital images using various texture features, this book along with [Tomita and Tsuji, 1990] should be read. Unlike [Tomita and Tsuji, 1990] though, who discuss the characterization of texture with conventional statistical and structural techniques, this book examines the characterization of texture through oriented patterns. The topics covered include computing oriented texture fields, analysis of oriented textures through phase portraits, analysis of strongly ordered textures, disordered textures, and compositional textures.

[Rasure and others, 1990a]
Rasure, John, Danielle Argiro, Tom Sauer, and Carla Williams. "Visual Language and Software Development Environment for Image Processing," *International Journal of Imaging Systems and Technology*, 2 : 183-199 (1990).

This paper begins with an overview of topics relevant to the Khoros environment: visual programming, software development environments, automatic generation of code and user interfaces, and image processing systems. Next, the Khoros environment is examined in detail, specifically the high-level user interface and code generation system, the *cantata* visual programming language, and the interoperable data exchange format and image processing library. The paper concludes using the ground level vegetative analysis application -- as discussed in [Rasure and others, 1990b] and [Sauer and others, 1991] -- to demonstrate the image processing capabilities of Khoros.

[Rasure and others, 1990b]
Rasure, John, Tom Sauer, and Charlie Gage. "A Workstation Based Image Acquisition and Processing Instrument for Spatial Analysis of Vegetation," *Proceedings of International Society for Photogrammetry and Remote Sensing, Commission V, Volume 1395, Close Range Photogrammetry Meets Machine Vision*, 526-533. Zurich Switzerland : International Society for Photogrammetry and Remote Sensing, Commission V, September 1990.

A paper similar to [Sauer and others, 1991], but with a few less figures and sentences, that discusses the analysis of near ground level images for the Sevilleta Long Term Ecological Research (LTER) project. The ultimate goal is to be able to correlate satellite and near ground level imagery.

[Richter and Claridge, 1993]
Richter, J. and E. Claridge. "Analysis of the Requirements for Quantitative Evaluation of Mammograms by Computers," *Proceedings of the International Symposium CAR' 93 Computer Assisted Radiology*, 618-623. 1993.

An enlightening paper covering the screening and symptomatic diagnosis of mammograms; image formation; use of object depiction, geometric, and brightness-related measurements; image enhancement; communication of results; and suggestions for the quantitative evaluation of mammograms. The measurement sections touch upon non-linear transformation of signals, breast compression, density measurements, and the effectiveness of the most common methods employed to detect lesions. The enhancement section identifies a number of schemes that can be used to draw the attention of the radiologist to diagnostic features. The paper should be mandatory reading for the beginner just entering this field.

[Rogers and others, 1994]
Rogers, Steven K., Dennis W. Ruck, and Matthew Kabrisky. "Artificial Neural Networks for Early Detection and Diagnosis of Cancer," *Cancer Letters*, 77 : 79-83 (1994).

A good introductory level paper on image processing and its use in computer-aided diagnosis. This paper provides (1) a brief historical background on neural networks; (2) an overview of segmentation, feature extraction, and classification; (3) an examination of the applicability of neural networks in these three image processing stages; and (4) a discussion of the current and future role neural networks will have in the detection and diagnosis of cancer.

[Romano, 1993]
Romano, Carol A. "Imaging: An Innovative Technology," *Computers in Nursing*, 11 (5) : 222-225 (September/October 1993).

This paper is tailored for the individual just entering or desiring an introduction to the field of digital image processing. An overview of the five major components of imaging technology -- acquisition, transmission, storage, enhancement, and analysis -- are furnished along with a description of the technology and a background. Surprisingly, considering the source of the journal, the examples provided in the *Applications and Implications* subheading encompasses a broad spectrum of fields rather than specifically to the medical domain.

[Rots and Herreid, 1992]
Rots, A. and L. Herreid. "Proto-Typing Astronomical Software in Khoros," *Astronomical Data Analysis Software and Systems I*, 25 : 145-147 (1992).

The paper provides a brief background on the X-ray Timing Explorer mission , succinctly discusses the requirements for a prototype astronomical software environment, lists what Khoros has to offer from the perspectives of a user and a software developer, and identifies the disadvantages of Khoros.

[Sauer and others, 1991]
Sauer, Tom, John Rasure, and Charlie Gage. "Near Ground Level Sensing for Spatial Analysis of Vegetation," *Multisource Data Integration in Remote Sensing, NASA Conference Publication 3099*. 11-25. University of Maryland MD : NASA Goddard Space Flight Center, June 1990.

Khoros is being used to conduct ground level vegetative analysis. The methodology being employed in the analysis of the near ground level images includes preprocessing, feature extraction, classification (with training and production phases), and region analysis. Although the algorithms for these techniques are discussed in this paper as well as in [Rasure and others, 1990a] and [Rasure and others, 1990b], and [Rasure and others, 1990a] includes a figure of the *cantata* workspace with the interconnection of each macro (technique), there is no mention of the *glyphs* that comprise these macros. The main *glyphs* used can be deduced by carefully reading the algorithms and referring to [Donohoe, 1993]. This paper, along with those mentioned previously, provide a good example of the potential of Khoros and why it warrants closer scrutiny.

[Schneeberger and others, 1992]
Schneeberger, Timothy, Robert Pierson, Dave Dayton, and John Gonglewski. "Image Recovery and Pattern Recognition Applications in a Visual Language Environment," *Astronomical Data Analysis Software and Systems I*, 25 : 140-144 (1992).

The paper provides a terse description of Khoros, and the role this visual language plays in a tracker simulation (pattern recognition application) and a self-referenced speckle holography system (astronomical image recovery application).

[Shtern, 1992]
Shtern, Faina. "Digital Mammography and Related Technologies: A Perspective from the National Cancer Institute," *Radiology*, 183 : 629-630 (June 1992).

This paper discusses the objectives, and outlines the research agenda, of the National Digital Mammography Development Group whose formation was proposed at a National Cancer Institute workshop on breast imaging. The group is to promote research in digital mammography along with its integration in image processing, computer-aided diagnosis, and teleradiology.

[Shtern and others, 1993]
Shtern, Faina, Sue Astley, David R. Dance, Nico Karssemeijer, Carolyn Kimme-Smith, Robert Nishikawa, Raj Rangayyan, Raman Paranjape, Liang Shen, and J. E. Leo Desautels. "Panel Discussion - Design of a Common Database for Research in Mammogram Image Analysis," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 535-551. San Jose CA : The International Society for Optical Engineering, February 1993.

A paper actually consisting of an introductory and six position papers submitted by some of the panel members deliberating the advantages, standard requirements, and use of a common database. These papers are written by well respected panelists with diverse backgrounds in the field of mammography and are highly recommended for those researching and developing mammographic image processing algorithms.

[The MathWorks, Inc., 1992a]
The MathWorks, Inc. *MATLAB User's Guide.* Natick MA : The MathWorks, Inc., 1992.

The guide is organized into two chapters: (1) *MATLAB for Unix Workstations* covers the system requirements, how to get started, the directory structure, available toolboxes, license manager administration, and how to use UNIX specific features of MATLAB. (2) The *Tutorial* introduces the basic features of MATLAB with descriptions and examples. The subheadings of this chapter include getting started, matrix and array operations, vector and matrix manipulations, data analysis, matrix functions, polynomials and signal processing, function functions, graphics, control flow, M and disk files, the debugger, file I/O, and sparse matrices.

[The MathWorks, Inc., 1992b]
The MathWorks, Inc. *MATLAB Reference Guide.* Natick MA : The MathWorks, Inc., 1992.

The guide includes (1) boxes containing functions, commands, operators, and debugging and language constructs within the same subject area; (2) a description of the operators; and (3) an alphabetized listing of the functions, commands, and debugging and language constructs, including a description for each.

[The MathWorks, Inc., 1993a]
The MathWorks, Inc. *MATLAB External Interface Guide.* Natick MA : The MathWorks, Inc., 1993.

To learn how to interface MATLAB to C or Fortran routines, this guide is an absolute necessity. The guide is organized into two chapters: (1) *External Interfaces to MATLAB* covers shell escape functions and MEX-files, importing and exporting data to and from MATLAB, and the MATLAB engine facility. (2) The *External Interface Library* includes boxes containing routines within the same class; a brief discussion of the files associated with the External Interface Library; and an alphabetized listing of the routines, including a description for each. Not mentioned in the guide is the ftp availability of C MEX-file function examples and information on how to write a C++ MEX-file function. This information can be obtained by downloading technical notes mxeng3.txt, mxeng4.txt, and mxeng5.txt located in the /pub/tech-support/tech-notes directory of the ftp site ftp.mathworks.com. Many C MEX-file examples can also be found in the $MATLAB/exter/src directory.

[The MathWorks, Inc., 1993b]
The MathWorks, Inc. *MATLAB Building a Graphical User Interface.* Natick MA : The MathWorks, Inc., 1993.

MATLAB provides a standard set of functions to build a graphical user interface for an application created within its environment, dramatically simplifying this otherwise daunting task. The types of graphical user elements that can be built include push buttons, check boxes, radio buttons, sliders, pop-up menus, static text, editable text, and frames. The guide includes a sample application and a discussion of advanced features.

[Thompson and Shure, 1993]
Thompson, Clay and Loren Shure. *Image Processing Toolbox.* Natick MA : The MathWorks, Inc., 1993.

The exceptionally written, easy-to-use guide to the MATLAB Image Processing Toolbox is organized into two chapters: (1) The *Tutorial* includes a description, example code, and accompanying figure for each function. Also included in this chapter is an introduction to image processing, covering image and file types, color and color maps, and image coordinate systems. (2) The *Reference* uses three subheadings -- purpose, synopsis, and description -- to detail each function. Many of the functions include additional subheadings and an accompanying figure. The toolbox supports the following image processing categories: image display; image conversion; color map operations; filtering operations; geometric, enhancement, and analysis operations; image transforms, and file I/O operations.

[Tomita and Tsuji, 1990]
Tomita, Fumiaki and Saburo Tsuji. *Computer Analysis of Visual Textures*. Boston MA : Kluwer Academic Publishers, 1990.

A really significant comment made on the back cover of this book is "Texture analysis is the first problem that computer vision faces in the real world, because every object has its own texture, and the result gives the basis for succeeding higher-order postprocessing of computer vision in both the two-dimensional and the three-dimensional world." This comment is motivation in itself for examining this book in greater detail. The topics covered include statistical texture analysis, image segmentation, shape analysis, structural texture analysis, grouping, system and evaluation, object recognition, and shape from texture. This reader friendly book did a superb job of blending the mathematical principles, techniques, and illustrative examples.

[Vyborny, 1993]
Vyborny, Carl J. "Clinical Considerations in the Analysis of Mammographic Images by Computer," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 42-43. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

An easy to read, introductory level paper that discusses (1) why masses and calcifications are radiographically difficult to detect, and (2) the principal reason why mammograms are misinterpreted and how it can be reduced from occurring.

[Wolfram, 1991]
Wolfram, Stephen. *Mathematica, 2nd ed.* Redwood City CA : Addison-Wesley Publishing Company, Inc., 1991.

Written by the main creator of Mathematica, this extensive primer text, containing 961 pages, provides an introduction to all of the capabilities of this numerical, symbolic, graphical, and high-level programming language package. The text includes an overview, sample sessions, color graphic examples, a tutorial presentation of the basic features, detailed examination of the principles and structure, discussion of advanced features, and eight appendices.

[Yaffe, 1993a]
Yaffe, Martin J. "Image Acquisition and Display Considerations for Digital Mammography," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 44-45. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

A well written, introductory level paper on (1) film-screen mammography and its limitations, and (2) digital mammography, specifically the challenges in developing a digital system, area detectors, slot-beam acquisition, and image display. The discussion on the operation of a diagrammed slot-beam digital mammography system is informative and surprisingly easy to understand.

[Yaffe, 1993b]
Yaffe, Martin J. "Digital Mammography," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

The overview provided in [Yaffe, 1993a] is expanded and discussed in greater detail in this paper. Once again, the focus of the paper is on (1) film-screen mammography and its limitations, and (2) digital mammography, specifically the various detectors currently being used or under development, problems in addition to detector design that need to be resolved, required spatial resolution and contrast sensitivity performance specifications, and potential uses of this technology.

Bibliography

[Astley and others, 1993]
Astley, Sue, Ian Hutt, Stephen Adamson, Peter Miller, Peter Rose, Caroline Boggis, Chris Taylor, Tim Valentine, Jack Davies, and Janette Armstrong. "Automation in Mammography: Computer Vision and Human Perception," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 716-730. San Jose CA : The International Society for Optical Engineering, February 1993.

[Bankman and others, 1993]
Bankman, Isaac N., William A. Christens-Barry, Dong W. Kim, Irving N. Weinberg, Olga B. Gatewood, and William R. Brody. "Automated Recognition of Microcalcification Clusters in Mammograms," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 731-738. San Jose CA : The International Society for Optical Engineering, February 1993.

[Cooper and Fisher, 1979]
Cooper, John D. and Matthew J. Fisher. *Software Quality Management.* New York NY : Petrocelli Books, Inc., 1979.

[Doi and others, 1990]
Doi, Kunio, Heber MacMahon, Shigehiko Katsuragawa, Heang-Ping Chan, Maryellen L. Giger, Kenneth R. Hoffmann, Nobuyuki Nakamori, Charles E. Metz, Hiroshi Fujita, Laura E. Fencil, and Carl J. Vyborny. "Utilization of Digital Image Data for Computer-Aided Diagnosis," *Proceedings of the First International Conference on Image Management and Communication in Patient Care*, 128-129. Washington D.C. : IEEE Computer Society, June 1989.

[Doi, 1991a]
Doi, Kunio. "Technology Requirements for Computer-Aided Diagnosis," *Proceedings, Technology Requirements for Biomedical Imaging - Strategic Defense Initiative Technology Applications Symposium*, 127-130. Los Alamitos CA : IEEE Computer Society, May 1991.

[Doi, 1991b]
Doi, Kunio. "Computer-Aided Diagnosis (CAD) Schemes Based on Quantitative Analysis of Digital Radiographic Images," *ICRU News*, 4-9 (June 1991).

[Doi and others, 1991]
Doi, K., H. MacMahon, S. Katsuragawa, M. L. Giger, K. R. Hoffmann, C. E. Metz, R. M. Nishikawa, H. Yoshimura, S. Sanada, X. Chen, R. A. Schmidt, S. M. Montner, T. Matsumoto, K. G. Chua, and C. J. Vyborny. "Computer-Aided Diagnosis: Present and Future," *Proceedings of the Tutorial Session in the World Congress on Medical Physics and Biological Engineering*, 59-66. Kyoto Japan : Elsevier Science Publishers B. V., July 1991.

[Doi and others, 1993a]
Doi, K., M. L. Giger, R. M. Nishikawa, K. R. Hoffmann, H. MacMahon, R. A. Schmidt, and K. G. Chua. "Digital Radiography," *Acta Radiologica*, 34 : 426-439 (1993).

[Doi and others, 1993b]
Doi, K., H. MacMahon, S. Katsuragawa, M. Giger, K. Hoffmann, R. Nishikawa, S. Sanada, R. Schmidt, F. Behlen, and D. Sluis. "Development of Digital Processing Techniques for Computer-Aided Diagnosis in Radiographic Images," *Proceedings of the Third International Conference on Image Management and Communication in Patient Care: Requirements, Implementation, and Assessment of Digital Imaging in Medicine*, 110-115. Berlin Germany : IEEE Computer Society, June 1993.

[Dronkers, 1992]
Dronkers, Daniel J. "Stereotaxic Core Biopsy of Breast Lesions," *Radiology*, 183 : 631-634 (June 1992).

[Evans and Marciniak, 1987]
Evans, Michael W. and John Marciniak. *Software Quality Assurance and Management.* New York NY : John Wiley & Sons, Inc., 1987.

[Gale and others, 1993]
Gale, A. G., A. R. M. Wilson, and E. J. Roebuck. "Mammographic Screening: Radiological Performance as a Precursor to Image Processing," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 458-464. San Jose CA : The International Society for Optical Engineering, February 1993.

[Giger, 1993a]
Giger, Maryellen L. "Computer-Aided Diagnosis in Digital Mammography," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 46-47. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

[Giger, 1993b]
Giger, Maryellen L. "Computer-Aided Diagnosis," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

[Giger and others, 1991]
Giger, Maryellen L., Robert M. Nishikawa, Kunio Doi, Fang-Fang Yin, Carl J. Vyborny, Robert A. Schmidt, Charles E. Metz, Yuzheng Wu, Heber MacMahon, and Hitoshi Yoshimura. "Development of a 'Smart' Workstation for Use in Mammography," *Proceedings of SPIE, Volume 1445, Medical Imaging V: Image Processing,* 101-103. San Jose CA : The International Society for Optical Engineering, February 1991.

[Giger and others, 1993a]
Giger, M. L., K. Doi, F. F. Yin, H. MacMahon, C. E. Metz, R. A. Schmidt, S. M. Montner, H. Yoshimura, and C. J. Vyborny. "Computer-Vision Schemes for Lung and Breast Cancer Detection," *Proceedings of the 2nd International Conference on Visual Search*, 225-230. Taylor and Francis Ltd, 1993.

[Giger and others, 1993b]
Giger, M. L., R. Nishikawa, R. Schmidt, C. Vyborny, P. Lu, Y. Jiang, Z. Huo, J. Papaioannou, Y. Wu, S. Cox, R. Kunst, U. Bick, K. Rosculet. "Preliminary Evaluation of an 'Intelligent' Mammography Workstation," *Proceedings of SPIE, Volume 1898, Medical Imaging 1993 : Image Processing*, 764-765. Newport Beach CA : The International Society for Optical Engineering, February 1993.

[Giger and others, 1993c]
Giger, Maryellen L., Kunio Doi, Heber MacMahon, Robert M. Nishikawa, Kenneth R. Hoffmann, Carl J. Vyborny, Robert A. Schmidt, Hong Jia, Katsumi Abe, Xuan Chen, Akiko Kano, Shigehiko Katsuragawa, Fang-Fang Yin, Noam Alperin, Charles E. Metz, Frederick M. Behlen, and Douglas Sluis. "An 'Intelligent' Workstation for Computer-Aided Diagnosis," *RadioGraphics,* 13 (3) : 647-656 (May 1993).

[Giger and Vyborny, 1993]
Giger, Maryellen L. and Carl J. Vyborny. "Computers Aid Diagnosis of Breast Abnormalities," *Diagnostic Imaging*, 98-102 (June 1993).

[Giles and others, 1993]
Giles, A., A. R. Cowen, and G. J. S. Parkin. "A Clinical Workstation for Digital Mammography," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 806-817. San Jose CA : The International Society for Optical Engineering, February 1993.

[Gillies, 1992]
Gillies, Alan C. *Software Quality - Theory and Management.* London UK : Chapman & Hall Computing, 1992.

[Hendrick and Parker, 1993]
Hendrick, R. Edward and Steve H. Parker. "Stereotaxic Imaging," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

[Kaplan and others, 1993]
Kaplan, Gadi, Kenneth R. Foster, Ken Kornbluh, and Michael J. Riezenman. "Focus Report: Engineering Software," *IEEE Spectrum*, 30 (11) : 40-87 (November 1993).

[Krupinski and Roehrig, 1994]
Krupinski, Elizabeth and Hans Roehrig. "Comparison of Digital X-ray Cameras for Stereotactic Breast Needle Biopsy: An Observer Performance Study," *Proceedings of the Second International Workshop on Digital Mammography.* York England, July 1994.

[Lippmann and others, 1993]
Lippmann, Richard P., Linda Kukolich, and Elliot Singer. "LNKnet: Neural Network, Machine-Learning, and Statistical Software for Pattern Classification," *The Lincoln Laboratory Journal*, 6 (2) : 249-268 (1993).

[Paredes, 1993]
Paredes, Ellen Shaw de. "Radiographic Breast Anatomy: Radiologic Signs of Breast Cancer," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

[Parker and others, 1990]
Parker, Steve H., Jeffrey D. Lovin, William E. Jobe, Brian J. Burke, Kenneth D. Hopper, James M. Luethke, and Wayne F. Yakes. "Stereotactic Breast Biopsy with a Biopsy Gun," *Radiology*, 176 : 741-747 (September 1990).

[Parker and others, 1991]
Parker, Steve H., Jeffrey D. Lovin, William E. Jobe, Brian J. Burke, Kenneth D. Hopper, and Wayne F. Yakes. "Nonpalpable Breast Lesions: Stereotactic Automated Large-Core Biopsies," *Radiology*, 180 : 403-407 (August 1991).

[Pressman, 1987]
Pressman, Rogers S. *Software Engineering - A Practitioner's Approach, 2nd ed.* New York NY : McGraw-Hill Book Company, Inc., 1987.

[Richter and Claridge, 1993]
Richter, J. and E. Claridge. "Analysis of the Requirements for Quantitative Evaluation of Mammograms by Computers," *Proceedings of the International Symposium CAR' 93 Computer Assisted Radiology*, 618-623. 1993.

[Rogers and others, 1994]
Rogers, Steven K., Dennis W. Ruck, and Matthew Kabrisky. "Artificial Neural Networks for Early Detection and Diagnosis of Cancer," *Cancer Letters*, 77 : 79-83 (1994).

[Romano, 1993]
Romano, Carol A. "Imaging: An Innovative Technology," *Computers in Nursing*, 11 (5) : 222-225 (September/October 1993).

[Shtern, 1992]
Shtern, Faina. "Digital Mammography and Related Technologies: A Perspective from the National Cancer Institute," *Radiology*, 183 : 629-630 (June 1992).

[Shtern and others, 1993]
Shtern, Faina, Sue Astley, David R. Dance, Nico Karssemeijer, Carolyn Kimme-Smith, Robert Nishikawa, Raj Rangayyan, Raman Paranjape, Liang Shen, and J. E. Leo Desautels. "Panel Discussion - Design of a Common Database for Research in Mammogram Image Analysis," *Proceedings of SPIE, Volume 1905, Biomedical Image Processing and Biomedical Visualization*, 535-551. San Jose CA : The International Society for Optical Engineering, February 1993.

[Tanne, 1993]
Tanne, Janice H. "Everything You Need to Know About Breast Cancer . . .," *New York*, 26 (40) : 52-62 (October 1993).

[The MathWorks, Inc., 1992a]
The MathWorks, Inc. *MATLAB User's Guide*. Natick MA : The MathWorks, Inc., 1992.

[The MathWorks, Inc., 1993a]
The MathWorks, Inc. *MATLAB External Interface Guide*. Natick MA : The MathWorks, Inc., 1993.

[The MathWorks, Inc., 1993b]
The MathWorks, Inc. *MATLAB Building a Graphical User Interface*. Natick MA : The MathWorks, Inc., 1993.

[Thomas, 1989]
Taber's Cyclopedic Medical Dictionary, 17th ed, ed. Clayton L. Thomas. Philadelphia PA : F. A. Davis Company, 1989.

[Thompson and Shure, 1993]
Thompson, Clay and Loren Shure. *Image Processing Toolbox*. Natick MA : The MathWorks, Inc., 1993.

[Vyborny, 1993]
Vyborny, Carl J. "Clinical Considerations in the Analysis of Mammographic Images by Computer," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 42-43. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

[W. B. Saunders Dictionary Staff, 1994]
W. B. Saunders Dictionary Staff. *Dorland's Illustrated Medical Dictionary, 28th ed.* Philadelphia PA : W. B. Saunders Company, 1994.

[Williams & Wilkins, 1990]
Stedman's Medical Dictionary, 25th ed, ed. William R. Hensyl. Baltimore MD : Williams & Wilkins, 1990.

[Wolfram, 1991]
Wolfram, Stephen. *Mathematica, 2nd ed.* Redwood City CA : Addison-Wesley Publishing Company, Inc., 1991.

[Yaffe, 1993a]
Yaffe, Martin J. "Image Acquisition and Display Considerations for Digital Mammography," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Volume 15, Number 1, 44-45. San Diego CA : IEEE Engineering in Medicine and Biology Society, October 1993.

[Yaffe, 1993b]
Yaffe, Martin J. "Digital Mammography," *Syllabus: A Categorical Course in Physics - Technical Aspects of Breast Imaging (Second Edition).* Oak Brook IL : Radiological Society of North America, 1993.

Vita

Captain John L. Kelley was born on 1 March 1961 in Riverside, California. He graduated from Mesa Verde High School in Citrus Heights, California in 1979 in the top one percent of his class, receiving a number of prestigious awards including the California Scholarship Federation highest award -- *Life Membership Scholarship Certificate*. After completing all course requirements for admission to medical school, he enlisted in the United States Air Force in 1983. In basic military training at Lackland AFB, Texas, he was awarded the Honor Graduate Ribbon for obtaining the highest composite score in his flight. At the School of Health Care Sciences, Medical Service Specialist Course, Sheppard AFB, Texas, he was the only honor student and the only student leader in his class. His assignments include: Medical/Surgical Technician and Obstetrics/Gynecology Technician, USAF Hospital Beale, Beale AFB, California; Emergency Room Technician, USAF Hospital Osan, Osan AFB, Korea; and Primary Care/Emergency Room Technician, USAF Clinic McClellan, McClellan AFB, California. During these tours, he became a Licensed Vocational Nurse - State of California, and earned state and national certification as an Emergency Medical Technician IA, and certification as an American Heart Association Basic Cardiac Life Support Instructor and Advanced Cardiac Life Support Provider. In 1987, he was accepted into the Airman Education and Commissioning Program and attended Arizona State University, Tempe, Arizona. Upon graduating with honors -- including *Eta Kappa Nu* and *Tau Beta Pi* -- and a Bachelor of Science in Engineering (Electrical Engineering major) in 1990, he returned to Lackland AFB, Texas to attend Officer Training School. His first assignment as a commissioned officer was as a Technical Engineering Team Chief, and later as the Intercontinental Ballistic Missile (ICBM) Systems Technical Director, for the 394th Maintenance Support Squadron, Vandenberg AFB, California. He directed the analysis, isolation, and resolution of the most complex weapon system failures impacting the USAF's nationally significant ICBM Follow-on Operational Test and Evaluation programs, contributing immensely to the success of nine Minuteman and seven Peacekeeper missile launches. In 1993, he entered the Air Force Institute of Technology at Wright-Patterson AFB, Ohio to pursue a Master of Science in Computer Science/Software Engineering.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE
Development of a
Mammographic Image Processing Environment
Using MATLAB

5. FUNDING NUMBERS

6. AUTHOR(S)
Capt John L. Kelley

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/94D-12

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Capt Jeff W. Hoffmeister
AL/CFHD
2255 H St BLDG 248A
Wright-Patterson AFB, OH 45433-7022

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Breast cancer is a disease that accounts for a disturbingly large number of deaths in females each year. Its prevalence is a topic of concern to all of us since it can affect our families, friends, and coworkers. Although mammographic screening is the most effective method currently available for the early detection of breast cancer, it is far from being an infallible procedure. Mammographic reading is error prone, partly because of the complexity of the task and partly because of the variability in human performance. Computers offer high reproducibility, and when used as an adjunct by the radiologist, may improve diagnostic accuracy and thus the mammographic screening process.

The goal of this research was to create using MATLAB version 4.2 (UNIX) and the MATLAB Image Processing Toolbox a menu-based, mouse-driven, and keyboard interactive *Mammogram Image Processing Program* that would support the future research and development of complex segmentation, feature extraction, and classification algorithms. McCall's *software quality factors* are used on a general level to assess the program's operational characteristics, ability to undergo change, and adaptability to new environments. An important criteria driving this research was to build a platform that would provide full source code accountability so that errors in earlier image processing stages could be fixed and not allowed to propagate through to later stages.

14. SUBJECT TERMS
Mammography, Computer-Aided Diagnosis, Image Processing, MATLAB

15. NUMBER OF PAGES
216

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. ***(If known)***

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

★U.S.GPO:1990-0-273-271