Prostate cancer recognition in ultrasound images

Theodores Dimitrios Lazarakis
University of Nevada, Las Vegas
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
PROSTATE CANCER RECOGNITION IN
ULTRASOUND IMAGES

by

Theodores Lazarakis

Bachelor of Science
University of Nevada, Las Vegas
2000

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2002
The Thesis prepared by

Theodores Lazarakis

Entitled

Prostate Cancer Recognition in Ultrasound Images

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Prostate Cancer Recognition in Ultrasound Images

by

Theodores Lazarakis

Dr. Evangelos A. Yfantis, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

Prostate cancer is the number one incident cancer and number two cause of cancer deaths among American men. It is estimated that 189,000 men will be diagnosed with prostate cancer and 30,500 will die from it in 2002.

Early detection increases the survival rate of patients and ultrasonography is one of the preferred methods used for diagnosis of prostate cancer due to its non-invasiveness and relative inexpensive operation. In ultrasound the produced signal is directed via a probe onto the prostate. Part of the sound signal is absorbed by tissue, part of it is reflected back onto the receiver and part of it scatters from one or more points before it is reflected back to the receiver. The received signal intensity is quantized usually on a gray scale between 0 and 255 and is therefore transformed from the ultrasound space to the image space.

Our purpose is to aid medical doctors in prostate cancer detection via computer automated analysis of prostatic ultrasound imagery. Survival of cancerous tissue is based upon the availability of blood supply to the cancerous tissue. Cancerous tissue develops its own blood supply system which is different than the blood supply system of normal
tissue. Due to this fact, absorption of ultrasound signals is different in cancerous areas than in non-cancerous areas. The energy of the signal, the continuity of the signal, the autocorrelation function and frequency domain properties of prostatic ultrasound images are different in normal tissue than in cancerous tissue.

This thesis presents an algorithm for automated cancer recognition in prostatic ultrasound imagery. Statistical and morphological based models are employed to classify regions of ultrasound imagery as either cancerous or non-cancerous. Application of our algorithm onto a limited set of cancerous and non-cancerous ultrasound images shows that our method has the ability to recognize cancer in cancerous ultrasound images. Misclassification occurs when cancerous tissue is classified as non-cancerous and non-cancerous tissue is classified as cancerous. Occurrences of misclassification have been observed and investigated.
TABLE OF CONTENTS

ABSTRACT ................................................................................................................................... iii

TABLE OF CONTENTS ............................................................................................................. v

LIST OF FIGURES ................................................................................................................ vi

ACKNOWLEDGMENTS ....................................................................................................... viii

CHAPTER 1 INTRODUCTION ............................................................................................... 1
  1.1 Overview and History of Prostate Cancer Disease .................................................. 1
  1.2 Ultrasonography and Image Acquisition Methods .................................................. 2
  1.3 Prostate Cancer Diagnosis and Treatment ............................................................ 4
  1.4 Problem Definition, Constraints and Assumptions .................................................. 8

CHAPTER 2 RELATED RESEARCH .................................................................................. 12
  2.1 Overview of Tissue Classification Methods ........................................................... 12
  2.2 Statistical and Texture Analysis ................................................................................ 14
  2.3 Frequency Analysis ..................................................................................................... 15
  2.4 Morphological Analysis ............................................................................................. 18

CHAPTER 3 THEORETICAL ANALYSIS ......................................................................... 26
  3.1 Model Overview ......................................................................................................... 26
  3.2 Statistical Properties and Indicators ......................................................................... 32
  3.3 Algorithm Training via Data Selection and Tissue Classification via
      Mahalanobis Distance Measurements .............................................................................. 40
  3.4 Final Pixel Classification into Cancerous or Non-Cancerous Sets via
      Morphological Tissue Analysis ..................................................................................... 46

CHAPTER 4 PRACTICUM .................................................................................................... 53
  4.1 Algorithm Overview ................................................................................................... 53
  4.2 Automatic Region of Interest Detection .................................................................. 54
  4.3 Image Analysis and Classification ............................................................................ 65

CHAPTER 5 CONCLUSION ................................................................................................. 67
  5.1 Summary and Results ................................................................................................. 67
  5.2 Further Research ......................................................................................................... 70

APPENDIX 1 COMPUTER PROGRAM LISTINGS .......................................................... 75

BIBLIOGRAPHY ..................................................................................................................... 123
LIST OF FIGURES

Figure 1: Incidence and Mortality rates between 1980-2002 ............................................. 1
Figure 2: Ultrasound image acquisition .................................................................................3
Figure 3: Gleason grades for prostatic tumors ...................................................................... 6
Figure 4: Noisy ultrasound image of the prostate................................................................. 9
Figure 5: Probability density function of a dark 21x21 pixel segment of the above noisy ultrasound image ................................................................. 9
Figure 6: Noise-free ultrasound image of the prostate.......................................................... 10
Figure 7: Probability density function of a dark 21x21 pixel segment of the above noise-free ultrasound image ............................................................... 10
Figure 8: B-mode ultrasound image of the prostate............................................................. 11
Figure 9: Cancerous region detected by an RF spectrum analysis algorithm .................... 13
Figure 10: Cancer detected at the granular periphery using again an RF spectrum approach ................................................................................................. 13
Figure 11: Cancerous regions detected using a statistical based approach ......................... 14
Figure 12: 3-D prostate reconstruction based on a histological approach ....................... 16
Figure 13: Average linear regression lines of spectra ......................................................... 18
Figure 14: Segmented masses in three mammographic images ......................................... 19
Figure 15: The eight possible pixel configurations in 8-connectivity ................................. 20
Figure 16: The chain code in 8-connectivity for the above partial border is 2,1,0,7,7,0,1,1 ..................................................................................................................... 20
Figure 17: An object (in black) and its corresponding convex hull (in red) ....................... 21
Figure 18: Convexity and solidity of irregular and regular objects ....................................... 22
Figure 19: The border of an object is shown in (a) with its corresponding signature function shown in (b) ......................................................................................... 25
Figure 20: A magnified portion of an ultrasound image with carcinoma ......................... 26
Figure 21: An ultrasound image of the prostate with hypoechoic, isoechoic and hyperechoic areas indicated ......................................................................................... 27
Figure 22: The pixel intensity histogram for a normal tissue segment ............................. 28
Figure 23: The pixel intensity histogram for a cancerous tissue segment ........................ 29
Figure 24: The suspicious cancerous location detected by the algorithm based only on statistical and textural analysis of the tissue segments .................................................... 30
Figure 25: The suspicious cancerous locations detected by the algorithm after the Morphological tissue analysis ...................................................................................... 33
Figure 26: Ultrasound image of the prostate. Biopsy confirmed carcinoma with Gleason score 6 as indicated on the image ................................................................. 41
Figure 27: Selection of cancerous ROI for algorithm training ............................................. 42
Figure 28: Non-cancerous ultrasound image with multiple ROI’s selected for Algorithm training ................................................................................................................. 42
Figure 29: Magnified portion of a cancerous region ............................................................ 47
ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Dr. Evangelos Yfantis, for his guidance and mentorship during this course of study. He was always there to help me even with his busy schedule and for that I will always be indebted to him. I would also like to thank Dr. Cochran and the cancer center at M. D. Anderson Cancer Center for their donation of the ultrasound images.

I am very grateful to Dr. Hal Berghel, Dr. John Harrison and Dr. Georg Mauer for taking time out of their busy schedule to be members of my committee.

I thank my very dear friends, Oksana Kubushyna and Alex Popovich, for their help and suggestions.

Lastly, I would like to thank my parents for their infallible love and support. They made all this possible.
CHAPTER 1

INTRODUCTION

1.1 Overview and History of Prostate Cancer Disease

Prostate cancer is the number one incident cancer and number two cause of cancer deaths among American men. The disease rarely affects men before the age of 40 and cancer incidence cases reach their peak in men between the ages of 60 and 79 [24], [25]. It is estimated that 189,000 men will be diagnosed with prostate cancer and 30,500 will die from it in 2002 [26]. The number of prostate cancer cases increased dramatically in the late 1980s and it reached a peak in 1992 (see figure 1).

![Incidence and Mortality Rates](image)

Figure 1. Incidence and Mortality rates between 1980-2002.
Doctors have attributed this phenomenon primarily due to more successful diagnostic methods and also due to an increase in cancer screenings. Indeed, in the late 1980s the prostate specific antigen (PSA) blood test was developed and used in prostatic cancer screenings. Since 1992, the number of cancer cases has decreased somewhat but it has still remained high with an average of 180,000 men diagnosed each year and with 40,000 dying yearly [25]. Studies show that early detection increases the survival and recovery rates of patients and different methods have been developed for the detection of the disease.

1.2 Ultrasonography and Image Acquisition Methods

In ultrasonography high frequency sound signals are employed towards the acquisition of tissue information. Ultrasound images are generated using a pulse-echo principle where high frequency sound waves in the range of 3 to 15 MHz are transmitted into the body via a transducer. As the ultrasound pulse traverses into the body, some of the ultrasound energy is reflected back toward the transducer. Different parts of the body absorb and reflect ultrasound waves differently; hence, the intensity of the echoed ultrasound signal back to the transducer is different depending upon the matter encountered by the ultrasound signal [15], [4]. The received signal intensity is quantized usually on a gray scale between 0 and 255. The time it took for an ultrasound signal to reflect back to the transducer and the quantized intensity of the reflected ultrasound signal generate an image. It is this image that we operate on. Unfortunately, due to multiple reflections of the sound signal (scattering of the signal), images generated using
pulse-echo ultrasound imaging incorporate significant noise with the signal [24], [9], [27], [24], [4].

Due to the location of the prostate gland image acquisition of the prostate via ultrasound is ideal [24]. Ultrasound images of the prostate gland are obtained with the aid of a transrectal ultrasound probe. This probe is shaped like a cylindrical rod with the actual transducer being mounted on the front end of the probe. Access to the prostate for ultrasound image acquisition is obtained via insertion of the probe into the patient's rectum. The transducer is then oriented to obtain a polar-type circular scan about the axis of the probe, generating a 2-D image slice of the prostate gland (see figure 2). The acquisition of multiple 2-D images from different angles also allows for 3-D image reconstruction using stereo image algorithms [10], [11]. Unlike invasive methods such as X-rays, CT machines and others, ultrasound is noninvasive. This

Figure 2. Ultrasound image acquisition.
property allows for real time visual inspection and multiple image acquisition of tissue and organs without having the need to take any special precautionary measures [9], [15]. Additionally, the logistics of storage and installation of ultrasound machines are easier than CT and MR machines since ultrasound machines are smaller and lighter. Furthermore, the costs associated with an ultrasound machine are orders of magnitude less than MR or CT machines; this relative inexpensiveness allows ultrasound machines to be affordable by all medical practitioners [7], [24].

1.3 Prostate Cancer Diagnosis and Treatment

Prior to the introduction of the PSA blood test for prostate cancer diagnosis in the late 1980s, doctors administered the digital rectal examination (DRE) procedure. This procedure entailed the medical doctor to place one of his fingers into the patients’ rectum for as to feel the prostate gland for any abnormalities. If abnormalities were detected during the procedure, the patient would undergo further testing via biopsy. Although the procedure was easy and quick to perform, doctors were unable to feel the entire prostate gland through the wall of the rectum. Hence, during initial screenings with DRE there were often many false negatives diagnoses, meaning that individuals were diagnosed as having no cancer when in fact they did. This lead positive diagnoses to occur during later stages of the disease when the cancer had already metastasized beyond the prostate. As a result, six months after their initial examination, half of the false negative patients were diagnosed with prostate cancer that had already spread beyond the prostate gland, making it essentially incurable [8]. Since early detection increased the survival rate of patients, better diagnostic methods were developed to detect prostate cancer when it was still
confined to the prostate gland. PSA is one such method that strives to detect cancer in early stages prior to any metastasis. Prostate specific antigen is a protein that is made in the prostate and prostate cancer cells. Elevated PSA concentration in the blood during a blood test could indicate cancer in the prostate. An elevated PSA or an abnormal DRE may prompt physicians to suggest having a biopsy procedure performed to further determine if the patient has the disease.

During the biopsy procedure, a transrectal ultrasound probe (see figure 2) is used to guide the placement of a biopsy needle. The doctor uses the biopsy needle to obtain tissue samples from several different areas of the prostate. The standard procedure currently calls to obtain six tissue samples from uniformly distributed locations within the prostate. The pathologist then examines the tissue cells under a microscope to determine the presence of cancer and its aggressiveness. Unfortunately, results indicate that approximately one in five cancers is missed. These results are attributed due to the way tissue samples are taken during biopsy procedures. Conventional transrectal ultrasound scan (TRUS) images cannot adequately distinguish suspicious prostatic tissue from normal tissue; therefore, the images cannot be used effectively to guide biopsy needles [8]. Better biopsy strategies are needed so as to reduce the number of missed cancer incidences [28], [29]. Once cancer has been detected, the Gleason grading system is used to indicate the cancer’s potential to grow and spread to other areas of the body. Each tissue region is assigned a Gleason grade from one to five. The final Gleason score is a number between two and ten and is formed based on the sum of the two most predominant tissue grades (see figure 3 for an example of Gleason grades). In general, a Gleason score of seven or more is considered high and it indicates a prostate tumor that is
likely to spread to other areas of the body, while a low Gleason score indicates a less aggressive tumor that is not likely to spread outside the prostate. If cancer is present, doctors will try to determine the extent of the tumor in the prostate and also to determine if cancer cells have spread to surrounding tissue. This examination is called “staging” and is usually accomplished through the use of ultrasound imaging techniques.

![Gleason grades for prostatic tumors.](image)

Figure 3. Gleason grades for prostatic tumors.

Once a patient is diagnosed, based on the stage of the disease different treatment options are available and they vary widely among doctors and treatment centers. Radical prostatectomy can be performed on patients with an early stage of the disease when it is still confined within the prostate gland. The procedure removes the cancer by surgically removing the prostate gland. Although the cancer is removed, radical prostatectomy has significant side affects including temporary difficulty of controlling urinary flow and at least a 70% patient impotency [25].

External beam radiotherapy is used when the prostate cancer is confined within the prostate gland or the surrounding tissue. The percentage of patients cured is proportional to the dose of radiation given to the prostate gland. Higher radiation doses result in a higher risk of side effects, which include bleeding from the rectum due to the development of small blood vessels called telangiectasia and impotence that has a slightly
less chance of occurrence than with radical prostatectomy. In order to reduce the side
effects of external beam radiotherapy and to deliver a high dose of radiation to the cancer
while giving a minimal dose of radiation to the surrounding tissue, the prostate seed
implantation technique was developed.

Patients having cancer that is confined to the prostate gland and that is not very
aggressive can undergo prostate seed implantation. A Gleason score of six or less and a
PSA score of less than ten allows for the procedure. During the prostate seed
implantation procedure the prostate is evaluated using ultrasound imaging. A 3-D map of
the prostate gland is constructed via a computer and the acquired prostatic ultrasound
images. The computer is then used to make a map for the placement of the radioactive
seeds such that a high dose of radiation is delivered to the cancer while minimizing the
radiation dosage given to the surrounding tissue. On the average, one hundred
radioactive seeds are place in the prostate gland via the use of needles that are entered
through the skin between the scrotum and the anus. The needles are guided to the
prostate via the use of ultrasound imaging. As the needles are removed from the prostate
gland the radioactive seeds are left behind and are removed only after the treatment has
been completed. About 10% of the patients develop swelling in the prostate gland and a
catheter is needed for about five to ten days after the procedure [25]. Patients having
locally advanced or metastasized cancer usually undergo hormonal modulation and
chemotherapy.
1.4 Problem Definition, Constraints and Assumptions

As stated above, early detection and treatment are essential for the survival and recovery of the patient. Ultrasound imaging plays an important role in the diagnosis and treatment of prostate cancer [8], [27], [15]. In this thesis we present an algorithm for cancer recognition in prostatic ultrasound imaging for aiding medical doctors in recognizing prostate carcinoma. Information obtained using our algorithm would aid medical doctors in staging procedures and prostate seed implantations during the diagnosis and treatment stages of the disease, respectively. A more accurate assessment of the disease and its location will allow doctors to obtain better tissue samples during biopsy procedures and also allow for a more effective placement of radioactive seeds during treatment of the disease.

The algorithm described in this thesis is based on statistical properties of ultrasound images and may fail when noise is incorporated into these images. Noise in ultrasound images has statistical properties which are different than those in normal ultrasound images (see figures 4, 5, 6 and 7). Therefore, the assumption that the ultrasound images analyzed by the algorithm are free from any noise is made. Furthermore, since the algorithm could be used in the staging procedure of the prostate cancer, analysis of the ultrasound image is not limited only to the prostate gland but also to the surrounding tissue (see figure 8). This enables the detection of cancer to the surrounding tissue if spreading has occurred beyond the prostate gland. Lastly, due to several modalities for ultrasound image acquisition of the prostate, our algorithm operates only on B-mode ultrasound images (see figure 8).
Figure 4. Noisy ultrasound image of the prostate. Histogram equalization was performed on a 60x60 pixel region (middle top) and overlapped on top of the original image to show the effects of noise on the image signal.

Figure 5. Probability density function of a dark 21x21 pixel segment of the above noisy ultrasound image.
Figure 6. Noise-free ultrasound image of the prostate. Histogram equalization was performed on a 60x60 pixel region (middle top) and overlapped on top of the original image.

Figure 7. Probability density function of a dark 21x21 pixel segment of the above noise-free ultrasound image. Note that the highest frequency count is at zero which corresponds to no noise in the image.
Figure 8. B-mode ultrasound image of the prostate. The prostate gland is outlined in green. The red outline shows the region where tissue is to be analyzed by our algorithm.
CHAPTER 2

RELATED RESEARCH

2.1 Overview of Tissue Classification Methods

A number of algorithms for prostate cancer recognition have been developed but with limited success [16]. The inability for attaining good results has been attributed to the low resolution of ultrasound images and due to the high speckle noise that is sometimes introduced into the signal during image acquisition [24], [9], [27], [4]. Such tissue recognition and classification algorithms can be found in literature in [1]-[8], [12]-[17], [27]. Since tissue recognition algorithms with similar recognition problems are not limited only to tissue recognition of the prostate, other recognition and classification algorithms for different tissue and organs, such as tumor detection in digital mammograms, are also discussed.

One of the foremost problems encountered by algorithms in automated prostate cancer recognition is the detection of suspicious areas over large regions of the prostate in ultrasound images (see figures 9, 10, 11) [30], [8], [17]. The recognition of suspicious areas over large regions of the prostate hinders the ability to obtain good tissue samples during biopsy procedures and even mislead the implantation of radioactive seeds during prostate seed implantation treatments [8]. The algorithm developed in this thesis strives to suppress the detection of large suspicious areas by the aid of more sophisticated analysis methods, which are discussed in chapter 3. Some of the tissue classification
methods used in the literature include statistical analysis, texture analysis, frequency analysis, data mining and morphological analysis.

Figure 9. Cancerous region detected by an RF spectrum analysis algorithm. Note that the algorithm has detected approximately half of the prostate as having cancer.

Figure 10. Cancer detected at the glandular periphery using again an RF spectrum approach. Note the relatively large cancerous area detected by the algorithm.
Figure 11. Cancerous regions detected using a statistical based approach. Note again the relatively large suspicious areas detected.

2.2 Statistical and Texture Analysis

Once healthy prostate tissue becomes cancerous its structure changes and the ultrasound signal is absorbed differently between cancerous and non-cancerous areas [9], [15], [17]. The theoretical approach behind the statistical and texture analysis algorithms is finding indicators that differ between cancerous and non-cancerous tissues in ultrasound images. Algorithms developed in [15] and [4] use indicators such as the minimum, maximum, median, mean and standard deviation of cross-sectional regions in ultrasound images to determine if a region is cancerous or non-cancerous. In [17] a texture analysis algorithm is used considering each pixel as being the center of a 9x9 pixel window. The angular second moment, entropy and the inverse difference moment are all chosen as indicators and used for statistical analysis of the tissue. Other texture analysis methods include the Fourier power spectrum and autocorrelation functions [17]. Texture analysis is performed for each 9x9 pixel window and the results are associated with the center pixel; hence, if the results indicate that the 9x9 pixel window exhibits cancerous characteristics then the associated center pixel is painted to indicate cancer. These algorithms commit a large error by detecting large areas of tissue as having cancer.
(see figures 9, 10 and 11). Additionally, texture analysis does not often yield good
detection results and further tissue analysis is needed [13], [16], [17]. To achieve better
results many algorithms use a combination of methods such as statistical and textural
analysis, texture and morphological tissue analysis, frequency and texture analysis [12]-
[14], [19], [31], [16].

Other statistical approaches try to construct 3-D probability maps of prostate
cancer based on the history of the disease. This histological analysis approach can be
found in [28], [29], [32], [35] and [36], where probability maps of the occurrence of
prostate cancer are constructed by using locations of the cancer determined during
biopsies of prostate glands that were removed during radical prostatectomies. Additional
information such as race, age, PSA levels and abnormal DRE are also used in the
construction of the model [32]. This method constructs a genetic probability model of
the prostate so that for any patient, a 3-D model of the most probable cancer locations is
build using the genetic model and the patients’ information. One such genetic probability
model can been seen in figure 12. Instead of using standard protocols and sampling
tissue from uniformly distributed locations within the prostate, the above approach can be
used as a more effective way of sampling tissue during needle biopsy procedures.

2.3 Frequency Analysis

Different approaches have been taken for detecting suspicious tissue regions of
the prostate during biopsy needle procedures. Due to the limitations of TRUS in
detecting suspicious regions, some researches have conducted studies to investigate the
frequency analysis of radio frequency echo signals before they are converted into
ultrasound images [8], [33]. This method strives to perform unknown tissue
colorization by comparing the normalized power spectra parameters of the unknown
tissue with parameters from a histologically constructed database. Construction of such a

database involves the following procedures. Each ultrasound image is segmented into
regions having dimensions of 3 x 3.5 mm. Prior to any needle biopsy, the radio
frequency echo signals ranging from 4.5 to 7.0 MHz from each biopsy location are
acquired and are multiplied by a Hamming window. Let $N$ be the number of samples
acquired then the Hamming window $w(k)$ is defined as

$$w(k) = 0.54 - 0.46 \cos\left(\frac{2\pi k}{N-1}\right), \quad k = 0, 1, \ldots, N-1.$$
The squared magnitude of the power spectrum is then computed from the above samples using the DFT (see section 2.4 on how to obtain the power spectrum of a signal). The average power spectrum from all regions of interest (ROI) is then computed. The spectrum is then converted to decibels using the following equation

\[ N_{db} = 20 \cdot \log_{10} N. \]

In order to normalize the power spectrum, radio frequency data for calibration were acquired from the surface of an optically flat glass-plate in a water bath with the transducer oriented to be normal to the scan plane. With spectral amplitudes expressed in decibels, normalization is then performed by subtracting the power spectrum of the planar calibration target from the average power spectrum of the RF data in the ROI. This normalization corrects any differences in gain settings of the ultrasound machine. The resulting normalized spectra appear to be different between cancerous and non-cancerous tissues. A line is fitted through the normalized spectra via linear regression. The slope and intercept of a line are computed so that the total distance between every point and the line is minimum. Since the normalized spectra for cancerous and non-cancerous tissues appear different, their corresponding slopes, intercepts and midband values of the lines generated via linear regression are also different (as can be seen in figure 13). Once the tissue sample has was obtained and clinical examinations by medical doctors had determined the tissue to be cancerous or non-cancerous, the tissue parameters are stored accordingly in the database. Therefore, only parameters of histologically proven tissue types are stored into the database. The parameters employed are the slope, intercept and midband values of the linear regression lines. Furthermore, classification of an unknown tissue segment as cancerous or non-cancerous is based on
how well the slope, intercept and the midband values of the line generated via linear regression of the corresponding power spectrum resembles the slope, intercept and midband values already in the database for cancerous and non-cancerous tissues, respectively.

![Graph](image)

**Figure 13.** Average linear regression lines of spectra attained from a learning set of 29 cancerous ROI's, 23 benign prostatic hyperplasia (BPH) ROI's and 37 ROI from unspecified benign tissue. Note that BPH and benign tissues are non-cancerous tissues.

2.4 Morphological Analysis

Morphological tissue analysis has been used in the detection and classification of mammographic lesions in digitized mammograms [19], [34]. Unfortunately, morphological tissue analysis does not appear to be used anywhere in literature for prostatic cancer recognition or analysis of tumors in the prostate gland. Analysis of mammographic masses for tumor classification first entails the segmentation of masses. A number of automated mass segmentation algorithms for mammographic tumor masses have been developed. Segmentation via simple thresholding and the K-Means clustering
algorithm are frequently used due to their ease of implementation. The segmentation via thresholding scheme uses the same principle as the image binarization algorithm. More specifically, for every pixel $P_i$ of the image, a new pixel $P'_i$ is generated such that

$$P'_i = \begin{cases} 0, & \text{if } P_i < T_b \\ 1, & \text{if } P_i \geq T_b \end{cases}$$

where $T_b$ is the threshold value. Let $P'$ be the binarized image having $N$ data points (points that have a value of 1). The K-Means clustering algorithm is used to cluster segments of one or more disjoint objects for as to create $K$ tumor masses in mammographic images. The algorithm clusters the $N$ data points into $K$ disjoint subsets $S_j$ containing $N_j$ data points so as to minimize the following error function, which is based on the sum-of-squares criterion:

$$E = \sum_{j=1}^{K} \sum_{x \in S_j} |x - c_j|^2$$

where $x$ is a vector representing the $n^{th}$ data point in the set $S_j$ and $c_j$ is the centroid of the data points in $S_j$. Images of some tumor segmentations can be seen in figure 14.

![Figure 14. Segmented masses in three mammographic images.](image-url)
Once the tumor masses have been segmented then classification via morphological analysis is performed. A number of algorithms for morphological object analysis have been described in literature and are further discussed below.

The convex hull of an object is the smallest convex contour enclosing that object. An example can be seen in figure 17. The perimeter of an object is the number of pixels in the boundary of that object. If \( x_1, x_2, x_3, \ldots, x_N \) is a boundary list, the perimeter \( P \) is given by:

\[
P = \sum_{i=1}^{N-1} d_i = \sum_{i=1}^{N-1} |x_i - x_{i+1}|
\]

where \( d_i \) is the distance between the two pixels \( x_i \) and \( x_{i+1} \), and it defined based on the 8-connectivity of the two pixels.

![Figure 15. The eight possible pixel configurations in 8-connectivity.](image1)

![Figure 16. The chain code in 8-connectivity for the above partial border is 2,1,0,7,7,0,1,1.](image2)
Boundaries of objects can be described using a sequence of line segments with a given orientation. A boarder is said to be 8-connected if the orientation of the line segments can be any of the eight possible ways of joining two adjacent pixels together as shown in figure 15. In 8-connectivity, the length of each line segment is defined as 1 for vertical and horizontal steps and $\sqrt{2}$ for diagonal steps. A chain code or Freeman’s code is a sequence of numbers that represents the orientation of the line segments of an object’s boundary. An example of an object’s chain code in 8-connectivity can be seen in figure 16.

Convexity is the relative amount that an object differs from its convex object. Convexity is measured by forming the ratio of the perimeter of the object’s convex hull to the perimeter of the object itself. Hence,

$$\text{Convexity} = \frac{\text{convex perimeter of object}}{\text{perimeter of object}}.$$ 

Therefore, objects having shapes that are convex and they closely match the shapes of their convex hulls will have convexity values near one. Objects having non-convex shapes such as objects having irregular boundaries will have convexity values near zero (see figure 18).
Solidity is another morphological descriptor that measures the density of an object. A measure of solidity is obtained as the ratio of the area of the object to the area of the convex hull of that object. Hence,

\[
\text{Solidity} = \frac{\text{area of object}}{\text{area of convex hull of object}}.
\]

Therefore, solidity values near one signify solid objects or objects that have a similar structure as the convex hull of that object. Solidity values closer to zero signify objects that have irregular boundaries (see figure 18).

![Figure 18](image)

Figure 18. The irregular object in (a) has a solidity = 0.60 and convexity = 0.48, while the object in (b) has a solidity = 0.94 and convexity = 0.72.

The \(k\)-slope of the boundary at location \((x_i, y_i)\) can be estimated from the slope of the line joining \((x_{i-k}, y_{i-k})\) and \((x_{i+k}, y_{i+k})\). The angle \(\varphi\) of the line joining \((x_{i-k}, y_{i-k})\) and \((x_{i+k}, y_{i+k})\) can be calculated as follows:

\[
\varphi = \tan^{-1}\left(\frac{y_{i+k} - y_{i-k}}{x_{i+k} - x_{i-k}}\right).
\]

The \(k\)-curvature, \(C_k\), of an object’s boundary at location \((x_i, y_i)\) can be estimated from the change in the \(k\)-slope as follows:
\[ C_k(x_i,y_i) = \left\{ \tan^{-1}\left( \frac{y_{i+k} - y_i}{x_{i+k} - x_i} \right) - \tan^{-1}\left( \frac{y_i - y_{i-k}}{x_i - x_{i-k}} \right) \right\} \mod(2\pi). \]

The total bending energy \( BE \) is used as a robust global shape descriptor for an object’s boundary. Let \( P_j \) be a pixel with coordinates \((x_j, y_j)\), the bending energy of an object’s boundary is defined as follows:

\[
BE = \frac{1}{L} \sum_{j=1}^{L} C_k^2(P_j).
\]

The bending energy of a border may be understood as the energy necessary to bend a rod to the desired shape [20].

Given a 2-D binarized image of an object, the \((p,q)\)-order spatial moment is defined as

\[
m_{pq} = \sum_{r=0}^{M-1} \sum_{v=0}^{N-1} [x^p \cdot y^q \cdot P(x,y)],
\]

where \(M\) is the size of the image width, \(N\) is the size of the image height, \(P(x,y)\) is the pixel value at location \((x,y)\) of the binarized image and \(p,q \geq 0\). By the above definition we see that the zero\(^{th}\)-order moment, denoted as \(m_{00}\), results in the measurement of the object’s area. The centroid or center of gravity of an object is used to specify the location of the center of mass of the object and can be calculated as follows:

\[
\text{centroid} = (\bar{x}, \bar{y}) = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right).
\]

The radial distance is the distance between the centroid of an object and the location of its boundary pixels. Given a boundary having \(N\) pixels, the radial distance is defined as

\[
d(n) = \sqrt{(x(n) - \bar{x})^2 + (y(n) - \bar{y})^2}.
\]
where $0 \leq n \leq N - 1$. Let $d_{\text{max}}$ be the maximal distance of $d(n)$. The normalized radial distance $r(n)$ is defined as $r(n) = \frac{d(n)}{d_{\text{max}}}$. 

The signature of an object's boundary is an alternative 2-D function representation of the boundary. The signature $S(\psi)$ is a function consisting of the normalized radial distance $r(n)$ and by an angle $\psi$. The signature function of a boundary with a total of $N$ boundary points is defined as

$$S(\psi) = r\left(\frac{\psi}{\Delta\psi}\right),$$

where $\Delta\psi = \frac{2\pi}{N}$ and $\psi = \Delta\psi, 2\Delta\psi, ..., (N-1)\Delta\psi$. An example of an object with its corresponding signature function can be seen in figure 19. Therefore, one circumnavigation of the boundary takes time $2\pi$ and a periodic function with period $2\pi$ is generated via multiple passes around the object's boundary. Since the signature function is a periodic function, frequency analysis is performed in order to obtain information about the boundary's shape.

Further analysis of the object's morphology is obtained via frequency analysis of its corresponding boundary signature. Analysis in the spectral domain is accomplished via the discrete Fourier transform (DFT) and the Fourier descriptors. The Fourier descriptors $T(u)$ are calculated from the DFT of a boundary having $N$ boundary points as follows:

$$T(u) = \frac{1}{N} \sum_{n=0}^{N-1} S(n \cdot \Delta\psi) e^{-j2\pi un / N},$$
where \( u = 0, 1, \ldots, N-1 \). Note that \( T(u) \) is a complex number and can be written as

\[
T(u) = a(u) - j \cdot b(u),
\]

where

\[
a(u) = \frac{1}{N} \sum S(n \cdot \Delta \psi) \cos \left( \frac{2 \pi u n}{N} \right)
\]

and

\[
b(u) = \frac{1}{N} \sum S(n \cdot \Delta \psi) \sin \left( \frac{2 \pi u n}{N} \right).
\]

Therefore, the power spectrum \( P(u) \) is then equal to \( P(u) = \sqrt{a^2(u) + b^2(u)} \). The low frequencies in the power spectrum correspond to the smooth behavior of the boundary and the high frequencies in the power spectrum correspond to the jagged edges of the boundary.

![Figure 19. The border of an object is shown in (a) with its corresponding signature function show in (b).](image-url)
CHAPTER 3

THEORETICAL ANALYSIS

3.1 Model Overview

It is known that developed cancer lesions cause changes in the tissue structure of the prostate gland [9], [15], [17]. These changes in tissue structure are depicted as tissue discontinuities in prostatic ultrasound imagery (see figure 20). Furthermore, in order for cancer to survive it develops its own blood supply system, which is different than the supply system of normal tissue. The velocity of the blood flowing through the cancerous

![Figure 20. A magnified portion of an ultrasound image with carcinoma. Histogram equalization has been applied to the region in order to make the discontinuities more visible.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Blood vessels is different than the velocity of the blood flowing through blood vessels of normal tissue. Due to these facts the ultrasound signal is absorbed differently in cancerous areas than in normal tissue areas [4]-[7].

Areas of prostatic tissue in ultrasound images can be classified as hypoechoic, isoechoic or hyperechoic depending on the intensity of the pixels in that area. Hypoechoic are areas with relatively low pixel values, isoechoic are areas with pixel values about equal to the average pixel value of the image and hyperechoic are areas with relatively high pixel values compared to the average pixel values of the image (see figure 21) [9]. Studies have shown that 96% of cancerous lesions are hypoechoic and are most often located in the peripheral zone of the prostate gland [15]. In order to classify that a tissue area represented by an ultrasound image is cancerous or non-cancerous we first classify the area as hypoechoic, isoechoic, hyperechoic or as a mixture. The sections are separated and each section is investigated independently from the others.
The algorithm presented in this thesis considers each pixel of the tissue area to be investigated as the center of a square pixel window having dimensions of 21x21 pixels. Similar approaches can be found in [17] and [13], but the algorithms developed in those papers use pixel windows of dimensions 9x9 and 16x16 pixels, respectively. The reason for choosing a 21x21 tissue segment is due to the fact that studies have shown that 0.2cc tumor sizes are likely to grow and develop into very significant diseases over a period of 10 to 15 years [30]. A size of 0.2cc corresponds roughly to a tissue segment of 21x21 pixels. Furthermore, a 21x21 tissue segment is also chosen since a large enough sample is needed in order to perform statistical and textural analysis. Since too large of a tissue segment could include healthy tissue and hinder recognition, larger tissue segments are

Figure 22. The pixel intensity histogram for a normal tissue segment. Note that it follows the normal probability distribution model.
not considered. Therefore, a tissue segment of dimensions 21x21 pixels is ideal in our study. The results obtained after analysis of the 21x21 pixel tissue segment are associated only with the center pixel of the current pixel window. Tissue recognition is performed by analysis of the 21x21 pixel tissue segment using statistical, textural and morphological methods. The mean, standard deviation and the normalized autocorrelation function at lag 1 are among the indicators used to determine if a tissue area is cancerous or non-cancerous. Furthermore, the pixel intensity histogram distributions of cancerous and non-cancerous tissue segments are also different; normal tissues have probability distributions that follow the normal probability distribution function while cancerous tissues have pixel values that are shifted towards zero and they follow the exponential probability distribution function (see figure 22 and 23). The

![Graph](image)

**Figure 23.** The pixel intensity histogram for a cancerous tissue segment. Note that it follows the exponential probability distribution model.
distribution of the histogram function is modeled using a gamma probability distribution function with parameters $\alpha$ and $\beta$. The method is discussed further in section two of this chapter. Similar approaches can be seen in [12] and [16]. The above statistical and textural indicators along with the $\alpha$ and $\beta$ parameters form a vector in the multidimensional space for every 21x21 pixel tissue segment. Furthermore, two different spaces are constructed: one space is the cancerous space and the other is the non-

Figure 24. The suspicious cancerous locations detected by the algorithm based only on statistical and textural analysis of the tissue segments.

cancerous space. These spaces are constructed by providing the program with learning sets of both cancerous and non-cancerous tissue segments of ultrasound images. Tissue recognition entails the computation of the vector of the predefined attributes for each of
the pixels in the tissue area of interest and subsequent tissue classification based on the smallest Mahalanobis distance between the vector of attributes and the two predefined centroids of the cancerous and non-cancerous spaces. This method is discussed further in section three of this chapter. This method alone yields similar results as previous detection algorithms and so large areas of prostate tissue are suspected of having cancer (see figure 24). Further processing of the suspicious areas is needed to obtain more accurate results.

Although hypoechoic areas are typically suspected to be cancerous lesions, other conditions yield non-cancerous hypoechoic regions that further interfere with the tissue analysis. Examples of conditions causing hypoechoic areas include muscle surrounding the prostatic urethra, the ejaculatory ducts, atrophic glands, as well as benign hypertrophy [15], [18]. Analysis of these hypoechoic areas yield spatial statistical properties similar to those with cancer and hence detection is severely hindered (see figure 24). Similar research has shown that merely texture analysis does not generally yield good detection results and hence further tissue analysis is needed [13], [16], [17].

Morphological tissue analysis has been performed in mammographic lesion characterization but does not appear anywhere in literature as being used in prostatic cancer recognition or analysis of tumors in the prostate gland [19], [34]. Morphological tissue analysis has been speculated in [17] as a possible solution to the problems caused by the inability of texture analysis methods to obtain good results on small pixel windows of prostatic ultrasound images. Unfortunately, this speculation was not investigated. The algorithm developed in this thesis uses a variation of the bending energy descriptor found in literature for performing structural analysis of the tissue (see section 2.4). If the
morphological analysis determines that a tissue structure does not appear to be cancerous, it is removed from the suspicious area. Figure 25 shows the recognized cancer in the ultrasound image from figure 24 after morphological tissue analysis has been performed. The morphological tissue analysis is the last step performed in the algorithm and the method is discussed further in section four of this chapter.

3.2 Statistical Properties and Indicators

As stated in section one of this chapter, due to tissue changes the ultrasound signal is absorbed differently between cancerous and non-cancerous tissue areas. The energy of the signal, the continuity of the signal, the autocorrelation and probability distribution functions are different in normal tissue than in cancerous tissue. The algorithm developed in this thesis considers every pixel as the center of a tissue segment having dimensions of 21x21 pixels. Furthermore, the algorithm uses seven attributes to determine if a pixel is a possible member of cancerous or non-cancerous tissue.

The ability to discriminate between cancerous and non-cancerous tissues entitles the knowledge of where cancer is most probable to occur. Since the majority of cancerous tumors occur at the locations that correspond to hypoechoic areas in the ultrasound image, an effective attribute is needed to differentiate between hypoechoic, isoechoic and hyperechoic areas of the ultrasound image. One such effective attribute that has also been investigated and applied elsewhere in literature is the mean $\mu$ of the tissue segment \[4], [15]. Let $P_y$ be the pixel at location $(i, j)$ of the tissue segment under investigation, then the first attribute towards recognition of cancerous tissue is the mean $\mu$ and it is defined as
By definition of the hypoechoic, isoechoic and hyperechoic areas, if $\mu_i$ is the mean of the entire tissue area of the ultrasound image then $A_i < \mu_i$ for hypoechoic tissue segments, $A_i = \mu_i$ for isoechoic tissue segments and $A_i > \mu_i$ for hyperechoic tissue segments.

Figure 25. The suspicious cancerous locations detected by the algorithm after the morphological tissue analysis.

Further investigations of tissue segments have yielded that, in general, cancerous tissue segments have less energy than non-cancerous tissue segments. This phenomenon is attributed to the fact that cancerous tissue does not reflect the ultrasound signal as
strong as non-cancerous tissue [8]. The intensity of pixels at hypoechoic tissue segments of the ultrasound image have relatively low values for cancerous tissues as opposed to non-cancerous tissues which have relatively higher pixel intensity values. An example of these characteristics can be seen on figures 29 and 30. Therefore, the second attribute used to discriminate between cancerous tissue and non-cancerous tissue in hypoechoic areas is the variance $\sigma^2$ of the corresponding tissue segment. If $\mu$ is the mean of the tissue segment then the variance $\sigma^2$ is defined as

$$A_2 = \sigma^2 = \frac{1}{441} \sum_{i=1}^{21} \sum_{j=1}^{21} (P_{ij} - \mu)^2.$$ 

In general cancerous hypoechoic tissue segments have a smaller variance than normal hypoechoic tissue segments.

Observations of the first order-density functions from the pixel intensities of tissue segments have indicated that cancerous and healthy tissue segments have different probability density functions. Cancerous tissue segments contain pixel values that are shifted towards the small values near zero and hence they follow the exponential probability distribution function (see figure 23). On the other hand, healthy tissue contains pixel values that follow the normal density function and hence it can be modeled using the Gaussian probability distribution function (see figure 22). Therefore, the distribution of the intensity histogram of any tissue segment is somewhere between that of exponential and Gaussian probability distribution functions for cancerous and non-cancerous tissues, respectively. This probability distribution function for any tissue segment is modeled here using a gamma probability distribution function with parameters $\alpha$ and $\beta$. More specifically, the gamma probability distribution function is of the form
\[ f(x) = \frac{x^{\alpha-1} e^{-\frac{x}{\beta}}}{\beta^\alpha \Gamma(\alpha)}, \]

where \( \alpha > 0 \) and \( \beta > 0 \). The mean \( \mu \) of the gamma probability distribution function is \( \mu = \alpha \cdot \beta \) and the variance is \( \sigma^2 = \alpha \beta^2 \). Estimates for \( \alpha \) and \( \beta \) can be obtained using the method of moments for the discrete case. Hence, for a specific tissue segment, given the distribution function of the intensity histogram \( H(i) \), let \( \hat{\mu} \) be the estimate of the mean and \( \hat{\sigma}^2 \) be the estimate of the variance, then \( \hat{\mu} \) and \( \hat{\sigma}^2 \) can be computed as follows:

\[ \hat{\mu} = \frac{\sum_{i=1}^{256} H(i-1) \cdot i}{\sum_{i=0}^{255} H(i)} = \frac{1}{441} \sum_{i=1}^{256} H(i-1) \cdot i \]

and

\[ \hat{\sigma}^2 = \frac{\sum_{i=1}^{256} (\mu - i)^2 \cdot H(i-1)}{\sum_{i=0}^{255} H(i)} = \frac{1}{441} \sum_{i=1}^{256} (\mu - i)^2 \cdot H(i-1). \]

From the above equation we obtain the estimated \( \hat{\alpha} \) and \( \hat{\beta} \) to be

\[ A_3 = \hat{\alpha} = \frac{\hat{\mu}^2}{\hat{\sigma}^2} \quad \text{and} \quad \hat{\beta} = \frac{\hat{\sigma}^2}{\hat{\mu}}. \]

Hence, \( \alpha \) is scale independent due to the fact that it is normalized by the variance. Since the gamma probability distribution function becomes the exponential probability distribution function when \( \alpha = 1 \) and \( \beta = \sigma \), and since the intensity histogram of cancerous tissue segments is modeled after the exponential probability distribution function, tissue segments that have estimated \( \hat{\alpha} \) and \( \hat{\beta} \) values approaching \( \hat{\alpha} \approx 1 \) and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
$\beta \approx \sigma$ are considered as cancerous tissue segments. Tissue segments having $\hat{a}$ and $\hat{b}$ values not approaching $\hat{a} \approx 1$ and $\beta \approx \sigma$ are considered to be free from cancer.

Observations of hypoechoic regions have indicated that tissue segments containing cancer are discontinuous with respect to healthy tissue segments. These discontinuities are caused due to the effects of the disease onto healthy tissue (see figure 20). Due to the discontinuities the autocorrelation function of tissue segments representing healthy tissue is different than the autocorrelation function of tissue segments representing cancerous tissue. Moreover, let $S$ be the tissue segment under consideration having mean $\mu$ and variance $\sigma^2$, the normalized autocorrelation function $\rho_1$ at lag 1 is defined as

$$A_2 = \rho_1 = \frac{1}{420 \cdot \sigma^2} \sum_{i=1}^{30} \sum_{j=1}^{21} (P_{ji} - \mu) \cdot (P_{ji+1} - \mu).$$

Therefore, since healthy tissue is relatively continuous with respect to cancerous tissue, the normalized autocorrelation functions at lag 1 of healthy tissue segments have values approaching one and tissue segments containing cancer have normalized autocorrelation functions with values that approach zero. Using this textural analysis approach the fifth attribute $A_2$ in the recognition process is defined.

Further observation on the texture of tissue segments has revealed that neighboring pixel intensity values have different correlations between cancerous and non-cancerous tissue segments. These correlations are exploited in order to develop the last two attributes in the recognition process.
Lemma 1.

For cancerous hypoechoic areas with $\alpha = 1$ and $\beta = \sigma$, let $X_1, X_2, X_3, \ldots, X_n$ be independent identically distributed random variables with mean zero and variance $\sigma^2_X$, then the distribution of the random variable $Y = X_i X_{i+k}$ has mean zero and variance equal to $\sigma^2_Y = \sigma^4_X$.

Proof

The mean $\mu_Y = E(X_i X_{i+k}) = E(X_i)E(X_{i+k}) = 0$. The variance of $Y$ is

$$\sigma^2_Y = E(Y^2) = E(X_i^2)E(X_{i+k}^2) = \sigma^4_X$$

Note that the variance is independent of the space lag.

Theorem 1.

Let $X_1, X_2, X_3, \ldots, X_n$ be a sequence of random numbers obtained from a wide sense stationary random process with mean zero and variance $\sigma^2_X$. Let also the space lag between $X_i$ and $X_{i+k}$ be fixed, and denoted by $\Delta X$. Then the random variable $Y = X_i X_{i+k}$ has mean positive and depended on the lag $k$. As $k$ increases the mean of $Y$ approaches to zero. The variance $\sigma^2_Y = E(X_{i+k}^2 E(X_i^2 | X_{i+k}))$ and $E(X_i^4) > \sigma^2_Y > \sigma^4_X$.

Proof

We have that

$$E(X_i^2 - E(X_i^2))^2 = E(X_i^4 - 2X_i^2 E(X_i^2) + E^2(X_i^2))$$

from the above equation we obtain

$$E(X_i^2 - E(X_i^2))^2 = E(X_i^4) - E^2(X_i^2)$$
or

\[ E(X_i^2 - E(X_i^2))^2 = E(X_i^4) - \sigma_{X_i}^4. \]

From the above equation, since the left side is nonnegative we obtain \( E(X_i^4) > \sigma_{X_i}^4. \)

Now, since \( \sigma^2_Y = E(X_i^2 X_{i+k}^2) = E(X_i^2 E(X_i^2 | X_{i+k})) \), as the space lag increases the correlation between \( X_i \) and \( X_{i+k} \) becomes smaller and tends to zero in which case \( \sigma^2_Y = \sigma_{X_i}^4. \) On the other hand, as \( k \) goes to zero \( \sigma^2_Y = E(X_i^4) \).

Theorem 2.

Consider the pixels \( X_{ij} \) and \( X_{ij+1} \), let \( Y = X_{ij} - X_{ij+1} \), then the expected value of \( Y \) is \( E(Y) = E(X_{ij} - X_{ij+1}) = 0 \) and ratio \( \frac{\sigma^2_Y}{\sigma^2_X} = 2(1 - \rho). \)

Proof

For \( Y = X_{ij} - X_{ij+1} \), since the random variables \( X_{ij} \) and \( X_{ij+1} \) have mean zero we then have

\[ E(Y) = E(X_{ij} - X_{ij+1}) = E(X_{ij}) - E(X_{ij+1}) = 0. \]

Furthermore, for the variance of \( \sigma^2_Y \) we have

\[
\sigma^2_Y = E(Y^2) = E(X_{ij} - X_{ij+1})^2 = E(X_{ij}^2 - 2X_{ij}X_{ij+1} + X_{ij+1}^2) = E(X_{ij}^2) - 2E(X_{ij}X_{ij+1}) + E(X_{ij+1}^2) = 2\sigma_{X_i}^2 - 2\rho,
\]

and hence

\[ A_o = \frac{\sigma^2_Y}{\sigma^2_X} = 2(1 - \rho). \]
Theorem 3.

Consider the pixel \( X_{ij} \) and the pixels \( X_{i,j-1}, X_{i,j+1}, X_{i-1,j}, X_{i+1,j} \) which have lag one from \( X_{ij} \). Now also consider \( Y = X_{ij} - 0.25(X_{i,j-1} + X_{i,j+1} + X_{i-1,j} + X_{i+1,j}) \) then the expected variable of \( Y \) is zero and the variance of \( Y \) is

\[
\sigma_Y^2 = \sigma_X^2 \left( 1.25 - 2\rho_1 + 0.25\rho_2 + 0.5\rho_{12} \right).
\]

Proof

\[
A_y = \frac{\sigma_Y^2}{\sigma_X^2} = 1.25 - 2\rho_1 + 0.25\rho_2 + 0.5\rho_{12}.
\]

Furthermore, let \( Y_{ij} = X_{ij} - \bar{X} \) then

\[
S_Y^2 = \frac{1}{441} \sum_{i=1}^{21} \sum_{j=1}^{21} Y_{ij}^2 \quad \text{and} \quad r_y = \frac{Y_{ij}}{S_Y}.
\]

Hence the distribution of the normalized data \( \frac{r_y r_{i,j+1}}{\sigma_r^2} \) is different for the cancerous and non-cancerous areas. From the above we infer that images with continuity and strong autocorrelation have normalized distribution of the lag product with variance which is greater than 1, where as the correlation decreases due to the discontinuities the normalized distribution of the lag product has variance close to 1. That implies if we consider the histogram of the normalized lag product multiplied by a factor 10, then in the non-cancerous areas we expect more energy namely bigger spread, than the cancerous areas.
3.3 Algorithm Training via Data Selection and Tissue Classification via Mahalanobis Distance Measurements

The previously defined attributes associated with one tissue segment constitute a vector in the multivariate space. Since each tissue segment is a pixel window of dimensions 21x21 pixels having a center pixel \( P_{ij} \), each pixel \( P_{ij} \) has a vector associated with it that characterizes the corresponding 21x21 tissue segment. Classification of pixel \( P_{ij} \) into either a cancerous or non-cancerous set is performed via weighting its corresponding attributes. This weighting is accomplished via a distance measurement between the vector of attributes of the tissue segment under consideration and two predefined centroids, one for cancerous and one for non-cancerous tissue areas. Once the two distances have been determined, if the distance from the non-cancerous centroid is smaller than that of the cancerous centroid then the center pixel \( P_{ij} \) of the tissue segment under consideration is classified as non-cancerous and placed in the non-cancerous set, otherwise \( P_{ij} \) is classified as cancerous and placed in the cancerous set. Each pixel from the cancerous set is painted red for as to show the location of cancer in the ultrasound image. Furthermore, although the distribution of each one of the attributes is not normal, which implies that their multivariate distribution is also not normal, the distribution of their centroid is normal due to the extension of the central limit theorem to the multivariate space. The Mahalanobis distance is used to determine the distance between the vector of attributes and the two centroids, one for cancerous and one for non-cancerous tissue areas. This section discusses in detail how the centroids for cancerous and non-cancerous tissue areas are obtained and how the Mahalanobis distance is used to classify a pixel as being cancerous or non-cancerous.
Comparison of how well the vector of attributes for a pixel $P_y$ matches against the average values of vector attributes from cancerous and non-cancerous tissue segments would allow for the classification of $P_y$. The first step towards classification of a pixel $P_y$ entails the determination of the average vector of attributes from multiple cancerous and non-cancerous tissue segments. Therefore, selections of cancerous and non-

![Ultrasound image of the prostate. Biopsy confirmed carcinoma with Gleason score 6 as indicated on the image.](image)

...cancerous regions of interest (ROI) from multiple ultrasound images are obtained for training the algorithm into determining better cancerous and non-cancerous attribute centroids, respectively. Examples of cancerous and non-cancerous ROI can be seen in figures 26, 27 and 28. Let $C(x, y, u)$ be a pixel from a cancerous ROI with location $(x, y)$ of an ultrasound image $u$ and let $N(x, y, u)$ be a pixel from a non-cancerous ROI...
Figure 27. Selection of cancerous ROI for algorithm training

Figure 28. Non-cancerous ultrasound image with multiple ROI's selected for algorithm training.
with location \((x, y)\) of an ultrasound image \(u\). Also, let \(\tilde{C}_{xyu}\) and \(\tilde{N}_{xyu}\) be the vectors of attributes of the tissue segment for the corresponding cancerous and non-cancerous center pixel, respectively. Therefore, we have that

\[
\tilde{C}_{xyu} = \begin{bmatrix}
A_1(x, y, u) \\
A_2(x, y, u) \\
A_3(x, y, u) \\
A_4(x, y, u) \\
A_5(x, y, u) \\
A_6(x, y, u) \\
A_7(x, y, u)
\end{bmatrix}
\quad \text{and} \quad
\tilde{N}_{xyu} = \begin{bmatrix}
A_1(x, y, u) \\
A_2(x, y, u) \\
A_3(x, y, u) \\
A_4(x, y, u) \\
A_5(x, y, u) \\
A_6(x, y, u) \\
A_7(x, y, u)
\end{bmatrix},
\]

where \(A_i(x, y, u)\) is the \(i^{th}\) recognition attribute defined in section 3.2 for the tissue segment from ultrasound image \(u\) having center pixel coordinates \((x, y)\). Hence, the cancerous centroid \(\bar{A}_c\) and non-cancerous centroid \(\bar{A}_N\) are defined as

\[
\bar{A}_c = \frac{\sum_{u} \sum_{x} \sum_{y} \tilde{C}_{x,y,u}}{N_C} \quad \text{and} \quad \bar{A}_N = \frac{\sum_{u} \sum_{x} \sum_{y} \tilde{N}_{x,y,u}}{N_N},
\]

where \(N_C\) is the total number of vectors of attributes from the cancerous ROI and \(N_N\) is the total number of vectors of attributes from the non-cancerous ROI. Note that although the distribution of each of the attributes \(A_i\) may not be normal as shown in section 3.2. By the central limit theorem we know that if \(A_{i1}, A_{i2}, A_{i3}, \ldots, A_{in}\) are a random sample from a population with mean \(\mu_i\) and standard deviation \(\sigma_i\) then if \(n\) is large, the probability model for \(\bar{A}_i\) is approximately normal with \(\mu_{\bar{A}_i} = \mu_i\) and \(\sigma_{\bar{A}_i} = \frac{\sigma_i}{\sqrt{n}}\).

Therefore, the cancerous and non-cancerous centroids, \(\bar{A}_c\) and \(\bar{A}_N\) respectively, for large values of \(N_C\) and \(N_N\) the centroids will have a normal distribution due to the extension.
of the central limit theorem to the multivariate space. Consequently, training the algorithm with large sets of ROI from multiple ultrasound images would generate a cancerous centroid with relatively small variance and hence, recognition of cancer would improve. Moreover, the computation of the two variance-covariance matrices, \( \Sigma_C \) and \( \Sigma_N \), resulting from \( N_C \) and \( N_N \) vectors of attributes from the selected ROI must also be determined in order to compute the Mahalanobis distance.

The autocovariance \( \sigma_{i,j} \) between two attributes \( i \) and \( j \) having \( N \) samples each is defined as

\[
\sigma_{i,j} = \frac{\sum_{k=0}^{N-1} (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{N},
\]

where \( A_{ik} \) is the \( k \)th sample attribute of \( i \), \( A_{jk} \) is the \( k \)th sample attribute of \( j \), \( \mu_i \) is the mean of the \( N \) samples of attribute \( i \) and \( \mu_j \) is the mean of the \( N \) samples of attribute \( j \).

The cross-correlation \( \rho_{i,j} \) between attributes \( i \) and \( j \) is defined as

\[
\rho_{i,j} = \frac{\sum_{k=0}^{N-1} (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{\sigma_i \sigma_j},
\]

where \( \sigma_i \) and \( \sigma_j \) are the standard deviations of the \( N \) samples of the \( i \) and \( j \) attributes, respectively. We are now able to define the covariance matrix. The cancerous variance-covariance matrix denoted as \( \Sigma_C \) is defined as
where $\sigma_{i,j}$ and $\rho_{i,j}$ are computed from the $N_C$ samples chosen from multiple ultrasound images and over cancerous ROI. Similarly, $\sum_{N} \sigma$ is also defined but by using $\sigma_{i,j}$ and $\rho_{i,j}$ that are computed from the $N_N$ samples chosen from multiple ultrasound images and over non-cancerous ROI. With relatively good cancerous and non-cancerous centroids having been defined by a relatively large sample of tissue segments from corresponding ROI, the recognition process via Mahalanobis distance is now possible.

As stated previously, each pixel $P_{i,j}$ is the center of a corresponding tissue segment characterized by a vector of attributes $\vec{A}_{i,j}$. Classification of $P_{i,j}$ into the possibly cancerous and non-cancerous sets is achieved by measuring the distance between the vector of attributes $\vec{A}_{i,j}$ and the two predefined centroids, $\vec{A}_C$ and $\vec{A}_N$.

Since the principal component analysis of the recognition attributes produces eigenvectors with directions dependent on presence or absence of cancer in the corresponding tissue segment, the Mahalanobis distance is used to correctly determine the minimum distance between the two centroids $\vec{A}_C$ and $\vec{A}_N$. The Mahalanobis distances $D_C$ and $D_N$ between a vector $\vec{A}_{i,j}$ and centroids $\vec{A}_C$ and $\vec{A}_N$ respectively, is defined as

$$
\sum_{C} = \begin{bmatrix}
\sigma_1^2 & \sigma_{1,2} & \sigma_{1,3} & \sigma_{1,4} & \sigma_{1,5} & \sigma_{1,6} & \sigma_{1,7} \\
\sigma_{2,1} & \sigma_2^2 & \sigma_{2,3} & \sigma_{2,4} & \sigma_{2,5} & \sigma_{2,6} & \sigma_{2,7} \\
\sigma_{3,1} & \sigma_{3,2} & \sigma_3^2 & \sigma_{3,4} & \sigma_{3,5} & \sigma_{3,6} & \sigma_{3,7} \\
\sigma_{4,1} & \sigma_{4,2} & \sigma_{4,3} & \sigma_4^2 & \sigma_{4,5} & \sigma_{4,6} & \sigma_{4,7} \\
\sigma_{5,1} & \sigma_{5,2} & \sigma_{5,3} & \sigma_{5,4} & \sigma_5^2 & \sigma_{5,6} & \sigma_{5,7} \\
\sigma_{6,1} & \sigma_{6,2} & \sigma_{6,3} & \sigma_{6,4} & \sigma_{6,5} & \sigma_6^2 & \sigma_{6,7} \\
\sigma_{7,1} & \sigma_{7,2} & \sigma_{7,3} & \sigma_{7,4} & \sigma_{7,5} & \sigma_{7,6} & \sigma_7^2
\end{bmatrix},
$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ D_C = (\vec{A}_{i,j} - \vec{A}_C)^\top \sum_c^{-1} (\vec{A}_{i,j} - \vec{A}_C) \quad \text{and} \]
\[ D_N = (\vec{A}_{i,j} - \vec{A}_N)^\top \sum_N^{-1} (\vec{A}_{i,j} - \vec{A}_N) \]

where \((\vec{A}_{i,j} - \vec{A}_C)^\top\) and \((\vec{A}_{i,j} - \vec{A}_N)^\top\) are the transpose of vectors \((\vec{A}_{i,j} - \vec{A}_C)\) and \((\vec{A}_{i,j} - \vec{A}_N)\), respectively. Additionally, \(\sum_c^{-1}\) and \(\sum_N^{-1}\) are the inverse of the variance-covariance matrices defined above. Finally, the pixel \(P_{i,j}\) is classified as suspicious and belonging to cancerous tissue if \(D_C < D_N\), else it is classified as belonging to a non-cancerous tissue. Note that this is not the final classification. A classification category refinement is still performed via morphological tissue analysis to further determine if a cancerous pixel has been misclassified and to correct this misclassification by placing the pixel from the cancerous set to the non-cancerous set. The description of this method is given in the next section.

3.4 Final Pixel Classification into Cancerous or Non-Cancerous Sets via Morphological Tissue Analysis

Although hypoechoic areas are typically suspected to be cancerous lesions, other conditions yield non-cancerous hypoechoic regions that further interfere with the tissue analysis. Examples of conditions causing hypoechoic areas include muscle surrounding the prostatic urethra, the ejaculatory ducts, atrophic glands, as well as benign hypertrophy \([15], [18]\). Analysis of these hypoechoic areas yield spatial statistical properties similar to those with cancer and hence detection is severely hindered. Similar research has shown that merely texture analysis does not generally yield good recognition results and hence further tissue analysis is needed \([13], [16], [17]\). Indeed, cancer analysis of our
algorithm on ultrasound images without performing morphological tissue analysis does not yield good results as can be seen in figure 24. The final step in the detection algorithm incorporates a morphological tissue analysis scheme on the cancerous tissue pixels defined by the Mahalanobis distance measurement in section 3.3. The morphological tissue analysis tries to measure the likelihood of a pixel $P_{i,j}$ as being part of a cancerous tissue. This is accomplished by analyzing the tissue structure of the corresponding 21x21 tissue segment and by assigning a continuity value to pixel $P_{i,j}$ based on the border continuity of the tissue object. Figures 29 and 30 show the differences between cancerous and non-cancerous tissues and their corresponding tissue objects.

Figure 29. Magnified portion of a cancerous region

Figure 30. Magnified portion of a normal region

Figure 31. Binarized image of cancerous tissue segment. The total number of tissue objects is four with an average MBE = 0.68

Figure 32. Binarized image of normal tissue segment. The total number of tissue objects is two with an average MBE=0.42

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
In order for pixel $P_{i,j}$ to undergo morphological tissue analysis the corresponding 21x21 tissue segment must be binarized in order to reveal the underlying tissue structure (see figures 31 and 32). A tissue segment binarization using a global threshold, like the one discussed in section 4.2, cannot be used here due to its inability to obtain good results between tissue segments with different contrast levels (see figure 33). Therefore, a locally adaptive, statistical threshold is first computed for the 21x21 tissue segment and is then used as a parameter into a non-linear filter kernel which is used to shrink and binarize the tissue for a clear separation of the underlying multiple tissue objects (see figures 29-32) [21]-[23].

![Tissue segment binarization using the local mean as the threshold.](image)

Each tissue segment can be considered as a function $z = f(x, y)$, where $x$ and $y$ are the pixel coordinates and $z$ is the intensity of pixel $P_{x,y}$. The corresponding 3-D representations of the tissue segments in figures 29 and 30 can be seen in figures 34 and 35, respectively. Binarization via simple thresholding strives to suppress low $z$ values from a function by making all $z$ values lower than a threshold $T_b$ equal to zero and values of $z$ greater or equal than $T_b$ equal to one. The results of binarization of two different tissue segments can be seen in figures 31 and 32. The purpose for applying this non-
linear filter on the tissue segment is to differentiate between the tissue and the underlying tissue structure. Hence a more effective method for suppressing lower $z$ values and enhancing higher $z$ values is needed. This is accomplished by applying for each pixel in the tissue segment an exponential filter which is based on the local statistical attributes of the tissue segment under consideration. Let $\mu$ and $\sigma^2$ be the mean and variance of the tissue segment under consideration, respectively. Also let $c$ be a constant such that $c>1$.

Consider the equation $q^{\mu+\sigma} = c$, solving for $q$ we obtain

$$q^{\mu+\sigma} = c$$

$$(\mu + \sigma)\ln(q) = \ln(c) \quad \text{or}$$

$$q = e^{\mu+\sigma}.$$ 

For each pixel $P_{i,j}$ of the tissue segment we define a new pixel $P'_{i,j}$ such that

$$P'_{i,j} = P_{i,j} \cdot \frac{q_{i,j}^{\mu+\sigma}}{c}.$$
Note that the pixel $P_{i,j}$ is multiplied by a scaling coefficient derived from an exponential function, namely $\frac{q_{P_{i,j}}}{c} = \left( \frac{\ln(c)}{e^{\mu+\sigma}} \right)^{P_{i,j}}$. Hence, the coefficient $k = \frac{q_{P_{i,j}}}{c}$ has the following characteristics

$$ k = \begin{cases} 
  k < 1, & \text{if } P_{i,j} < (\mu + \sigma) \\
  k = 1, & \text{if } P_{i,j} = (\mu + \sigma) \\
  k > 1, & \text{if } P_{i,j} > (\mu + \sigma) 
\end{cases} $$

Therefore, each pixel $P_{i,j}$ is exponentially suppressed to a new value $P'_{i,j}$ if $k < 1$ or enhanced to a new value $P'_{i,j}$ if $k > 1$. The effects of this exponential filter kernel can be seen in Figure 36. Binarization is now performed on pixel $P'_{i,j}$ to create a new pixel $P''_{i,j}$ such that

![Figure 36. The effects of the exponential filter on function $f(x) = x$.](image)
Results of the Binarization of two tissue segments into tissue objects via the use of the exponential filter kernel can be seen in figures 31 and 32.

The morphological tissue analysis for a tissue segment with center pixel \( P_{x,y} \) entails the morphological analysis of the tissue objects within the tissue segment. Observations of tissue objects indicate that an object's border is more rigid in cancerous tissue objects than in non-cancerous tissue objects (see figures 31 and 32). This phenomenon is attributed to the discontinuities created by the disease on the prostatic tissue. Each tissue object is assigned a value which is based on the smoothness of the object's border. The average of all these values describe how smooth the tissue borders are within the tissue segment. Due to the relative small size of the tissue objects, morphological analysis of tissue objects using the methods described in chapter 2.4 did not yield encouraging results and a new method, which is based on the bending energy scheme, was developed. Let \( a_1, a_2, a_3, \ldots, a_L \) be the chain code in 8-connectivity of a tissue object border. The definition of the modified bending energy of a tissue object border is as follows:

\[
MBE = \frac{1}{L} \sum_{k=1}^{L} c(k)
\]

where \( L \) is the border length and \( c(k) \) is a function based on the chain code such that

\[
c(k) = 1, \text{ if the chain codes at location } k-1 \text{ and } k \text{ are different, and}
\]

\[
c(k) = 0, \text{ if the chain codes at location } k-1 \text{ and } k \text{ are the same.}
\]
Morphological analysis results based on the modified bending energy scheme indicate that tissue segments having an average $MBE$ value for all the tissue objects of the tissue segment greater than 0.6 correspond to cancerous tissue segments. Therefore, if a pixel $P_{i,j}$ that was previously suspected of being cancerous can be placed into the non-cancerous set if its $MBE$ value is less than or equal to the threshold value of 0.6. Figures 24 and 25 show the results of the algorithm classification before and after morphological tissue analysis, respectively.
CHAPTER 4

PRACTICUM

4.1 Algorithm Overview

The developed program operates on 8-bit grayscale ultrasound images that are physically stored on the computer as bitmap images (see figure 6). Since cancer can metastasize and spread to the surrounding tissue, recognition of cancer by the program should not be limited only to the prostate gland in the bitmap images but it should also include the neighboring tissue (see figure 8). One approach of tissue selection for recognition is for the doctor to manually select the area for analysis using a device such as a mouse. This method, although correct, would not be very practical for cancer recognition in large sets of prostate ultrasound images since it would require the aid of humans for the manual selection of regions of interest (ROI). Hence an automated segmentation of the ROI is developed in section two of this chapter. Once the ROI for cancer detection has been determined it is then used by the recognition algorithm to analyze the corresponding ultrasound image area. This method is further explained in section three of this chapter.
4.2 Automatic Region of Interest Detection

Since the recognition algorithm may be applied on large sets of ultrasound images, automated extraction of ROI for cancer analysis is crucial. A set of filters is applied on ultrasound images for the purpose of extracting ROI's, namely the tissue portion of an ultrasound image that is to be analyzed for cancer (see figure 8). There are four main steps for the extraction of the ROI. The first step involves the sub-frame extraction from the ultrasound image. This sub-frame has been highlighted in figure 37 and may be located anywhere within the ultrasound image. Once the sub-frame has been extracted, the second step involves the removal of characters that appear in the sub-frame.

Figure 37. The ultrasound image from figure 8 with the sub-frame highlighted.
(see figure 43). The third step involves the tissue boundary detection after the character removal from the sub-frame (see figure 45). The final step involves the fitting of a B-Spline through the boundary points in order to create a continuous and solid boundary (see figure 46).

The ultrasound tissue image that is contained within the sub-frame is enclosed within four line segments that form a rectangle (see figure 37). The first step towards sub-frame segmentation is the automatic detection of the four line segments. A Binarization approach is used to first segment the image from a 256 pixel intensity grayscale image to a binary image. We apply the following filter for Binarization:

For every pixel $P_{ij}$ of the ultrasound image, a new pixel $P'_{ij}$ is generated such that

$$P'_{ij} = \begin{cases} 0, & \text{if } P_{ij} < T_b \\ 1, & \text{if } P_{ij} \geq T_b \end{cases}$$

where the constant threshold value $T_b = 60$ and it has been predetermined based on the average pixel values that generate the line segments. The new binarized image can be seen in figure 38.

The next step in the process is to perform an edge detection algorithm so that only the line segments and other image boundaries are visible. Since the only interest is in edge magnitudes without regard to their orientations, the Laplacian linear differential operator is used. The Laplacian operator is defined as follows:

$$\nabla^2 g(x, y) = \frac{\partial^2 g(x, y)}{\partial x^2} + \frac{\partial^2 g(x, y)}{\partial y^2}.$$  

The Laplacian operator $\nabla^2 g(x, y)$ acts on the second derivative of a signal $g(x, y)$ and it is used to measure the "smoothness" of that signal. A signal that has minimum fluctuations...
will have small Laplacian values compared to a signal that has large fluctuations. This is due to the fact that the second derivative values of the first signal will be less than the second derivative values of the second signal. The above Laplacian equation is approximated in digital images by a convolution sum between the 2-D image signal and a mask $h$ [20]. Each pixel $P_i$ of the image is considered as the center of a 3x3 matrix. The

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 38. Binarized ultrasound image
The convolution sum between $P_{ij}$ and $h$ is defined as:

$$P'_{ij} = P_{ij} * h = \begin{bmatrix} P_{i-1,j-1} & P_{i-1,j} & P_{i+1,j-1} \\
                           P_{i-1,j} & P_{i,j} & P_{i+1,j} \\
                           P_{i-1,j+1} & P_{i+1,j+1} & P_{i+1,j+1} \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\
                           -1 & 4 & -1 \\
                           0 & -1 & 0 \end{bmatrix} = 4P_{ij} - (P_{i-1,j-1} + P_{i+1,j-1} + P_{i-1,j+1} + P_{i+1,j+1})$$

Applying a threshold $T_h$ to $P'_{ij}$ yields the new Laplacian filtered image $P''_{ij}$ such that

$$P''_{ij} = \begin{cases} 0, & \text{if } P'_{ij} < T_h \\ 1, & \text{if } P'_{ij} \geq T_h \end{cases}$$

The threshold $T_h$ was chosen experimentally to be 1. The Laplacian filter acts as a high-pass filter and its effects can be seen in figure 39.

![Figure 39. Results of the Laplacian filter on the binarized ultrasound image.](image-url)
Once the line segments and other image boundaries have been segmented, the line segment detection process is initiated. Hough transforms have been used extensively in literature for object detection, such as lines and circles, in edge-segmented images [20]. The Hough transform is a technique that can be used to isolate features of a particular shape within an image. The parametric equation of a line is defined as
\[ r = x \cdot \cos(\theta) + y \cdot \sin(\theta), \]
where \( r \) is the length of a normal from the origin to this line and \( \theta \) is the orientation of \( r \) with respect to the x-axis (see figure 40). During the object recognition phase, the coordinates \((x_i, y_i)\) of the pixel points of edge segments from the image are known and therefore serve as constants in the above parametric line equation, while \( r \) and \( \theta \) are the unknown variables that need to be determined. The Hough transform takes a single point from the Cartesian image space to curves in the Hough space by plotting all possible \((r, \theta)\) values defined by each \((x_i, y_i)\). Points that are collinear in the Cartesian image space yield curves that intersect at a common \((r, \theta)\)
point in the Hough space. This single \((r, \theta)\) point determines a line in the Cartesian image space. Since the line segments forming the sub-frame are only horizontal and vertical, the program will select only the four largest line segments having angles \(\theta = 0^\circ\) and \(\theta = 90^\circ\). The effects of the line detection and the sub-frame extraction can been seen in figure 41.

![Figure 41. Extracted sub-frame](image)

The removal of characters from the sub-frame image is very crucial to the analysis phase. Due to their statistical properties, characters must be removed from the ROI so that no erroneous recognition occurs. This character removal is accomplished by a series of filters that are applied to the newly extracted sub-frame image. The first filter
applied is the above Laplacian edge detection filter with threshold $T_s = 160$. This filter is applied to segment the edges of the characters from the remaining tissue image. Unfortunately, some noise is incorporated to the edge-segmented image due to sharp tissue discontinuities from the original sub-frame image (see figure 42). This noise is removed by applying a noise removal filter. The noise removal filter is defined as a function $NRF$ having four variables ($x, y, \text{perimeter, frequency}$). The $x$ and $y$ variables represent the horizontal and vertical offsets into the image, respectively. The $\text{perimeter}$
variable represents the dimensions of the block with center pixel $P_{x,y}$. As an example consider \textit{perimeter} = 2. This implies a block having dimensions 3x3 with center pixel $P_{x,y}$. A perimeter value of six would imply a block having dimensions 7x7 with center pixel $P_{x,y}$. Given the above parameters, $x$, $y$ and \textit{perimeter}, the \textit{frequency} variable represents the minimum number of edge pixels that can be found in the block with center $P_{x,y}$ so as for the corresponding block not to be considered as having noise and removed.

Therefore, for every edge pixel $P_{x,y}$ of the image,
\[ NRF(x, y, \text{perimeter}, \text{frequency}) = 1, \] if the corresponding block defined by attributes \( x, y \) and \( \text{perimeter} \) has a total edge pixels greater than \( \text{frequency} \) and in which case \( P_{x,y} \) remains as one.

\[ NRF(x, y, \text{perimeter}, \text{frequency}) = 0, \] if the corresponding block defined by attributes \( x, y \) and \( \text{perimeter} \) has a total edge pixels less than \( \text{frequency} \). In this case the entire corresponding block is zeroed so as for noise to be removed.

Once the noise has been eliminated using \( \text{perimeter} = 30 \) and \( \text{frequency} = 5 \), the remaining edge pixels are considered. Each edge pixel is considered as the center of a

---

Figure 44. Superset of tissue border pixels.
square block having dimensions 31x31 pixels. The character removal entitles setting these blocks to zero. Figure 43 shows the results of the character removal from the sub-frame image.

Determining a set of pixels that are part of the tissue border is very crucial to the border segmentation scheme. Once this set has been determined, B-Spline interpolation can then be performed to generate a smooth and continuous border around the tissue area that is the ROI. The following filters are applied for finding a set of segmented points that lie on the tissue border. Binarization of the current sub-frame image is performed with $T_b = 10$. The Laplacian filter with $T_b = 1$ and the $NRF$ filter with \textit{perimeter} = 30
and frequency = 28 are then used to obtain a superset of the edge pixels that are part of the tissue border (see figure 44). The border is determined by keeping only the edge pixels that are on the outer border of the sub-frame image (see figure 45). This is accomplished by scanning the sub-frame image with a virtual scan line. A vertical scan line is used to scan the sub-frame image from left to right in order to determine which edge pixels belong to the tissue border. The pixels first encountered by the scan line are considered as being part of the left side of the tissue border while all other are considered not to be part of the tissue border. The vertical scan line is then used to scan the sub-frame image from right to left to further determine which edge pixels are part of the right side of the tissue border. A similar approach is taken with a horizontal scan line in order to determine the top and bottom side of the tissue border. The final result after applying the noise removal filters is the detection of border segments that need to be joined together in order to create the final continuous tissue border (see figure 45).

Cubic Bézier curves are used towards the construction of a smooth and continuous tissue border from the previously determined points. Since the Bézier curve is cubic, four points are required each time as the control points for defining a single curve. Let \( P_0, P_1, P_2 \) and \( P_3 \) be the control points having coordinates \((x_0, y_0), (x_1, y_1), (x_2, y_2)\) and \((x_3, y_3)\) respectively. The curve generated by the above control points is defined as follows:

\[
x = x_0 \cdot (1-u)^3 + x_1 \cdot 3u \cdot (1-u)^2 + x_2 \cdot 3u^2 \cdot (1-u) + x_3 \cdot u^3
\]

\[
y = y_0 \cdot (1-u)^3 + y_1 \cdot 3u \cdot (1-u)^2 + y_2 \cdot 3u^2 \cdot (1-u) + y_3 \cdot u^3
\]
where, $0 \leq u \leq 1$. The results of fitting the points with a Cubic Bézier curve and then flooding the interior of the border to generate the ROI can be seen in figure 46. With the ROI determined, analysis of the ultrasound image is now possible.

![Figure 46. ROI after fitting the border points with Cubic Bézier curves.](image)

**4.3 Image Analysis and Classification**

Analysis of the ROI of an ultrasound image entails the application of the algorithm described in chapter 3 for each pixel $P_{i,j} \in ROI$. More specifically, given a pixel $P_{i,j} \in ROI$, the seven recognition attributes $A_1, A_2, ..., A_7$ are computed from the corresponding 21x21 tissue segment. Furthermore, the predetermined centroids, $\hat{A}_c$ and
\( \hat{A}_n \), and variance-covariance matrices, \( \sum_C \) and \( \sum_N \), are loaded from the database for cancerous and non-cancerous ultrasound image areas, respectively. The two Mahalanobis distances, \( D_C \) and \( D_N \), are computed from the vector of attributes and the two centroids, \( \hat{A}_C \) and \( \hat{A}_N \). The pixel \( P_{i,j} \) is considered suspicious if \( D_C < D_N \) and are placed in the cancerous set \( S_C \). Lastly, for each \( P_{i,j} \in S_C \), if the morphological tissue analysis for the corresponding tissue segment yields a \( MBE < 0.6 \) then \( P_{i,j} \) is considered to have too many smooth tissue objects in its corresponding tissue and it is removed from the cancerous set \( S_C \). The remaining pixels that are in \( S_C \) are classified as belong to cancerous tissue and are colored red to indicate cancer. Classification of cancerous pixels can be seen in figures 49 through 53.
CHAPTER 5

CONCLUSION

5.1 Summary and Results

A prostate cancer recognition algorithm of ultrasound images was developed in this thesis. The algorithm is based on calculating seven attributes in the tissue segment of interest and then finding the Mahalanobis distance in the multivariate space of the vector of attributes from two predetermined centroids, one for cancerous regions and one for non-cancerous regions. If the distance from the non-cancerous centroid is smaller than the distance from the cancerous centroid then the region is classified as non-cancerous, otherwise it is classified as being a probable cancerous region. Morphological tissue analysis is then performed on the probable cancerous regions to further determine if they have structural tissue characteristics resembling those of cancerous tissue or normal tissue. This study was conducted on eighteen B-mode ultrasound images of which eight had carcinoma and ten were free from cancer. A total of eight images were used to build the database for the predefined cancerous and non-cancerous tissue regions. A total of five ultrasound images containing no cancer were used for the selection of the non-cancerous tissue regions for the database while three of the images had carcinoma and were used for the selection of the cancerous tissue regions for the database construction. From the total of 8 ultrasound images with carcinoma, one image containing isoechoic cancerous tissue was not detected by our program at the same location as the medical
doctor had indicated (see figure 47). Furthermore, our algorithm also failed to detect one of the two selected isoechoic cancerous regions in an ultrasound image containing both hypoechoic and isoechoic cancerous regions (see figure 48). On the other hand, our algorithm was able to detect the hypoechoic cancerous regions from all the cancerous ultrasound images. Our algorithm also detected metastasis of the cancer from the prostate gland to the surrounding tissue on two of the examined ultrasound images (see figures 50 and 51). Unfortunately, we were unable to confirm this via biopsy since the patients had already undergone treatment for the disease. No cancer was detected in the cancer free ultrasound images. Based on a limited number of data, our algorithm has the
ability to detect cancer in cancerous ultrasound images. There was one false negative classification of our algorithm as can be seen in figure 47. All cancer free ultrasound images were classified by our program as true negative. The majority of the cancerous ultrasound images were classified correctly as true positive (see figures 48-52) even though our algorithm indicated that on some ultrasound images the cancer had metastasized to the surrounding tissue outside the prostate gland (see figures 50 and 51). Our algorithm has the ability to detect cancer in ultrasound image regions which are consistent with the findings of needle biopsies performed by the medical doctor. We conclude that application of our algorithm on ultrasound images prior to a needle biopsy
would reduce the number of false negatives biopsies and increase the number of true positives biopsies since the medical doctor would be able to guide the biopsy needle at more probable cancerous tissue regions as detected by our algorithm. Likewise, prostate seed implantation would also benefit with the use of our algorithm since probable cancerous tissue regions can be detected by our algorithm and implantation of radiation seeds could be concentrated at those regions. Furthermore, since metastasis of the cancer from the prostate gland to the surrounding tissue was also detected by our algorithm, our algorithm could be used as a tool for staging prostate cancers. Based on a limited number of ultrasound images we concluded that our algorithm is capable of detecting cancer in cancerous ultrasound images.

5.2 Further Research

Due to the inability to perform biopsies on the patients having metastasized prostate cancer to the surrounding tissue as detected by our algorithm, we were unable to conclude with certainty if cancer had truly metastasized. Further research is needed to indicate if our findings with respect to metastasized cancer are correct or not. False negative results obtained from the cancerous isoechoic regions are worrisome and further research is needed in order to concluded if our algorithm was unable to detect cancer in some isoechoic regions or if the medical doctor had erroneously selected a normal isoechoic region. Due to the limited number of ultrasound images available, extensive testing with large sets of ultrasound images is needed before any practical application of the algorithm could be performed.

Furthermore, in order for cancer detection to be more effective, 3-D models of the prostate gland constructed from multiple ultrasound images could also be analyzed for
prostate cancer. Visualization of cancerous tumors detected during the 3-D analysis of the prostate gland would yield important information as of the shape and size of the cancerous tumors that would further aid in the diagnosis and treatment of the disease. Further research is needed for the development of more effective tools for the diagnosis of the disease.

![Image of cancerous tissue detected by the algorithm](image)

Figure 49. Cancerous tissue detected by the algorithm. Note the location of the carcinoma is indicated by the doctor after a biopsy procedure was performed.
Figure 50. Cancerous tissue recognized by the algorithm (in red) and the cancerous location detected by the biopsy (white region). Note that metastasis is detected (left bottom side of the image).
Figure 5.1. Cancerous tissue recognized by the algorithm (in red) and the cancerous location detected by the biopsy (white region). Note that metastasis is detected (left bottom side of the image).
Figure 52. Cancerous tissue detected by the algorithm. Note the location of the carcinoma is indicated by the doctor after a biopsy procedure was performed.
APPENDIX I

COMPUTER PROGRAM LISTINGS

/*
Analysis_Detection.h

This class will receive an image pointer and a boolean pointer that
tells the area of the image to be analyzed. The results of the
analysis are stored in m_CancerImg.
*/

#if !defined(H_ANALYSIS_AND_DETECTION_INCLUDED_)
define H_ANALYSIS_AND_DETECTION_INCLUDED_
#define BLOCK_SIZE 20
#define NUMBER_OF_ATTRIBUTES 7

class AnalysisAndDetectionClass
{

public:

double BendingEnergy(int *PtrImg, unsigned int width, unsigned int height);

int TraceBorder(int *PtrImgTmp, unsigned int width, unsigned int height, CPoint CurrLoc, int OldDirection);

AnalysisAndDetectionClass()
{
    m_CancerImg = NULL;
    m_width = m_height = 0;
    m_pf_LSC = m_pf_LSNC = m_pf_LSP = NULL;
} //PtrImag points to the image to be analyzed and BoarderArray points

75

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
//to the area to be analyzed.

void AnalysisDetectionSetup(unsigned char *PtrImg, bool *BoarderArray,
unsigned int width, unsigned int height)
{
    m_PtrImg = PtrImg;
    m_BoarderArray = BoarderArray;
    m_width = width;
    m_height = height;

    if(m_CancerImg != NULL && (m_width != width || m_height != height))
    {
        delete [] m_CancerImg;
        m_CancerImg = new int[width*height];
    }
    else
    m_CancerImg = new int[width*height];
}

~AnalysisAndDetectionClass() { delete [] m_CancerImg; };

double m_local_attributes[NUMBER_OF_ATTRIBUTES];
//attributes are arranged in array as follows:
//mean -> 0 element.
//variance -> 1 element.
//alpha -> 2 element.
//beta -> 3 element.
//R1 -> 4 element.
//ratio_B -> 5 element.
//ratio_V -> 6 element.

void Analyze(unsigned int x, unsigned int y); //analyzes the current
       //21x21 block and sets the m_local_attributes variable.

void CancerDetectionQ;

void LearnCancer(bool *selected_area);
//it will use the selected_area and compute all the attributes for each
//pixel in this area and write the attribute values for each pixel to
//the file (m_LSC).
void LearnNonCancer(bool *selected_area);
//same as LearnCancer() but only for non cancerous areas.

void LearnPeriphary(bool *selected_area);
//same as LearnCancer() but only for periphery areas...

//Function starts a new learning session for cancer and stores results
//in file "filename".
    int OpenNewLearningSessionCancer(CString filename);

//Function starts a new learning session for non cancer areas and
//stores results in file "filename".
    int OpenNewLearningSessionNonCancer(CString filename);

//Function starts a new learning session for the periphery areas and
stores results in file "filename".
    int OpenNewLearningSessionPeriphary(CString filename);

void CloseLearningSessionCancer(); //This will close the file m_LSC.
void CloseLearningSessionNonCancer(); //This will close the file m_LSNC.
void CloseLearningSessionPeriphary(); //This will close the file m_LSP.

//saves the learning session so that it can be loaded next time...
//(so that we wouldn't have to "teach" the computer every time...)
    int SaveLearnedSession(CString filename);

//loads a previously saved session to be used in the cancer detection...
    int LoadPreviousSession(CString filename);

void ComputeVCM(CString, double[NUMBER_OF_ATTRIBUTES]
    *[NUMBER_OF_ATTRIBUTES], double Centroid[NUMBER_OF_ATTRIBUTES]);
//it finds the covariance matrix and it builds the inverted covariance
//matrix... input is a cstring to the learned data... output is the
//7x7 inverted covariance matrix.

//location of the results for the learning session of cancer...
CString m_LSC;

//location of the results for the learning session of non cancer...
CString m_LSNC;

//location of the results for the learning session of periphery...
CString m_LSP;
double m_InvVCM_C[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES],
m_InvVCM_NC[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES],
m_InvVCM_P[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
//Inverses of covariance matrices of cancer and non-cancer and periphery respectively (of our learning data set).

double m_CentroidC[NUMBER_OF_ATTRIBUTES];
double m_CentroidNC[NUMBER_OF_ATTRIBUTES];
double m_CentroidP[NUMBER_OF_ATTRIBUTES];
//Keeps the centroid of the learned cancer, non-cancer, periphery data.

//pointer to the learning files for cancer and non cancer and periphery.
FILE *m_pf_LSC, *m_pf_LSNC, *m_pf_LSP;

unsigned char *m_PtrImg; //points to the image to be analyzed...
bool *m_BoarderArray;  //points to the area of the image to be
                        //analyzed...

unsigned int m_width;
unsigned int m_height;
int *m_CancerImg; //The results after the m_PtrImg has been analyzed...

//This function will compute the Inverse of the matrix passed.
int MatrixInversion(double VCMI[NUMBER_OF_ATTRIBUTES]
[NUMBER_OF_ATTRIBUTES]);

int m_ucolors[266*10]; //temp buffer.
double m_AutoCorrH[BLOCK_SIZE];
double m_AutoCorrV[BLOCK_SIZE];
double m_AutoCorrH_std[BLOCK_SIZE];
double m_AutoCorrV_std[BLOCK_SIZE];
int m_PxlMin, m_PxlMax;

#endif
void AnalysisAndDetectionClass::Analyze(unsigned int x, unsigned int y)
/{
  analyzes the current 21x21 block and sets the m_local_attributes variable.
/*
{   int i, j;

  for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
    m_local_attributes[i] = 0.0;

  //Remember:
  //mean -> 0 element.
  //variance -> 1 element.
  //alpha  -> 2 element.
  //beta   -> 3 element.
  //R1     -> 4 element.
  //ratio_B -> 5 element.
  //ratio_V -> 6 element.

  for(j = -BLOCK_SIZE/2; j <= BLOCK_SIZE/2; j++)
    for(i = -BLOCK_SIZE/2; i <= BLOCK_SIZE/2; i++)
    {
      if( double(m_PtrImg[x+i+( (y+j) * m_width )]) >
          m_local_attributes[7])
        m_local_attributes[7] = double(m_PtrImg
          [x+i+( (y+j) * m_width )]);
    }

  //First compute the mean...

  for(j = -BLOCK_SIZE/2; j <= BLOCK_SIZE/2; j++)
    for(i = -BLOCK_SIZE/2; i <= BLOCK_SIZE/2; i++)
      m_local_attributes[0] += double( m_PtrImg[x+i+( (y+j) * m_width )] );

  m_local_attributes[0] /= double( (BLOCK_SIZE+1)*(BLOCK_SIZE+1) );

  //-----end mean computation------

  Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
//Compute the variance...

for(j = -BLOCK_SIZE/2; j <= BLOCK_SIZE/2; j++)
for(i = -BLOCK_SIZE/2; i <= BLOCK_SIZE/2; i++)
    m_local_attributes[1] += double(( double(m_PtrImg
        [x+i+( (y+j) * m_width )]) -
        m_local_attributes[0] ) *
    ( double(m_PtrImg[x+i+( (y+j) *
        m_width )]) - m_local_attributes[0] ) );

m_local_attributes[1] /= double((BLOCK_SIZE+1)*(BLOCK_SIZE+1));

//-----end the variance computation--------

//Compute alpha and beta...

m_local_attributes[3] = m_local_attributes[1] / m_local_attributes[0];
m_local_attributes[2] = (m_local_attributes[0]*
m_local_attributes[0]) / m_local_attributes[1];

//-----end alpha, beta computation---------

//Compute the normalized lag 1...

for(j = -BLOCK_SIZE/2; j <= BLOCK_SIZE/2; j++)
for(i = -BLOCK_SIZE/2; i < BLOCK_SIZE/2; i++)
    m_local_attributes[4] += double(( double
        (m_PtrImg[x+i+( (y+j) * m_width )]) -
        m_local_attributes[0] ) * ( double
        (m_PtrImg[x+i+1+( (y+j) * m_width )]) -
        m_local_attributes[0] ) );

m_local_attributes[4] /= double(m_local_attributes[1]*
    double(BLOCK_SIZE*(BLOCK_SIZE-1)));

//-----end computation of normalized lag1---------

double lag2, lag_s2;

lag2 = lag_s2 = 0.0;
for(j = -BLOCK_SIZE/2; j < BLOCK_SIZE/2 - 1; j++)
for(i = -BLOCK_SIZE/2; i <= BLOCK_SIZE/2; i++)
    lag2 += double( ( double(m_PtrImg[x+i+( (y+j) * m_width )])
                   - m_local_attributes[0] ) * ( double(m_PtrImg[x+i+( (y+j+2) * m_width )]) - m_local_attributes[0] ) );
lag2 /= double(m_local_attributes[1]*BLOCK_SIZE*(BLOCK_SIZE-2));

for(j = -BLOCK_SIZE/2; j < BLOCK_SIZE/2; j++)
for(i = -BLOCK_SIZE/2; i < BLOCK_SIZE/2; i++)
    lag_s2 += double( ( double(m_PtrImg[x+i+((y+j) * m_width)]) - m_local_attributes[0] ) * ( double(m_PtrImg[x+i+( (y+j+1) * m_width )]) - m_local_attributes[0] ) );
lag_s2 /= double(m_local_attributes[1]*(BLOCK_SIZE-1)*
                (BLOCK_SIZE-1));

                      0.25*lag2 + 0.5*lag_s2;
}

void AnalysisAndDetectionClass::CancerDetection()
{
    unsigned int x, y;
    double a[NUMBER_OF_ATTRIBUTES];
    double a2[NUMBER_OF_ATTRIBUTES];
    double dst_c, dst_nc, dst_p;
    memset(m_CancerImg, 0, m_width*m_height*sizeof(int));
    int i,j;
    double m_InvVCM_C_tmp[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
    double m_InvVCM_NC_tmp[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
    double m_InvVCM_P_tmp[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
    int mc = 3;

    for(y = 0; y < m_height; y++)
        for(x = 0; x < m_width; x++)
            if(m_BoarderArray[x+y*m_width])
{ 
    Analyze(x, y);
    
    // first make a copy of the variance/covariance matrices.
    for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
        for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
            {
                m_InvVCM_C_tmp[i][j] = m_InvVCM_C[i][j];
                m_InvVCM_NC_tmp[i][j] = m_InvVCM_NC[i][j];
                m_InvVCM_P_tmp[i][j] = m_InvVCM_P[i][j];
            }
    
    // compute Mahalanobis distances...
    for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
        a[i] = m_local_attributes[i] - m_CentroidC[i];
    for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
        a2[j] = 0.0;
    for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
        for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
            a2[j] += (a[i] * m_InvVCM_C_tmp[i][j]);
    dst_c = 0.0;
    for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
        dst_c += (a2[i] * a[i]);
    for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
        a[i] = m_local_attributes[i] - m_CentroidNC[i];
    for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
        a2[j] = 0.0;
    for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
        for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
            a2[j] += (a[i] * m_InvVCM_NC_tmp[i][j]);
    dst_nc = 0.0;
    for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
        dst_nc += (a2[i] * a[i]);
}
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
a[i] = m_local_attributes[i] - m_CentroidP[i];

for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
a2[j] = 0.0;

for(j=0; j<NUMBER_OF_ATTRIBUTES; j++)
for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
a2[j] += (a[i] * m_InvVCM_P_tmp[i][j]);

dst_p = 0.0;

for(i=0; i<NUMBER_OF_ATTRIBUTES; i++)
dst_p += (a2[i] * a[i]);

//Classify pixel as possible cancerous.
if(fabs(dst_nc) < fabs(dst_c))
    m_CancerImg[x+y*m_width] = 0;
else
    m_CancerImg[x+y*m_width] = 1;

int AnalysisAndDetectionClass::OpenNewLearningSessionCancer(CString filename)
{" 
Function starts a new learning session for cancer and stores results in
file "filename".

in: filename.

out: I/O output, m_LSC.

return values: -2 (learning session in progress).
               -1 (error opening output file).
               0 (file output successful).

*/
{
    if(m_pf_LSC != NULL)
return -2; //already opened!

m_LSC = filename; //Store the location of the data file (needed
 //for later computations...).

m_pf_LSC = fopen(m_LSC, "wb");

if( m_pf_LSC == NULL )
    return -1; //something went wrong during opening...
else
    return 0; //everything went OK.

int AnalysisAndDetectionClass::OpenNewLearningSessionNonCancer(CString filename)
{ /*
   Function starts a new learning session for non cancer areas and stores results in file
   "filename".

   in: filename.

   out: I/O output, m_LSNC.

   return values: -2 (learning session in progress).
                  -1 (error opening output file).
                    0 (file output successful).
   */
{
    if(m_pf_LSNC != NULL)
        return -2; //already opened!

    m_LSNC = filename; //Store the location of the data file (needed
                    //for later computations...).

    m_pf_LSNC = fopen(m_LSNC, "wb");

    if( m_pf_LSNC == NULL )
        return -1; //something went wrong during opening...
    else
        return 0; //everything went OK.
}
int AnalysisAndDetectionClass::OpenNewLearningSessionPeriphary(CString filename)
{" /*
Function starts a new learning session for periphery and stores results in file "filename".

in: filename.

out: I/O output, m_LSP.

return values: -2 (learning session in progress).
-1 (error opening output file).
0 (file output successful).
*/
{
    if(m_pf_LSP != NULL)
        return -2; //already opened!

    m_LSP = filename; //Store the location of the data file (needed
                     //for later computations...).

    m_pf_LSP = fopen(m_LSP, "wb");

    if( m_pf_LSP == NULL )
        return -1; //something went wrong during opening...
    else
        return 0; //everything went OK.
}

void AnalysisAndDetectionClass::CloseLearningSessionCancer()
{" /*
Closes the result file opened by OpenNewLearningSessionCancer() function and computes the inverted covariance matrix.
*/
{
    fclose(m_pf_LSC);
    m_pf_LSC = NULL;

    ComputeVCM(m_LSC, m_InvVCM_C, m_CentroidC);
}
void AnalysisAndDetectionClass::CloseLearningSessionNonCancer()
/*
 * Closes the result file opened by OpenNewLearningSessionCancer() function
 * and computes the inverted covariance matrix.
 */
{
    fclose(m_pf_LSNC);
    m_pf_LSNC = NULL;

    ComputeVCM(m_LSNC, m_InvVCM_NC, m_CentroidNC);
}

void AnalysisAndDetectionClass::CloseLearningSessionPeriphary()
/*
 * Closes the result file opened by
 * OpenNewLearningSessionPeriphary() function
 * and computes the inverted covariance matrix.
 */
{
    fclose(m_pf_LSP);
    m_pf_LSP = NULL;

    ComputeVCM(m_LSP, m_InvVCM_P, m_CentroidP);
}

void AnalysisAndDetectionClass::ComputeVCM(CString filename, double
VCM[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES], double
mean[NUMBER_OF_ATTRIBUTES])
/*
 * it finds the covariance matrix and it builds the inverted
 * covariance matrix...
 *
 * input:
 * filename to the learned data...
 *
 * output:
 * 7x7 inverted covariance matrix.
 */
/*

FILE *tmp_fp;

int i;

tmp_fp = fopen(filename, "rb");

// Attributes_Structure mean; //this structure will hold the centroids for each attribute...

//First lets go ahead and compute the centroid (or mean) of each attribute...

int count = 1;

for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
    mean[i] = 0.0;

while(!feof(tmp_fp))
{
    fread(&m_local_attributes,
         sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, tmp_fp);

    if(!feof(tmp_fp))
    {
        for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
        {
            mean[i] += m_local_attributes[i];
        }
        count++;
    }
}

for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
    mean[i] /= double(count);

//----------end finding mean----------

rewind(tmp_fp);

//lets go ahead and compute the variance of the attributes...
//read in the attributes from the file and compute the variance...
double variance[NUMBER_OF_ATTRIBUTES]; //this array will hold
  //the variances of the
  //attributes...
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
  variance[i] = 0.0;
while(!feof(tmp_fp))
{
  fread(&m_local_attributes,
        sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, tmp_fp);
  if(!feof(tmp_fp))
  {
    for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
      variance[i] += ( (m_local_attributes[i]-
                      mean[i])*(m_local_attributes[i]-mean[i]) );
  }
}
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
  variance[i] /= double(count);

//----------end finding variance----------

rewind(tmp_fp);

//lets go ahead and compute the co-variance of the attributes...

double std[NUMBER_OF_ATTRIBUTES]; //this will hold the standard
  deviation of the attributes...
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
  std[i] = sqrt(variance[i]);

double
covariance[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
  //The covariance matrix will be stored in a
  //NUMBER_OF_ATTRIBUTES(7)-element array in the following way:
  // m, mv, mAlpha, mBeta, mR1... -> covariance[0][0], c[0][1],
  //c[0][2], ...
  //...
int j;

for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
for(j = 0; j<NUMBER_OF_ATTRIBUTES; j++)
covariance[i][j] = 0.0;

//first copy the variances of each attribute in the offdiagonal...
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
covariance[i][i] = variance[i];

while(!feof(tmp_fp))
{
    fread(&m_local_attributes,
sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, tmp_fp);
    if(!feof(tmp_fp))
    {
        //compute covariance matrix... (when j = 0 then mean*x, when j=1
then variance*x, ...)
        for(j = 0; j<NUMBER_OF_ATTRIBUTES-1; j++)
for(i = j+1; i<NUMBER_OF_ATTRIBUTES; i++)
covariance[j][i] += ( (m_local_attributes[j]-
mean[j])*(m_local_attributes[i]-mean[i]) );
    }
}

for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
for(j = 0; j<NUMBER_OF_ATTRIBUTES; j++)
covariance[i][j] /= double(count);

//----------end finding covariance----------

//Go ahead and put the results into a variance/covariance matrix...
for(int k = 0; k<NUMBER_OF_ATTRIBUTES; k++)
{
    for(i = k; i<NUMBER_OF_ATTRIBUTES; i++)
    {
        VCM[k][i] = covariance[k][i];
    }
}
for(j = 0; j<k; j++)
    VCM[k][j] = VCM[j][k];

//-------end variance/covariance matrix construction-------

//Now call function to compute the inverse of the VCM.
if( MatrixInversion(VCM) == -1)
    AfxMessageBox("Error! Covariance matrix is singular!");

CString cs1, cs2;
for(i = 0; i<NUMBER_OF_ATTRIBUTES; i++)
    for(j = 0; j<NUMBER_OF_ATTRIBUTES; j++)
    {
        cs1.Format("%f ", VCM[i][j]);
        cs2 += cs1;
    }
    cs2 += 'n';
AfxMessageBox(cs2);

cs2.Empty();
for(j = 0; j<NUMBER_OF_ATTRIBUTES; j++)
    {
        cs1.Format("%f ", mean[j]);
        cs2 += cs1;
    }
AfxMessageBox(cs2);

cs2.Empty();
for(j = 0; j<NUMBER_OF_ATTRIBUTES; j++)
    {
        cs1.Format("%f ", variance[j]);
        cs2 += cs1;
    }
AfxMessageBox(cs2);
fclose(tmp_fp);
void AnalysisAndDetectionClass::LearnCancer(bool *selected_area)
{
    /*
    The function will compute the attributes of the selected_area (which is
    supposently cancerous)
    and write the attribute values to the file.
    */
    unsigned int x, y;
    for(y = 0; y < m_height; y++)
        for(x = 0; x < m_width; x++)
            if(selected_area[x+y*m_width])
                Analyze(x, y);
        fwrite(m_local_attributes,
              sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, m_pf_LSC);
}

void AnalysisAndDetectionClass::LearnNonCancer(bool *selected_area)
{
    /*
    The function will compute the attributes of the selected_area (which is
    supposently non cancerous)
    and write the attribute values to the file.
    */
    unsigned int x, y;
    for(y = 0; y < m_height; y++)
        for(x = 0; x < m_width; x++)
            if(selected_area[x+y*m_width])
                Analyze(x, y);
        fwrite(m_local_attributes,
              sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, m_pf_LSNC);
void AnalysisAndDetectionClass::LearnPeriphary(bool *selected_area)

/*
The function will compute the attributes of the selected_area (which is
supposently periphary)
and write the attribute values to the file.
*/
{
    unsigned int x, y;
    for(y = 0; y < m_height; y++)
        for(x = 0; x < m_width; x++)
            if(selected_area[x+y*m_width])
            {
                Analyze(x, y);
                fwrite(m_local_attributes, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, m_pf_LSP);
            }
}

int AnalysisAndDetectionClass::SaveLearnedSession(CString filename)

/*
saves the learning session so that it can be loaded next time...
(so that we wouldn't have to "teach" the computer every time...)

returns -1 if unable to open the file...
else
    returns 0 and it saves all the attributes and inverse covariance matrices (to
    be used in another session...)
*/
{
    FILE *pf = fopen(filename, "wb");
    if(pf == NULL)
        return -1;

    //go ahead and write the centroids for cancer, non-cancer and
    //periphary...
    fwrite(m_CentroidC, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);
fwrite(m_CentroidNC, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);
fwrite(m_CentroidP, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);

//go ahead and write the inverse covariance matrices for cancer, //non-cancer and periphery...
fwrite(m_InvVCM_C,
sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
1, pf);
fwrite(m_InvVCM_NC,
sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
1, pf);
fwrite(m_InvVCM_P,
sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
1, pf);
fclose(pf);
return 0;

int AnalysisAndDetectionClass::LoadPreviousSession(CString filename)
/*
loads a previously saved session to be used in the cancer detection...
returns -1 if unable to open the file...
else
returns 0 and it loads all the saved attributes and inverse covariance matrices...
*/
{
FILE *pf = fopen(filename, "rb");
if(pf == NULL)
    return -1;
    //go ahead and read the centroids for cancer, non-cancer and //periphery...
fread(m_CentroidC, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);
fread(m_CentroidNC, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);
fread(m_CentroidP, sizeof(double[NUMBER_OF_ATTRIBUTES]), 1, pf);
//go ahead and read the inverse covariance matrices for cancer,    //non-cancer
and periphery...
    fread(m_InvVCM_C,
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        sizeof(double[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES]),
        p);
    fclose(p);
    return 0;
}

int AnalysisAndDetectionClass::MatrixInversion(double
    VCMI[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES])
    /*
    inverts a matrix of size NUMBER_OF_ATTRIBUTES x NUMBER_OF_ATTRIBUTES.
    returns -1 if matrix is singular...
    */
    {
        int i,j,ipass,imx,icol,irow;
        double det,temp,pivot,factor;
        double a[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES],
            ainv[NUMBER_OF_ATTRIBUTES][NUMBER_OF_ATTRIBUTES];
        for (i=0;i<NUMBER_OF_ATTRIBUTES;i++)
            for (j=0;j<NUMBER_OF_ATTRIBUTES;j++)
                a[i][j] = VCMI[i][j];

        // INITIALLY STORE THE IDENTITY MATRIX IN AINV AFTER THE LAST PASS
        // THIS
        // WILL BE REPLACED BY THE INVERSE OF A
        for (i=0;i<NUMBER_OF_ATTRIBUTES;i++)
            for (j=0;j<NUMBER_OF_ATTRIBUTES;j++)
                ainv[i][j] = 1.0 / a[i][j];
\begin{verbatim}
{
    for(j=0;j<NUMBER_OF_ATTRIBUTES;j++)
    {
        ainv[i][j]=0;
        if(i==j) ainv[i][i]=1;
    }
}

// COMPUTATION //
// THE CURRENT PIVOT ROW IS IPASS, FOR EACH PASS, FIRST FIND THE
// MAXIMUM
// ELEMENT IN THE PIVOT COLUMN
for (ipass=0;ipass<NUMBER_OF_ATTRIBUTES;ipass++)
{
    imx=ipass;
    for(irow=ipass;irow<NUMBER_OF_ATTRIBUTES;irow++)
    {
        if(fabs(a[irow][ipass])>fabs(a[imx][ipass])) imx=irow;
    }
    // INTERCHANGE THE ELEMENTS OF ROW IPASS AND ROW IMX IN BOTH A
    // AND AINV
    if(imx!=ipass)
    {
        for(icol=0;icol<NUMBER_OF_ATTRIBUTES;icol++)
        {
            temp=ainv[ipass][icol];
            ainv[ipass][icol]=ainv[imx][icol];
            ainv[imx][icol]=temp;
            if(icol >= ipass)
            {
                temp=a[ipass][icol];
                a[ipass][icol]=a[imx][icol];
                a[imx][icol]=temp;
            }
        }
    }
}

// THE DETRMINANT IS THE PRODUCT OF THE PIVOT ELEMENTS
pivot=a[ipass][ipass];
det = 1.0;
det=det*pivot;

if (det==0)
    return -1; //Singular matrix returns error!

for(icol=0;icol<NUMBER_OF_ATTRIBUTES;icol++)
\end{verbatim}
\{
//NORMALIZING PIVOT ROW BY DIVIDING ACCROSS BY
//THE PIVOT ELEMENT/
ainv[ipass][icol]=ainv[ipass][icol]/pivot;
if (icol>=ipass) a[ipass][icol]=a[ipass][icol]/pivot;
\}
for(irow=0;irow<NUMBER_OF_ATTRIBUTES;irow++)
// NOW REPLACE EACH ROW BY THE ROW PLUS A MULTIPLE OF THE PIVOT
// ROW WITH A FACTOR CHOSEN SO THAT THE ELEMNT OF A ON THE
// PIVOT COLUMN IS 0
\{
if(irow!=ipass) factor=a[irow][ipass];
for(icol=0;icol<NUMBER_OF_ATTRIBUTES;icol++)
\{
   if(irow!=ipass)
   \{
      ainv[irow][icol]=ainv[irow][icol]-factor*ainv[ipass][icol];
      a[irow][icol]=a[irow][icol]-factor*a[ipass][icol];
   \}
\}
\}

for (i=0;i<NUMBER_OF_ATTRIBUTES;i++)
for(j=0;j<NUMBER_OF_ATTRIBUTES;j++)
   VCM[i][j] = ainv[i][j];
return 0;
\}

void FindAngle(float *theta, int x0, int y0, int x1, int y1, int x2, int y2)
\{
   (*theta) = acos( double( (x0-x1)*(x2-x1)+(y0-y1)*(y2-y1)) )/
              ( sqrt( double( (x0-x1)*(x0-x1) + (y0-y1)*(y0-y1) ) ) *
                 sqrt( double( (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) ) ) );
\}

bool IsLeftTurn(int x0, int y0, int x1, int y1, int x2, int y2)
\{
   if(x0*(y1-y2)-y0*(x1-x2)+(x1*y2-y1*x2) > 0)
      return true;
   return false;
\}
```cpp
bool IsRightTurn(int x0, int y0, int x1, int y1, int x2, int y2) {
    if(x0*(y1-y2)-y0*(x1-x2)+(x1*y2-y1*x2) < 0)
        return true;
    return false;
}

int AnalysisAndDetectionClass::TraceBorder(int *PtrImgTmp, unsigned int width, unsigned int height, CPoint CurrLoc, int OldDirection) {
    /*
    Input is the bitmap image with the current location of the point on the border of the object and the direction of which we found this point. Now determine the next point on the border by scanning clockwise on the binary image (and to the pixel next to the direction found last time so that we don't find the last pixel again...). i.e.: If the direction is 3
    (3) 2 1
     \ / /  
     4 - X - 0
     / | \ 
       5  6  7
    then we need to start the clockwise direction search at location (x,y+1) (or the direction of 2).
    */
    
    int i, j;
    int NewDirection;
    bool exit = false;

    int Ptrlmg[21][21];
    for(j = 0; j<21; j++)
        for(i = 0; i<21; i++)
            Ptrlmg[j][i] = Ptrlmg[i+j*21];

    switch(OldDirection) {
        case 0: OldDirection = 4;
                break;
    }
}
```
break;
case 1: OldDirection = 5;
    break;
case 2: OldDirection = 6;
    break;
case 3: OldDirection = 7;
    break;
case 4: OldDirection = 0;
    break;
case 5: OldDirection = 1;
    break;
case 6: OldDirection = 2;
    break;
case 7: OldDirection = 3;
    break;
default: OldDirection = 9;
}

if(OldDirection < 9 && OldDirection > 0)
    OldDirection--;
else
    if(OldDirection == 0)
        OldDirection = 7;

while(!exit)
{
    //Initially computes the starting position for the clockwise  //search...
    //after that it acts as a translator for Direction to pixel    //location.
    switch(OldDirection)
    {
        case 0: i = 1;
            j = 0;
            break;
        case 1: i = 1;
            j = 1;
            break;
        case 2: i = 0;
            j = 1;
            break;
        case 3: i = -1;
            j = 1;
            break;
        case 4: i = -1;
            break;
        // Other cases...
    }

    // Further code...

    // Additional code...
}

// End of the code...

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
j = 0;
break;
case 5: i = -1;
j = -1;
break;
case 6: i = 0;
j = -1;
break;
case 7: i = 1;
j = -1;
break;
default: i = 0;
j = i;
}

if(PtrImg[CurrLoc.x+i][CurrLoc.y+j] > 0)
{
    if(i == 1 && j == 0)
        NewDirection = 0;
    else
        if(i == 1 && j == 1)
            NewDirection = 1;
        else
            if(i == 0 && j == 1)
                NewDirection = 2;
            else
                if(i == -1 && j == 1)
                    NewDirection = 3;
                else
                    if(i == -1 && j == 0)
                        NewDirection = 4;
                    else
                        if(i == -1 && j == -1)
                            NewDirection = 5;
                        else
                            if(i == 0 && j == -1)
                                NewDirection = 6;
                            else
                                if(i == 1 && j == -1)
                                    NewDirection = 7;
                                else
                                    NewDirection = 9;
    exit = true;
}
if(OldDirection == 9)
    OldDirection = 1;
else
    if(OldDirection == 0)
        OldDirection = 7;
    else
        OldDirection--;

return NewDirection;

double AnalysisAndDetectionClass::BendingEnergy(int *PtrImgTmp, unsigned int width, unsigned int height)
//fix, [21][21]
/*
This Function will return the Bending energy of the binary bitmap passed to it.
Note that the edges of the image must be 0. i.e.:
000000000000000000000000000000000000000000000000
OxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxO
OxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxO
OxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxO
OxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxO
000000000000000000000000000000000000000000000000
*/
{
int Ptrlmg[21][21];

int periphery = 1;
int bending_energy = 0;
CPoint StartLoc = (0,0);
CPoint CurrLoc = (0,0);
int x, y;
int OldDirection = 9; //valid values 0..8. For example:
    // 3 2 1
    // 4 x 0
    // 5 6 7

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
int NewDirection;

int i, j;

for(j = 0; j<21; j++)
for(i = 0; i<21; i++)
{
    PtrImg[j][i] = PtrImgTmp[i+j*21];

    if(PtrImg[j][i] > 0)
        PtrImg[j][i] = 50;
}

for(i = 0; i<21; i++)
{
    PtrImg[i][0] = 0;
    PtrImg[0][i] = 0;
    PtrImg[i][20] = 0;
    PtrImg[20][i] = 0;
}

/*
int ii, jj;

for(jj = -10; jj<=10; jj++)
for( ii = -10; ii<=10; ii++)
    Original_Data_Array[(Image_Width*(jj+temp_my))+ii+temp_mx] =
    int(PtrImg[ii+10][jj+10]);
*/

bool exit = false;

for(y = height-1; y >= 0 && !exit; y--)
for(x = 0; x < width && !exit; x++)
{
    if(PtrImg[x][y] > 0)
    {
        StartLoc.x = x;
        StartLoc.y = y;
        exit = true;
    }
\begin{verbatim}

CurrLoc = StartLoc;

do{
    PtrImg[CurrLoc.x][CurrLoc.y] = 255;

    NewDirection = TraceBorder(&(PtrImg[0][0]), width, height, CurrLoc, OldDirection);

    if(OldDirection != NewDirection)
        bending_energy+=1; //Penalize since not on a straight line.
    else
        bending_energy+=0; //Reward since on a straight line.

    periphery++;
    OldDirection = NewDirection;

    switch(NewDirection)
    {
        case 0: CurrLoc.x += 1;
            break;
        case 1: CurrLoc.x += 1;
               CurrLoc.y += 1;
            break;
        case 2: CurrLoc.y += 1;
            break;
        case 3: CurrLoc.y += 1;
               CurrLoc.x -= 1;
            break;
        case 4: CurrLoc.x -= 1;
            break;
        case 5: CurrLoc.x -= 1;
               CurrLoc.y -= 1;
            break;
        case 6: CurrLoc.y -= 1;
            break;
        case 7: CurrLoc.x += 1;
               CurrLoc.y -= 1;
            break;
    }
}while(CurrLoc != StartLoc);
\end{verbatim}
```c
  return double(double(bending_energy)/double(periphery));
}
class Border_Detection_Class
{
public:
    Border_Detection_Class();
    ~Border_Detection_Class();
    void DetectBorder(unsigned int width, unsigned int height, unsigned char *PtrImg, CDC *, unsigned int block_size);

    bool *m_border_array;
    unsigned int m_width;
    unsigned int m_height;

private:
    int *m_array1; //Temporary arrays used int DetectBorder().
    int *m_array2;

    unsigned int *S;

    void Edge_Detection(int *, int *, unsigned int, unsigned int, int[3][3], float);
    void eliminate_noise(int *array1, int *array2, unsigned int width, unsigned int height, int block, int max_white_pixels);
    void BinaryColor(int * array, unsigned int width, unsigned int height, int threshold);
    void Detect_HV_Lines(int *array1, unsigned char *array2, unsigned width, unsigned int height);
    void remove_from_original(int *array1, int *array2, unsigned int width, unsigned int height, int block);
    void F_H_Line_removal(int * array1, int *array2, unsigned int width, unsigned int height);
    void Fit_B_Spline(int *, unsigned int width, unsigned int height, unsigned int block_size);

    CDC *m_dcp;
};
/*
   Border_Detection_Routines.cpp
*/

#include "stdafx.h"
#include "Border_Detection_Routines.h"
#include "math.h"
#define pi acos(-1)

Border_Detection_Class::Border_Detection_Class()
{
    m_border_array = NULL;
    m_array1 = m_array2 = NULL;
    m_width = m_height = 0;
    S = new unsigned int[1000*360];
}

Border_Detection_Class::~Border_Detection_Class()
{
    if( m_border_array != NULL)
        delete [] m_border_array;

    if(m_array1 != NULL)
        delete [] m_array1;

    if(m_array2 != NULL)
        delete [] m_array2;

    delete [] S;
}

void Border_Detection_Class::DetectBorder(unsigned int width, unsigned int height, unsigned char *PtrImg, CDC *cdcp, unsigned int block_size)
{
    m_dcp = cdcp;

    if( m_border_array != NULL)
        delete [] m_border_array;

    if(m_array1 != NULL)
        delete [] m_array1;

    if(m_array2 != NULL)
        delete [] m_array2;

    m_array1 = new int[width*height];
m_array2 = new int[width*height];
m_border_array = new bool[width*height];

unsigned char *ptr_i_t = Ptrlmg;
int *a_p = m_array1;

for(unsigned int i = 0; i < width*height; i++)
{
    (*a_p) = (*ptr_i_t);
    a_p++;
    ptr_i_t++;
}

unsigned char *PtrImgMod = new unsigned char[width*height];
memcpy(PtrImgMod, Ptrlmg, width*height*sizeof(unsigned char));

m_width = width;
m_height = height;

//Detect border lines...
BinaryColor(m_array1, width, height, 60 /*intensity threshold*/);

int x, y;

int matrix_filter[3][3];
float divisor;

matrix_filter[0][0] = 0;
matrix_filter[0][1] = -1;
matrix_filter[0][2] = 0;

matrix_filter[1][0] = -1;
matrix_filter[1][1] = 4;
matrix_filter[1][2] = -1;

matrix_filter[2][0] = 0;
matrix_filter[2][1] = -1;
matrix_filter[2][2] = 0;

Edge_Detection(m_array1, m_array2, width, height, matrix_filter, 1);

//detect window within ultrasound image and copy it to m_array1.
Detect_HV_Lines(m_array1, PtrImgMod, width, height);

// Apply Laplacian filter
Edge_Detection(m_array1, m_array2, width, height, matrix_filter, 160);

// eliminate noise.
eliminate_noise(m_array1, m_array2, width, height, 30, 5);

ptr_i_t = PtrImgMod;
a_p = m_array2;

for(i = 0; i < width*height; i++)
{
    (*a_p) = (*ptr_i_t);
a_p++;
    ptr_i_t--;
}

//Remove characters from original (Block size 30).
remove_from_original(m_array1, m_array2, width, height, 30);

BinaryColor(m_array2, width, height, 10);

//Laplacian edge detection.
Edge_Detection(m_array2, m_array1, width, height, matrix_filter, 1);

// eliminate noise.
    eliminate_noise(m_array2, m_array1, width, height, 30, 28);

//Outer border isolation...
F_H_Line_removal(m_array2, m_array1, width, height);

//eliminate noise.
    eliminate_noise(m_array1, m_array2, width, height, 30, 28);

//eliminate noise.
    eliminate_noise(m_array1, m_array2, width, height, 4, 3);

//eliminate noise.
    eliminate_noise(m_array1, m_array2, width, height, 4, 3);

memset(m_border_array, false, width*height*sizeof(bool));

Fit_B_Spline(m_array1, width, height, block_size);

delete [] PtrImgMod;
void Border_Detection_Class::BinaryColor(int *array, unsigned int width, unsigned int height, int threshold /*intensity threshold*/)  
/*  
This function will take the array and quantize the colors to either 0 or 1 if above threshold.  
*/
{
    for(unsigned int i = 0; i < width*height; i++)
    {
        if (*array) >= threshold
            (*array) = 1;
        else
            (*array) = 0;

        array++;
    }
}

void Border_Detection_Class::Edge_Detection(int *array, int *array2, unsigned int width, unsigned int height, int matrix_filter[3][3], float divisor, float filter_threshold)  
{
    int temp_x2, temp_y2;
    float filter_result;

    memset(array2, 0, width*height*sizeof(int));

    for(temp_y2 = 1; temp_y2 < (int)height-1; temp_y2++)
        for(temp_x2 = 1; temp_x2 < (int)width-1 ; temp_x2++)
        {
            filter_result = 0;

            filter_result += matrix_filter[-1+1][-1+1] *  
                float(array[(width*(temp_y2+1))+(temp_x2-1)]);
            filter_result += matrix_filter[0+1][-1+1] *  
                float(array[(width*(temp_y2))+(temp_x2-1)]);
        }
    }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
filter_result += matrix_filter[1+1][-1+1] * 
float(array[(width*(temp_y2-1))+(temp_x2-1))];

filter_result += matrix_filter[-1+1][0+1] * 
float(array[(width*(temp_y2+1))+(temp_x2+0)]);

filter_result += matrix_filter[0+1][0+1] * 
float(array[(width*(temp_y2))+(temp_x2+0)]);

filter_result += matrix_filter[1+1][-1+1] * 
float(array[(width*(temp_y2+1))+(temp_x2-1)]);

if(filter_result >= filter_threshold)
array2[(width*temp_y2)+temp_x2] = 1;
else
array2[(width*temp_y2)+temp_x2] = 0;

memcpy(array, array2, width*height*sizeof(int));

void Border_Detection_Class::Detect_HV_Lines(int *array, unsigned char *array2,
unsigned width, unsigned int height)
{
int temp_x2, temp_y2;
int c1, c2;
float d_to_r = float(pi / 180.0f);

int final_theta1, final_S1, final_theta2, final_S2, final_theta3, final_S3,
final_theta4, final_S4;

int image_rect_x1 = 0;
int image_rect_x2 = 639;
int image_rect_y1 = 0;
int image_rect_y2 = 479;
for(cl = 0; cl<360; cl++)
for(c2 = 0; c2 < 1000; c2++)
    S[c2*360+c1] = 0;

for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
if( array[(width*temp_y2)+temp_x2] > 0)
    for(cl = 0; cl < 360; cl++)
    {
        if(int(float(float(float(temp_y2)
            float(sin( cl*d_to_r ))) +
            float( float(temp_x2) *
            float(cos( cl*d_to_r ))))) < 0 )
            break;
    }

    //Build the Hough transform table.
    S[ int(float( float(float(temp_y2) * float(sin( c1*d_to_r ))) + float( float(temp_x2) *
    float(cos( c1*d_to_r )))))) *360 + cl]++;

unsigned int temp=0;

int done_with_theta90 = 0, done_with_theta0 = 0;

for(c1 = 0; c1<360; c1++)
for(c2 = 0; c2 < 1000; c2++)
    if(S[c2*360+c1] > temp)
    {
        temp = S[c2*360+c1];
        if( !(c1 == 90 || c1 == 0) )
            temp=0;
        final_theta1 = c1;
        final_S1 = c2;
    }

if(final_theta1 == 90)
    done_with_theta90++;
else
    done_with_theta0++;

for(c1 = -20; c1 <= 20; c1++)
for(c2 = -10; c2<10; c2++)
if( (final_S1+c1 >= 0) && (final_S1+c1 < 1000) &&
    (final_theta1+c2 >= 0) && (final_theta1+c2 < 360) )
    S[(final_S1+c1)*360+final_theta1+c2] = 0;

temp = 0;

for(c1 = 0; c1<360; c1++)
for(c2 = 0; c2 < 1000; c2++)
if(S[c2*360+c1] > temp)
   {
      temp = S[c2*360+c1];

      if( !(c1 == 90 || c1 == 0 )
         temp=0;

      final_theta2 = c1;
      final_S2 = c2;
   }

if(final_theta2 == 90)
   done_with_theta90++;
else
   done_with_theta0++;

for(c1 = -20; c1<= 20; c1++)
   for(c2 = -10; c2<10; c2++)
      if( (final_S2+c1 >= 0) && (final_S2+c1 < 1000) &&
         (final_theta2+c2 >= 0) && (final_theta2+c2 < 360) )
         S[(final_S2+c1)*360+final_theta2+c2] = 0;

   temp = 0;

for(c1 = 0; c1<360; c1++)
for(c2 = 0; c2 < 1000; c2++)
if(S[c2*360+c1] > temp)
   {
      temp = S[c2*360+c1];

      if( !(c1 == 90 || c1 == 0 )
         temp=0;

      final_theta3 = c1;
      final_S3 = c2;
if(done_with_theta0 == 2 && final_theta3 == 0)
    S[c2*360+c1] = temp = 0;
if(done_with_theta90 == 2 && final_theta3 == 90)
    S[c2*360+c1] = temp = 0;
}
if(final_theta3 == 90)
    done_with_theta90++;
else
    done_with_theta0++;
for(c1 = -20; c1<= 20; c1++)
    for(c2 = -10; c2<10; c2++)
        if( (final_S3+c1 >=0) && (final_S3+c1 < 1000) &&
            (final_theta3+c2 >= 0) && (final_theta3+c2 < 360) )
            S[(final_S3+c1)*360+final_theta3+c2] = 0;
    temp = 0;
for(c1 = 0; c1<360; c1++)
    for(c2 = 0; c2 < 1000; c2++)
        if(S[c2*360+c1] > temp)
        {
            temp = S[c2*360+c1];
            if( !(c1 == 90 || c1 == 0 )
                temp=0;
            final_theta4 = c1;
            final_S4 = c2;
            if(done_with_theta0 == 2 && final_theta4 == 0)
                S[c2*360+c1] = temp = 0;
            if(done_with_theta90 == 2 && final_theta4 == 90)
                S[c2*360+c1] = temp = 0;
        }
    if(final_theta4 == 90)
        done_with_theta90++;
    else
        done_with_theta0++;
if(final_theta1 == 90)
    image_rect_y1 = int(final_S1/sin(pi/2));
else
    if(final_theta2 == 90)
image_rect_y1 = int(final_S2/sin(pi/2));

if(final_theta3 == 90)
    image_rect_y1 = int(final_S3/sin(pi/2));
else
    if(final_theta4 == 90)
        image_rect_y1 = int(final_S4/sin(pi/2));

if(final_theta2 == 90 && image_rect_y1 != int(final_S2/sin(pi/2)))
    image_rect_y2 = int(final_S2/sin(pi/2));
else
    if(final_theta3 == 90 && image_rect_y1 != int(final_S3/sin(pi/2)))
        image_rect_y2 = int(final_S3/sin(pi/2));
else
    if(final_theta4 == 90)
        image_rect_y2 = int(final_S4/sin(pi/2));

if(final_theta1 == 0)
    image_rect_x1 = final_S1;
else
    if(final_theta2 == 0)
        image_rect_x1 = final_S2;
else
    if(final_theta3 == 0)
        image_rect_x1 = final_S3;
else
    if(final_theta4 == 0)
        image_rect_x1 = final_S4;

if(final_theta2 == 0 && image_rect_x1 != final_S2)
    image_rect_x2 = final_S2;
else
    if(final_theta3 == 0 && image_rect_x1 != final_S3)
        image_rect_x2 = final_S3;
else
    if(final_theta4 == 0)
        image_rect_x2 = final_S4;

if(image_rect_y1 > image_rect_y2)
{
    temp = image_rect_y1;
    image_rect_y1 = image_rect_y2;
}
image_rect_y2 = temp;
}

if(image_rect_x1 > image_rect_x2)
{
    temp = image_rect_x1;
    image_rect_x1 = image_rect_x2;
    image_rect_x2 = temp;
}

image_rect_y1+=5;
image_rect_y2-=5;
image_rect_x1+=5;
image_rect_x2-=5;

for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
    if(temp_y2>image_rect_y1 && temp_y2<image_rect_y2 &&
       temp_x2>image_rect_x1 && temp_x2<image_rect_x2)
        array[(width*temp_y2)+temp_x2] =
        array2[(width*temp_y2)+temp_x2];
    else
        array[(width*temp_y2)+temp_x2] =
        array2[(width*temp_y2)+temp_x2] = 0;
}

void Border_Detection_Class::eliminate_noise(int *array1, int *array2, unsigned int width, unsigned int height, int block, int max_white_pixels)
{
    int temp_y2, temp_x2;
    int x,y;
    int frequency;
    memcpy(array2, array1, width*height*sizeof(int));

    for(temp_y2 = block/2; temp_y2 < (int)height-(block/2); temp_y2++)
    for(temp_x2 = block/2; temp_x2 < (int)width-(block/2); temp_x2++)
    {
        if(array2[(width*temp_y2)+temp_x2] > 0)
        {
            Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
frequency = 0;

for(y = -(block/2); y<(block/2); y++)
for(x = -(block/2); x<(block/2); x++)
if( array2[(width*(temp_y2+y))+temp_x2+x] > 0)
    frequency++;

if(frequency < max_white_pixels)
    for(y = -(block/2); y<(block/2); y++)
        for(x = -(block/2); x<(block/2); x++)
            array1[(width*(temp_y2+y))+temp_x2+x] = 0;

void Border_Detection_Class::remove_from_original(int *array1, int *array2, unsigned int width, unsigned int height, int block)
{

int temp_y2, temp_x2;
int x,y;

for(temp_y2 = block/2; temp_y2 < (int)height-(block/2); temp_y2++)
for(temp_x2 = block/2; temp_x2 < (int)width-(block/2); temp_x2++)
{
    if( array1[(width*temp_y2)+temp_x2] > 0)
        for(y = -(block/2); y<(block/2); y++)
            for(x = -(block/2); x<(block/2); x++)
                array2[(width*(temp_y2+y))+temp_x2+x] = 0;

}

}

void Border_Detection_Class::F_H_Line_removal(int *array1, int *array2, unsigned int width, unsigned int height)
{

int temp_x2, temp_y2;

//Now it is time to isolate only the outer border of our image...:
memset(array2, 0, width*height*sizeof(int));

//first from the left...:
for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
    if(array1[(width*temp_y2)+temp_x2]>0)
    {
        array2[(width*temp_y2)+temp_x2] = 1;
        temp_x2 = width; //Bail out...
    }

//Now from the Right...
for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
for(temp_x2 = (int)width; temp_x2 > 0 ; temp_x2--)
    if(array1[(width*temp_y2)+temp_x2]>0)
    {
        array2[(width*temp_y2)+temp_x2] = 1;
        temp_x2 = 0; //Bail out...
    }

//Now from the bottom...:
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
    if(array1[(width*temp_y2)+temp_x2]>0)
    {
        array2[(width*temp_y2)+temp_x2] = 1;
        temp_y2 = height; //Bail out...
    }

//Now from the top...
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
for(temp_y2 = (int)height-1; temp_y2 >= 0; temp_y2--)
    if(array1[(width*temp_y2)+temp_x2]>0)
    {
        array2[(width*temp_y2)+temp_x2] = 1;
        temp_y2 = 0; //Bail out...
    }
void Border_Detection_Class::Fit_B_Spline(int *array1, unsigned int width, unsigned int height, unsigned int block_size)
{

    CPoint *ptr;

    int temp_x2, temp_y2;

    unsigned int total_point_count = 1;
    unsigned int total_point_count_corrected;

    for(unsigned int i = 0; i<width*height; i++)
        if(array1[i] > 0)
            total_point_count++;

    ptr = new CPoint[total_point_count];

    unsigned int hc, wc;
    int k;
    int min_loc;
    int min_dst = 10000;
    int dst;
    CPoint temp_cp;

    i = 1;

    for(hc = 0; hc<height; hc++)
        for(wc = 0; wc<width; wc++)
            if(array1[wc+hc*width] > 0)
            {
                ptr[i].x = wc;
                ptr[i].y = hc;
                if(ptr[i].y < min_dst)
                {
                    min_dst = ptr[i].y;
                }
                min_loc = i;
            }
            i++;

        ptr[0] = ptr[min_loc];
CPoint center_pixel;

//Find a point with min x value...
min_dst = 100000;
for(i = 0; i<total_point_count; i++)
    if( min_dst > ptr[i].x)
        min_dst = ptr[i].x;
center_pixel.x = min_dst;

//Find a point with max x value...
min_dst = -1;
for(i = 0; i<total_point_count; i++)
    if( min_dst < ptr[i].x)
        min_dst = ptr[i].x;
center_pixel.x += min_dst;
center_pixel.x /= 2;

int total_a_c = 0;

//Find a point with min y value...
min_dst = 100000;
center_pixel.y = 0;
for(i = 0; i<total_point_count; i++)
    if( ptr[i].x > center_pixel.x-40 && ptr[i].x < center_pixel.x+40)
        {for(k = 0; k<total_point_count; k++)
            if(k != i && ptr[k].x == ptr[i].x)
                {
                center_pixel.y += ((ptr[i].y+ptr[k].y)/2);
                total_a_c++;
                }
        }
center_pixel.y /= total_a_c;

i = 1;
for(k = i; k<total_point_count; k++)
{
    dst = (ptr[i-1].x - ptr[k].x)*(ptr[i-1].x -
    ptr[k].x) + (ptr[i-1].y - ptr[k].y)*
    (ptr[i-1].y - ptr[k].y);
    if( dst < min_dst && dst != 0)
    {
        min_dst = dst;
        min_loc = k;
    }
}

temp_cp = ptr[i];
ptr[i] = ptr[min_loc];
ptr[min_loc] = temp_cp;

int direction;
int pixel_direction;

int total_count_final = 1;
int old_min_loc = -10;
bool exit = false;

direction = center_pixel.x*(ptr[0].y-ptr[1].y)-
center_pixel.y*(ptr[0].x-
ptr[1].x)+ptr[0].x*ptr[1].y-
ptr[0].y*ptr[1].x;

for(i = 1; i<total_point_count && !exit; i++)
{
    min_dst = 100000;
    for(k = i; k<total_point_count; k++)
    {
        dst = (ptr[i-1].x - ptr[k].x)*(ptr[i-1].x - ptr[k].x)
        + (ptr[i-1].y - ptr[k].y)*(ptr[i-1].y - ptr[k].y);
        pixel_direction = center_pixel.x*(ptr[i-1].y-
        ptr[k].y)-center_pixel.y*(ptr[i-1].x.ptr[k].x)+
        ptr[i-1].x*ptr[k].y.ptr[i-1].y.ptr[k].x;
        if( dst < min_dst)
{ if( ( dst == 0 && i > (total_point_count/2) ||
     dst > 0) && (pixel_direction * direction >= 0) )
    {
      min_dst = dst;
      min_loc = k;
    }
}

if( !(ptr[0].x == ptr[min_loc].x && ptr[0].y ==
     ptr[min_loc].y) )
{
  temp_cp = ptr[i];
  ptr[i] = ptr[min_loc];
  ptr[min_loc] = temp_cp;
  total_count_final++;
}
else
  exit = true;

int total_count_final_2 = 1;
for(i = 1; i<total_count_final; i++)
{
  dst = (ptr[i-1].x - ptr[i].x)*(ptr[i-1].x - ptr[i].x) +
        (ptr[i-1].y - ptr[i].y)*(ptr[i-1].y - ptr[i].y);
  if( dst <= 16 )
    ptr[i].x = ptr[i].y = 0;
  else
    total_count_final_2++;
}

int Rmdr = 1 + total_count_final_2 - 4;

if(Rmdr%3 != 0) //not divisible by 3! go ahead and make it
  //divisible by 3...
  Rmdr += (3-(Rmdr%3));

  total_point_count_corrected = Rmdr+4;

CPoint *ptr_final = new CPoint[total_point_count_corrected];
k = 0;

for(i = 1; i<total_count_final; i++)
    if(ptr[i].x > 0 I I  ptr[i].y > 0)
    {
        ptr_final[k] = ptr[i];
        k++;
    }

i = 0;

total_count_final_2--;

while(total_count_final_2 < total_point_count_corrected)
{   ptr_final[total_count_final_2] = ptr_final[i]; //close loop
    total_count_final_2++;
    i++;
}

CRect rect2(0, 0, 645, 520);
CBrush brBackground(RGB(0,0,0));

CDC for_temp_bezier;
for_temp_bezier.CreateCompatibleDC(m_dcp);

CBitmap mBezier_bak;
CBitmap* my_bezier_bak;

mBezier_bak.CreateCompatibleBitmap(m_dcp, 645, 520);
ASSERT(mBezier_bak.m_hObject != NULL);

my_bezier_bak = for_temp_bezier.SelectObject(&mBezier_bak);
for_temp_bezier.FillRect(rect2, &brBackground);

CPen pen1(PS_SOLID,0,RGB(255,255,255));
(&for_temp_bezier)->SelectObject(pen1);

if( (&for_temp_bezier)->PolyBezier( ptr_final,
    total_point_count_corrected) == 0)
    AfxMessageBox( "Error while trying to fit the Bezier curve
    (incorrect number of points).", MB_OK, 0 );
(&for_temp_bezier)->FloodFill( center_pixel.x, center_pixel.y,
RGB(255,255,255) );

for(temp_y2 = 0; temp_y2 < (int)height; temp_y2++)
for(temp_x2 = 0; temp_x2 < (int)width; temp_x2++)
    if( for_temp_bezier.GetPixel(temp_x2,temp_y2) ==
RGB(255,255,255) )
        m_border_array[(width/*height*/temp_y2)+temp_x2] = true;
    else
        m_border_array[(width/*height*/temp_y2)+temp_x2] = false;

bool *TmpBool = new bool[width*height];

memset(TmpBool, 0, width*height*sizeof(bool));

//Erode the image by block_size...
for(int j = 0; j<block_size; j++)
{
    for(temp_y2 = 1; temp_y2 < (int)height-1; temp_y2++)
    for(temp_x2 = 1; temp_x2 < (int)width-1; temp_x2++)
        if( m_border_array[(width*(temp_y2))+temp_x2] &&
            m_border_array[(width*(temp_y2))+temp_x2-1] &&
            m_border_array[(width*(temp_y2-1))+temp_x2] &&
            m_border_array[(width*(temp_y2-1))+temp_x2-1] &&
            m_border_array[(width*(temp_y2+1))+temp_x2] &&
            m_border_array[(width*(temp_y2+1))+temp_x2-1] &&
            m_border_array[(width*(temp_y2-1))+temp_x2+1] &&
            m_border_array[(width*(temp_y2+1))+temp_x2+1])
            TmpBool[(width*(temp_y2))+temp_x2] = true;
        else
            TmpBool[(width*(temp_y2))+temp_x2] = false;

    memcpy(m_border_array, TmpBool, width*height*sizeof(bool));
}

delete [] ptr;
delete [] ptr_final;
delete [] TmpBool;
BIBLIOGRAPHY


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

VITA

Graduate College
University of Nevada, Las Vegas

Theodores Lazarakis

Home Address:
3722 Spitze Drive
Las Vegas, Nevada 89103

Degrees:
Bachelor of Science, Computer Science, 2000
University of Nevada, Las Vegas

Special Honors and Awards:
Dean's Honors List
Minor in Mathematics

Publications:


E. A. Yfantis, B. L. Hagstrom, A. Angelopoulos, T. Lazarakis and A. Popovich, "Digital Motion Distribution and Image Feature Location," ISCA 13th International Conference on Parallel Distributed Computing Systems, August 8-10, 2000, Las Vegas, Nevada


Thesis Title: Prostate Cancer Recognition in Ultrasound Images

Thesis Examination Committee:
- Chairperson, Dr. Evangelos Yfantis, Ph. D.
- Committee Member, Dr. Hal Berghel, Ph. D.
- Committee Member, Dr. John Harrison, Ph. D.
- Graduate Faculty Representative, Dr. Georg Mauer, Ph. D.