Texture Analysis Of Corpora Lutea In Ultrasonographic Ovarian Images Using Genetic Programming and Rotation Invariant Local Binary Patterns

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Meng Dong

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Ultrasonography is widely used in medical diagnosis with the advantages of being low cost, non-invasive and capable of real time imaging. When interpreting ultrasonographic images of mammalian ovaries, the structures of interest are follicles, corpora lutea (CL) and stroma. This thesis presents an approach to perform CL texture analysis, including detection and segmentation, based on the classifiers trained by genetic programming (GP). The objective of CL detection is to determine whether there is a CL in the ovarian images, while the goal of segmentation is to localize the CL within the image.

Genetic programming (GP) offers a solution through the evolution of computer programs by methods inspired by the mechanisms of natural selection. Herein, we use rotationally invariant local binary patterns (LBP) to encode the local texture features. These are used by the programs which are manipulated by GP to obtain highly fit CL classifiers. Grayscale standardization was performed on all images in our data set based on the reference grayscale in each image. CL classification programs were evolved by genetic programming and tested on ultrasonographic images of ovaries. On the bovine dataset, our CL detection algorithm is reliable and robust. The detection algorithm correctly determined the presence or absence of a CL in 93.3% of 60 test images. The segmentation algorithm achieved a mean ($\pm$ standard deviation) sensitivity and specificity of 0.87 ± 0.14 and 0.91 ± 0.05, respectively, over the 30 CL images. Our CL segmentation algorithm is an improvement over the only previously published algorithm, since our method is fully automatic and does not require the placement of an initial contour. The success of these algorithms demonstrates that similar algorithms designed for analysis of *in vivo* human ovaries are likely viable.

Acknowledgements

I am heartily grateful to my supervisor Dr. Eramian. It is an honor for me to work on this research with his supervision and support. Besides, I learn from him the attitudes towards work as a good researcher and a great mentor, which will be a priceless treasure for the rest of my life.

Many thanks to Dr. Simone Ludwig for her valuable advice and expertise on genetic programming and for providing the GP framework program used in this research.

I would like to express my sincere gratitude to Dr. Roger A. Pierson who provides the ultrasound images and ground truth of CL regions used in this thesis, without which this research would not have been possible.

Thanks to my friends for helping me during the work.

Thanks to my parents, for everything.

Contents

List of Tables

List of Figures

List of Abbreviations

GP	Genetic Programming
LBP	Local Binary Patterns
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
LOF	List of Figures
LOT	List of Tables

Chapter 1

Introduction

The purpose of this thesis is to develop computer algorithms to automatically detect and segment corpora lutea (CL) from ovarian ultrasound images. The CL is a gland that forms in the ovary after ovulation and plays a role in maintaining pregnancy. Monitoring CL formation and development is crucial to the understanding ovarian dynamics [43], and to the assessment of reproductive health. Ultrasound imaging is routinely used to monitor the ovaries and their dynamic changes during the estrous cycle in large animal research. The rest of this chapter is organized as follows: background on texture analysis will be introduced first. Then introduction of ultrasonography and some background on ovarian biology are given in the following section. At last, we will elaborate on the objectives and motivations for this project.

1.1 Introduction to Texture Analysis

Texture is a fundamental property of surfaces. It can be seen almost anywhere. For example, trees, bushes, grass, sky, lakes, roads, buildings etc. appear as different types of texture. Two images of natural texture are shown in Figure 1.1.

The process of texture analysis usually produces some kind of numeric descriptions of the texture which are called texture features. The process of computing the texture features is known as feature extraction. Texture classification and segmentation are two main problem domains in texture analysis [49].

Classification refers to assigning a physical object into one of a set of predefined categories. The goal of texture classification is to assign an unknown sample image to one of a set of known texture classes. Texture classification process involves two phases: the learning phase and the recognition phase. The purpose of the learning phase is to build a feature model for the texture content of each texture class present in the training data, which is comprised of images with known class labels. The texture content of the training images are captured with the chosen texture analysis method which yields a set of texture features for each image. These features, which can be scalar numbers or discrete histograms or empirical distributions, characterize textural properties of the images, such as spatial structure, contrast, roughness, orientation, etc. In the recognition phase the same set of learned texture features is generated to form a description of the unknown sample. Then the textural features of the sample are compared to those of the training images with a classification algorithm, and the sample is assigned to the category with the best match.

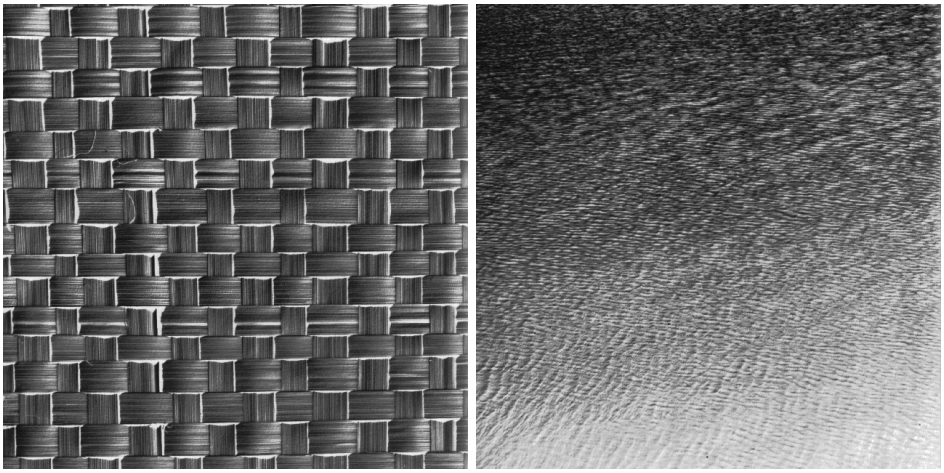


Figure 1.1: Examples of texture images. The first one is the texture of handwoven oriental rattan photograghed in sunlight. The second one is the texture of water with the view of wind and wave patterns.

Texture Segmentation is a more complex texture analysis task which is to partition an image into regions which are homogeneous with respect to textures. Texture segmentation can be considered as an extension of texture classification when classification is used to determine the texture category of a pixel or a region. Texture segmentation shares some principles of image segmentation such as partitioning based on inhomogeneity. However, textured objects pose a great challenge for segmentation since patterns and boundaries can be difficult to identify in the presence of changing scale and lighting conditions. The intensity variation of textures is often overlapping with the background, which may add further difficulty [11]. Therefore, common image segmentation methods such as edge detection, thresholding and region growth are not applicable to texture segmentation [49].

Texture analysis techniques have played an important role in several medical applications. In general, the applications involve the extraction of features from the images which are then used for a variety of tasks, such as distinguishing normal tissue from abnormal tissue. Depending upon the particular classification task, the extracted features capture morphological properties, color properties, or certain textural properties of the image [10].

1.2 Introduction to Ovarian Ultrasonography

1.2.1 Ultrasonography

Ultrasound techniques were first used for measuring distance underwater using SONAR as early as the 19th century [56]. The use of ultrasound in medicine was introduced around the 1930's and the first publication using brightness B-mode ultrasound imaging appeared in the 1950's [56]. There has been a rapid development

of ultrasonographic imaging techniques for clinical diagnosis in recent years. A number of different ultrasound imaging techniques are now available such as 2-dimensional, 3-dimensional ultrasound, Doppler and color-flow ultrasound.

Different from other medical imaging methods such as MRI (Magnetic Resonance Imaging), ultrasound waves are mechanical waves [55], which transmit through the medium but with no permanent displacement of the medium particles. When the sound source is turned off, the particles go back to their original position [55]. Therefore, no foreign substances need to be introduced into the body to interact with the waves, so ultrasound is considered a non-invasive technique. For this reason ultrasound has been found to be a valuable diagnostic tool in wide range of medical disciplines, especially in the fields of obstetrics and gynecology.

Ultrasonography is a very appealing modality for imaging both soft tissue and bony structures. It is low cost, may be used to acquire images in real-time, is non-invasive, and easy to use. However, the speckle noise in ultrasonographic images affects echotexture characteristics making ultrasonographic image interpretation difficult, even under ideal imaging conditions [21].

1.2.2 Background on Ovarian Biology

Since Pierson and Ginther et al. used transrectal ultrasonography to support the wave theory of ovarian follicular development [38], ovarian ultrasonogaphy has developed rapidly and been used to diagnose reproductive pathologies and monitor reproductive physiology for research, such as estimating stage of the estrous cycle [16, 39, 40, 41], detecting ovulation [7, 38, 54] and monitoring hemorrhagic follicles [1].

The ovaries of all mammalian species, including humans and cattle, contain follicles and corpora lutea (CL) which are two important dynamic structures of interest for interpreting ultrasound images. Monitoring the development of ovarian follicles and CL over time is crucial to the understanding of human and bovine reproductive biology, fertility and timing of fertility therapy, the effects of contraception, and the diagnosis of ovarian diseases such as ovarian cysts, cancers, and polycystic ovarian syndrome [43].

Ovarian follicles are fluid-filled cyst-like structures in which oocytes (eggs) develop. Follicles are non-echogenic and appear on the ultrasonographic images as black, roughly spherical structures [15]. Sometimes if the follicles abut with each other they appear more irregularly-shaped. In addition, follicles may take on an ellipsoidal shape before ovulation [14]. The follicle wall is a thin and hypoechoic (reflects relatively few sound waves) area surrounding the follicle antrum. Sometimes, the imaged follicle wall becomes discontinuous or absent because of imaging artifacts that may occur along the curved boundary of follicles. Some of the artifacts may arise from the design of the scanner system, while others are caused by human operation error. Specular and nonspecular reflection, shadowing, enhancement, and reverberation are common types of artifacts in ovarian ultrasound images [15]. The high density and the morphological structure of the follicle wall change during different physiological phases [45]. The smallest follicles that can be detected will vary greatly depending on the design of ultrasound equipment. The minimal diameter of follicles detectable by ultrasonography is around 2mm to 3mm.

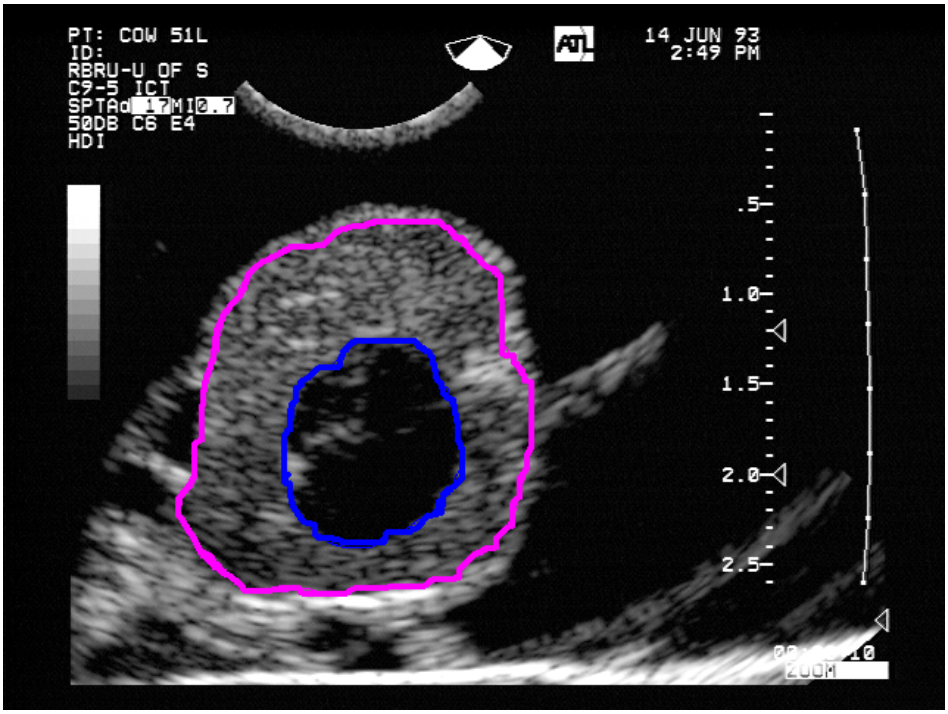


Figure 1.2: An example of a CL image with a central cavity. The magenta contour denotes the boundary for the CL region. The blue contour identifies the boundary of central cavity inside the CL.

The corpus luteum is a temporary endocrine gland that forms from the follicular tissue after ovulation. It is a hypoechogenic structure and the intensity of the corpus luteum varies according to its age [47]. Its primary function is to produce the steroid hormone progesterone, which is necessary for maintaining pregnancy [13]. The central portion of the corpus luteum may show as a hypoechoic structure (formed by solid material) or an anechoic (a structure in which there are no echoes being reflected) structure (due to a fluid-filled cavity) [14]. About 79% of bovine corpora lutea exhibit fluid-filled central cavities [23]. Over time, the central cavity gradually fills with luteal tissues and disappears [26]. Figures 1.2, 1.3 and 1.4 contain examples of a CL image with central cavity, a CL image without central cavity and a non-CL image, respectively. The magenta contour in the CL image denotes the boundary for the CL region as manually determined by an expert. The blue contour identifies the boundary of central cavity inside the CL.

1.3 Motivations and Objectives

The presence of a CL is a direct indication that ovulation has recently occurred. Automated detection and segmentation of the CL in ultrasonographic images would represent a valuable tool in the assessment of reproductive health. In practice, CL detection and segmentation is performed manually, however, this manual delineation is time consuming, and subject to variance in human interpretation of images. The only previously published study on bovine CL segmentation was limited in size and scope and the algorithm

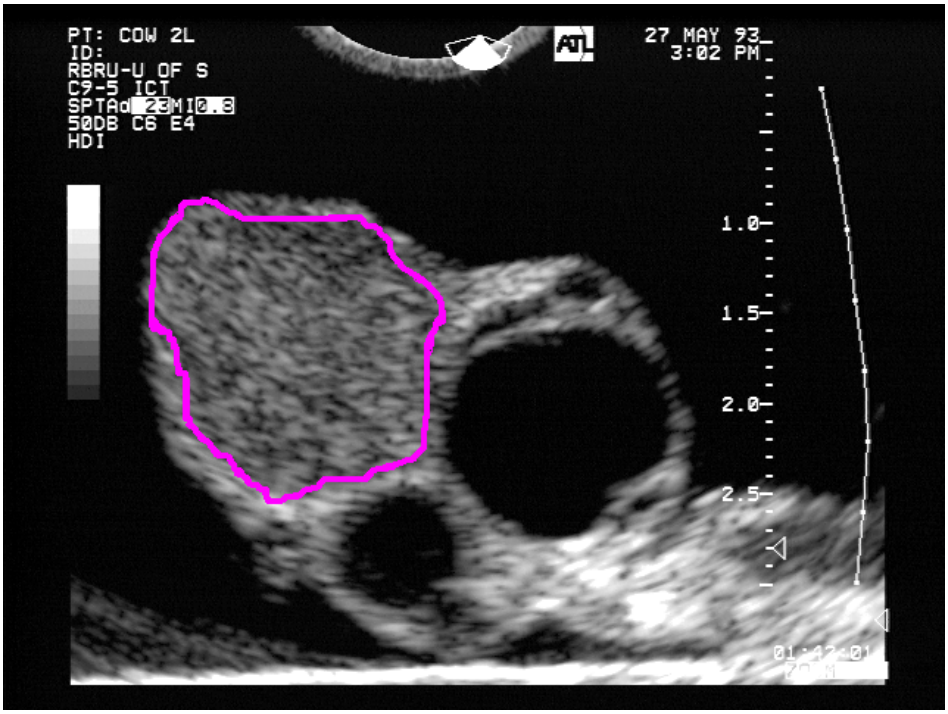


Figure 1.3: An example of a bovine CL image without a central cavity. The magenta contour denotes the boundary for the CL region.

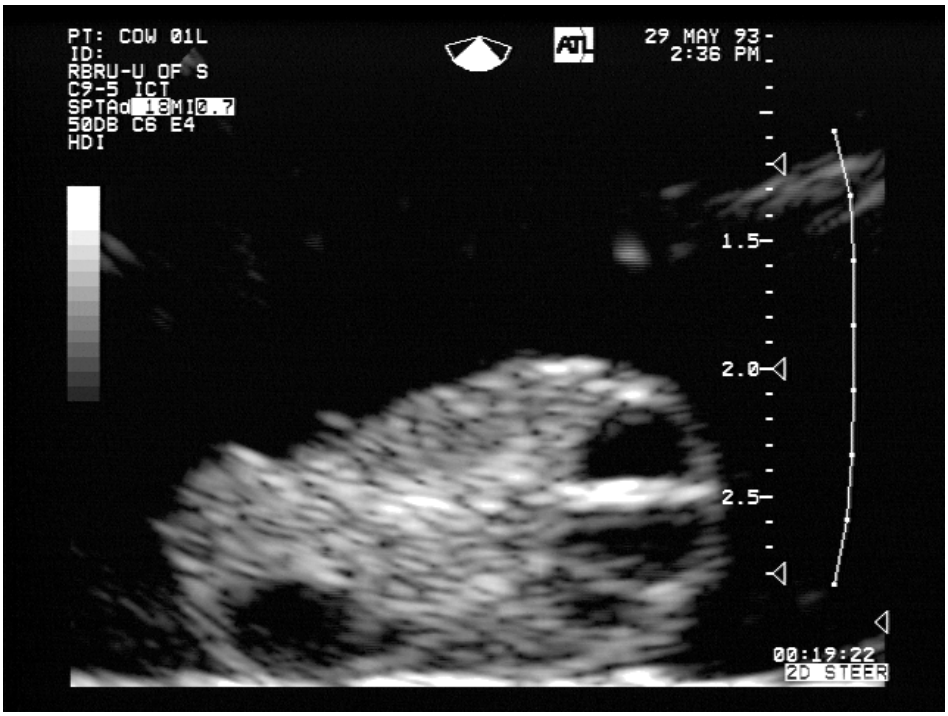


Figure 1.4: An example of a bovine non-CL image. Compared with Figure 1.3, there is no visible CL.

presented was not completely automatic [43]. The authors of that study reported that the mean ($\pm$ standard deviation) sensitivity and specificity of the CL segmentations produced by their algorithm were 0.814 ± 0.171 and 0.990 ± 0.00786, respectively. Their algorithm requires an initial contour to be manually placed within the CL region to be segmented. The motivation of this study is to develop an automatic method to analyze the bovine CL textures without prior knowledge. This will give us a better understanding of ovarian dynamics which is helpful for cattle reproduction research and management [51].

In this thesis, genetic programming (GP) is used to evolve computer programs that perform texture classification using rotation invariant local binary patterns (LBP) to find highly fit algorithms for automatic detection and segmentation of CL's. The performance of the algorithm was evaluated using a data set consisting of ultrasonographic images of bovine ovaries imaged *in vitro*. The bovine model for studying human ovarian function has been well established [2, 48]. This study will inform and facilitate a subsequent study using a more difficult data set of images of human ovaries imaged *in vivo*.

The objectives for this study are as follows:

1. Develop an algorithm to segment CL regions from bovine CL images automatically and accurately without no prior knowledge about the location and shape of the CLs.

2. Develop an algorithm to detect the presence or absence of a CL in an image automatically and classify the Bovine ovarian images into the CL images and non-CL images accurately.

3. Test and evaluate the performance of the segmentation and detection algorithms on a bovine dataset containing images from different time points during estrous cycle and prove the method's generality on CLs from different time points.

Chapter 2

Literature Review and Background

2.1 General Review on Statistical Texture Analysis Approaches

Texture analysis is different from other image analysis methods in that it tries to characterize echotexture of ultrasound images which mostly can not be discriminated visually. This technique has been recognized to be a powerful tool in a variety of applications, such as object segmentation and pattern recognition.

The commonly-used texture analysis approaches are based on the extraction of statistical texture features. Statistical texture description methods define textures based on describing the spatial distribution of gray values. They can be measured from the first-order and second-order statistics [25]. Gray scale histogram is a basic first-order statistical texture feature which was utilized in some of the early studies to measure local textural information in ultrasound images [32, 33]. However, because the first-order features are generated using the absolute gray level values of the images, they are difficult to apply in practice since they depend significantly on the gain or amplification settings of the ultrasound equipment [25]. Gray level co-occurrence matrix (GLCM) [18] is a representative second-order method which has already been widely used in texture classification. A co-occurrence matrix depicts the joint gray-level histogram of the image (or a region of the image) in the form of a matrix with the dimensions of $N_g \times N_g$, where N_g is the number of gray levels of the image. The entries are the joint probability density of pairs of gray levels that occur at pairs of points separated by a specified displacement vector (a given direction at a given interpixel distance). The features derived from GLCM include contrast, correlation, energy and homogeneity, etc. However, there is an inherent problem to choose the optimal interpixel distance in a given situation. Also, the GLCM method, in general, is not efficient since a new co-occurrence matrix needs to be calculated for every selected angle and inter-pixel distance.

The Local Binary Pattern (LBP) operator was first introduced by Ojala et al. [36], as a non-parametric, gray-scale invariant texture analysis model, which summarizes the local spatial structures of an image. The LBP operator is computationally simple and efficient methodology for texture analysis. Later in [37], Ojala et al. proposed to use LBP histogram for rotation invariant texture classification and impressive classification results were achieved on Brodatz texture database. Brodatz textures are 8-bit grayscale images labeled from D1 to D112, and originate from a photographic album published in 1966 [8]. LBP has also been adapted to many other applications, such as face recognition [4], dynamic texture recognition [58] and shape localization

[20]. Zhang et al. [57] defined a new operator called local derivative pattern (LDP), a higher-order directional operator based on LBP, to capture the change of derivative directions among local neighbors, which provided more detailed discriminative information. Face recognition experiment based on LDP in [57] demonstrated that the feasibility and the effectiveness of using LDP for face recognition.

Dimitris et al. [21] developed a texture descriptor called fuzzy local binary patterns (FLBP) by incorporating fuzzy logic in LBP methodology, and evaluated the FLBP performance on thyroid ultrasound images. The fuzzification of LBP includes the transformation of the input pixels to respective fuzzy variables, according to the specified fuzzy rules. Therefore, instead of getting one LBP value from one local neighborhood, each neighborhood contributes to multiple bins of the FLBP histogram with corresponding probabilities. The authors reported a 84% accuracy for classifying ultrasound images of nodular and normal thyroid tissues and demonstrated the better performance of FLBP over other features like GLCM. This study showed that the use of texture representation based on LBP is promising for ultrasound texture classification.

2.2 Ultrasound Ovarian Image Analysis

Ultrasound imaging has the benefits of being non-invasive, easy to operate and inexpensive. Therefore, the use of ultrasonographic technologies in medical image processing have become very common in the last two decades. Ultrasound studies have been conducted on humans as a popular tool for clinical diagnosis and researches such as detecting liver diseases [30, 52], breast cancer [24], and diseases of the uterus [50] and ovary [6]. Besides diagnosing pathologic changes using ultrasound images, researchers have also tried to understand and monitor the biological status of tissues and cells in humans and animals by analyzing the image attributes. The researches on ultrasound ovarian image analysis belong to this category.

The understanding of ovarian follicles has been improved a lot by studies of ovarian follicular status [5, 9, 22, 17]. A good survey of image analysis techniques used in ovarian research was given by Singh et. al. [46], which included spot analysis, line analysis, region analysis, wavelet packet texture analysis and mathematical modelling. These varieties of techniques have been used widely in medical applications. Follicle size and growth rates are important for understanding follicular dynamics. Follicles destined to ovulate will continue to grow, and follicles that die without ovulation will regress. Schwartz et al. [44] and Baerwald [5] both studied ovulatory follicles by measuring the maximum follicle size in ultrasonographic images during the menstrual cycles.

Most of the studies on ultrasound ovarian corpora lutea (CL) are using non-image-processing approaches. The only image-processing-based CL segmentation was given in [43], which used the level set segmentation method together with some image preprocessing techniques. The algorithm requires an initial contour to be manually placed within the CL region to be segmented. The authors of that study reported that the mean ($\pm$ standard deviation) sensitivity and specificity of the CL segmentations produced by their algorithm were 0.81 ± 0.17 and 0.99 ± 0.01, respectively. But the dataset used was limited in size and scope and the algorithm

presented was not completely automatic [43].

2.3 Genetic Programming

2.3.1 Review on Evolutionary Computation

The evolutionary approach to machine learning is based on computational models of natural selection and genetics, which are called evolutionary computation. However, there are different ways of performing evolutionary computation, and two important approaches of them are genetic algorithms (GA) and genetic programming (GP). They both simulate natural evolution, generally by creating a population of individuals, evaluating their fitness, generating a new population through genetic operations, and repeating this process a number of times.

Genetic Algorithms

Genetic algorithms are first introduced by John Holland in the early 1970s who is one of the founders of evolutionary computation [19]. His aim was to make computers do what nature does. As a computer scientist, Holland was concerned with algorithms that manipulate strings of binary digits. He viewed these algorithms as an abstract form of natural evolution. His GA can be represented by a sequence of procedural steps for moving from one population of artificial "chromosomes" to a new population. It uses "natural" selection and genetics-inspired techniques known as crossover and mutation. Each chromosome consists of a number of "genes", and each gene is represented by 0 or 1. However, there are also many other types of string-encoding techniques that have been invented and they are application-dependent [12].

Genetic algorithms use fitness values of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome. After a number of successive reproductions, the less fit chromosomes become extinct, while those with high fitness gradually come to dominate the population. GA does not need knowledge of the problem domain, but it requires a fitness function to evaluate the fitness of a solution. Therefore, a typical process of GA development includes the following steps [35]:

1. Specify the problem, define constraints and optimum criteria.
2. Represent the problem domain as a chromosome.
3. Define a fitness function to evaluate the chromosome's performance.
4. Construct the genetic operators.
5. Run the GA and tune its parameters.

GA is a very powerful tool. However, the fundamental difficulty of GA lies in the representation of problem solutions, that is, in the fixed-length encoding using bit strings. For example, symbolic regression problems, which require the solutions to discover the relationships between different dependent or independent variables, can not be encoded as bit strings, so GA is not able to solve this kind of problems. Therefore, in order to achieve dynamic structural complexity of chromosomes, genetic programming (GP) was introduced as an extension of GA, which creates computer programs as the solutions.

Genetic Programming

Genetic programming was introduced as a problem solving paradigm by Koza in 1992 [27]. Genetic Programming offers a solution to user-defined tasks through the evolution of computer programs by methods inspired by the mechanisms of natural selection [27, 28]. GP searches the space of possible computer programs for one program that is highly fit for solving the problem at hand [35]. GP is a flexible method of problem solving for a diverse range of complex problems, such as classification [49].

Solving a problem by genetic programming involves determining the set of terminals, selecting the set of functions, defining a fitness function to evaluate the performance of created computer programs, and choosing the method for designating a result of the run [35]. More details are given in the following Section 2.3.2.

A number of researchers have successfully used GP for various image-related tasks, such as object detection [31] and image segmentation [42]. Classifiers learned via GP were also proved to be capable of classifying Brodatz textures [49].

2.3.2 The Procedure of Genetic Programming

Preparatory Steps

GP can operate on programs given by various representations, such as program trees, graphs, and linear representations. In our study, since the output classifier of GP is a symbolic expression combining different operations, functions, variables and constants, program trees are the most suitable representations. A simple example of a program tree is displayed in Figure 2.1.

In GP, the inputs of the computer programs are called terminals. All the leaf nodes in Figure 2.1 are terminals. The functions in GP can be presented by standard arithmetic operations, logical operations, mathematical functions and domain-specific functions, etc. All the internal nodes in Figure 2.1 are functions.

When applying GP to a problem, we have to accomplish five preparatory steps [28], which are described as follows:

1. Determine the set of terminals. The choice of terminal set is dependent on the specific task to solve. For image-related problems, pixels or some other extracted features using pixel statistics can be used as terminals. Besides, the use of constant terminals can be found in the majority of image-related GP applications, although there is no consistent choice of constants [49].

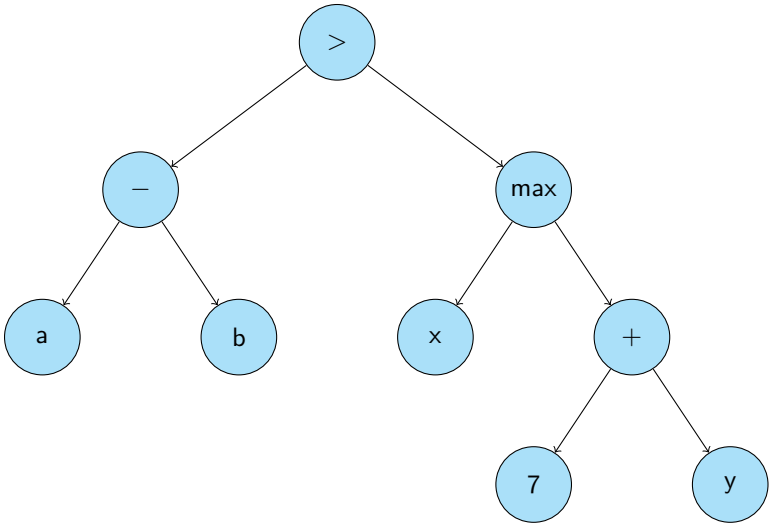


Figure 2.1: An example of program tree. Terminal nodes contain program inputs and internal nodes represent program functions.

2. Determine the set of functions. The choice of functions also depends on the task. The functions can be presented by standard arithmetic operations, standard mathematical functions, logical functions or domain-specific functions.

   Terminals and primitive functions together constitute the building blocks from which genetic programming constructs a computer program to solve the problem [35].

3. Define the fitness function. A fitness function evaluates how well a particular computer program can solve the problem. The choice of the fitness function depends on the problem, and may vary greatly from one problem to the next [35]. However, since most of the work used GP as a supervised learning method, it is relatively easy to define fitness to be the differences between program outputs and predefined goals.

4. Decide the parameters for controlling the run. Usually, the parameters for controlling the run includes the population size, the maximum number of generations to run and the probabilities for genetic operators.

5. Choose the method for designating a result of the run. It is common practice in GP to designate the best-so-far generated program as the result of a run.

Once these five steps are complete, a run can be made. The run of genetic programming starts with a random generation of an initial population of computer programs. In the initial population, all computer programs usually have poor fitness, but some individuals are more fit than others; the more fit programs are more likely to survive into the next generation [35].

Genetic Operators

In genetic programming, the next population of programs is constructed one at a time. Each member of the new population is generated by one of several genetic operators. The genetic operators used in GP are cloning, crossover and mutation. Each of these three operations occurs with a pre-defined probability. Genetic operators operate on computer programs, with some recombining their operands to form completely new programs. Operands for each application of a genetic operator are selected based on their fitness, using some pre-defined selection method.

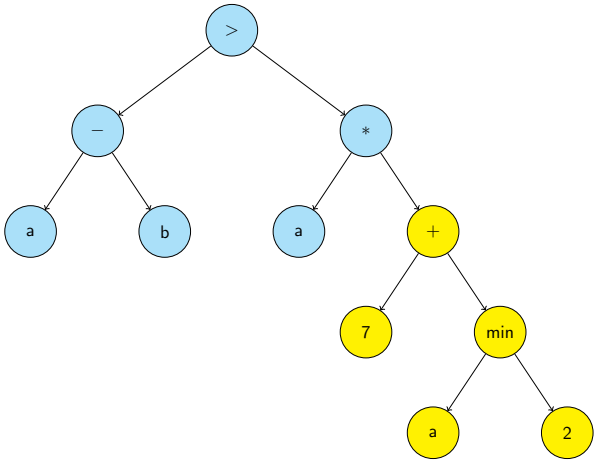
The cloning operator simply copies a program tree, unmodified, to the next generation.

The crossover operator operates on two "parent" program trees which are selected on the basis of their fitness. These program trees can have different size and shape. The two "offspring" programs are composed by recombining randomly chosen subtrees of their parents; a subtree of one parent program is exchanged with a subtree of the other parent.

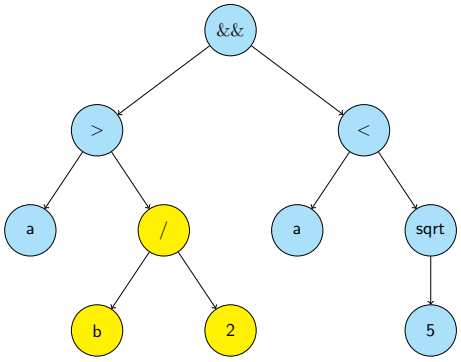
Any internal or external node can be chosen as a crossover point. Two program trees are displayed in Figure 2.2. Suppose that the crossover point for the first parent is the function "+" and the crossover point for the second parent is the function "/". As a result, we obtain the two crossover fragments rooted at the chosen crossover points which are the yellow subtrees in Figure 2.2. The crossover operator creates two offspring by exchanging the crossover fragments of two parents. Thus, the first offspring is created by inserting the crossover fragment of the second parent into the place of the crossover fragment of the first parent. Similarly, the second offspring is created by inserting the crossover fragment of the first parent into the place of the crossover fragment of the second one. The two children programs are shown in Figure 2.3. The crossover operator produces valid offspring computer programs regardless of the choice of crossover points [35].

The mutation operator operates on a single program tree and randomly changes a function or terminal in that tree. Under mutation, a function can only be replaced by a function and a terminal can only be replaced by a terminal. Two examples are given in Figure 2.4 and Figure 2.5. Figure 2.4(a) displays one parent program tree in which the yellow node is the mutation point, while the offspring is given in Figure 2.4(b) where the original function "+" is mutated to "−". Another example of terminal mutation is given in Figure 2.5 where the terminal "a" in the original program tree is mutated to "b".

In our experiment, individual program trees are selected as the operand(s) for the chosen genetic operator using *tournament selection* [34]. A tournament is run by choosing k participant programs (k is the tournament size) at random from the current population. The winner of a tournament is the participant with the highest fitness and is used as an operand of the genetic operator. A tournament is run for each operand needed by the operator. The result of the genetic operator on the tournament winner(s) is then added to the new population. This process is repeated until the new population has the same size as the current population. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, less fit individuals are less likely to win a tournament.

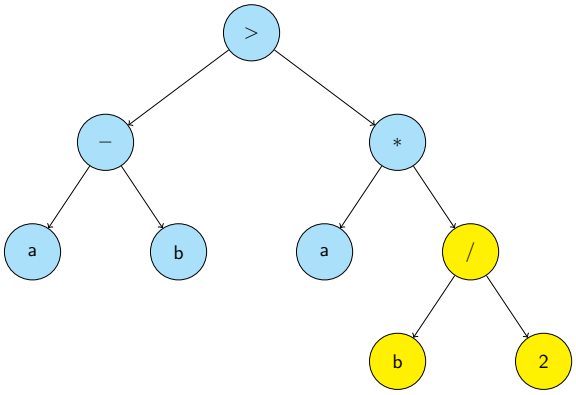


(a) The first parent program tree

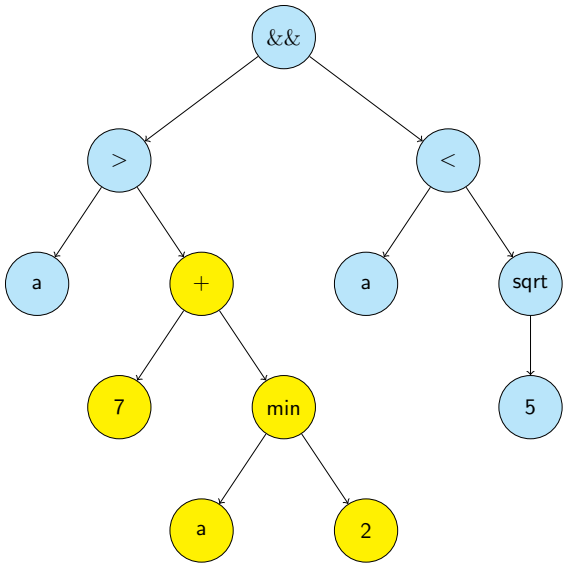


(b) The second parent program tree

Figure 2.2: Two parent program trees for crossover. The crossover fragments are the nodes filled with yellow color.

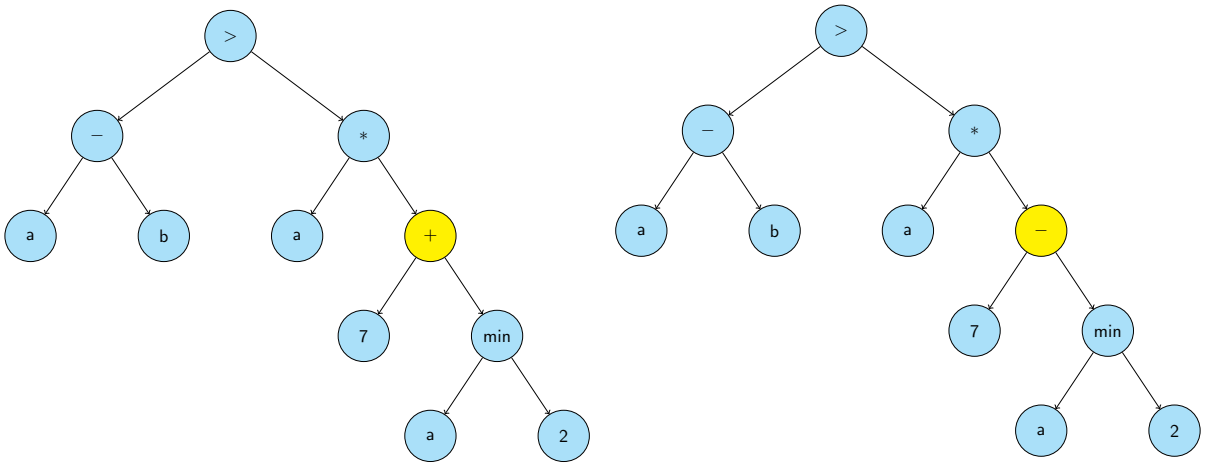


(a) The first offspring



(b) The second offspring

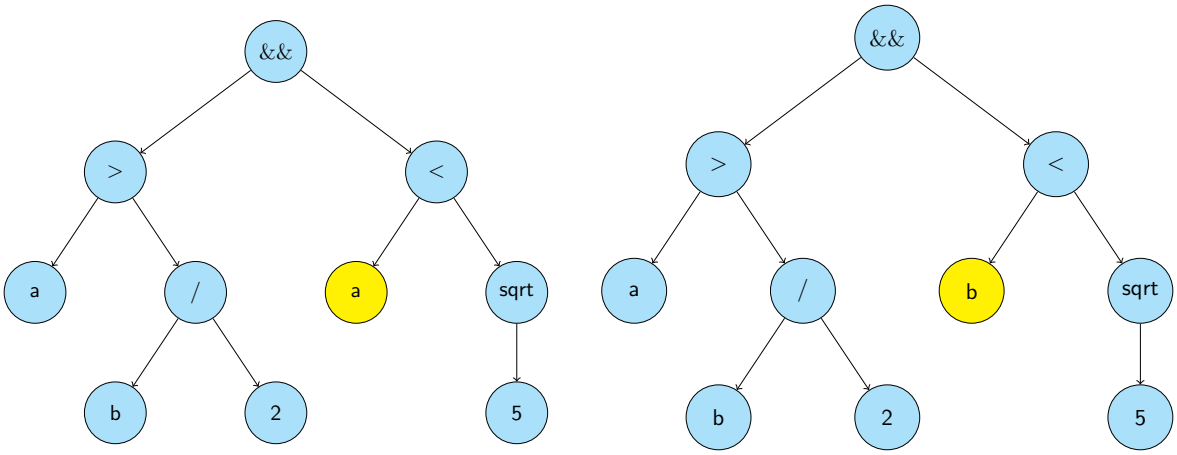
Figure 2.3: The offspring produced by the crossover operation illustrated in Figure 2.2. The exchanged crossover fragments are the nodes filled with yellow color.



(a) The original program tree

(b) The mutated offspring program tree

Figure 2.4: An example of function mutation. The function + in original program tree is mutated to −.



(a) The original program tree

(b) The mutated offspring program tree

Figure 2.5: An example of terminal mutation. The terminal a in original program tree is mutated to b.

GP Procedure

The steps of creating computer programs using GP are executed as follows:

1. Determine the maximum number of generations to run and probabilities of occurrence of cloning P_{cl}, crossover P_c and mutation P_m.

2. Generate an initial population of computer programs of size N by combining randomly selected functions and terminals. The initial program trees are generated subject to a pre-established maximum depth D specified by the user, so the initial generation can contain program trees with different shapes and depths (less than or equal to D).

3. Execute each program in the population and calculate its fitness with an appropriate fitness function. Record the most highly fit program as the *best-so-far* individual I_{bsf}.

4. With the assigned probabilities from step 1, select a genetic operator (crossover, mutation or cloning) to perform.

5. (a) Define the selection method in GP runs. In this study, tournament selection is used to select operands for all the genetic operators.

   (b) If the cloning operator is chosen, use the pre-defined selection method (e.g. tournament selection) to select one computer program from the current population of programs and copy it into a new population.

   (c) If the crossover operator is chosen, select a pair of computer programs (parents) from the current population using the pre-defined selection method, create a pair of offspring programs by recombining randomly chosen parts of their parents and place them into the new population.

   (d) If the mutation operator is chosen, select one computer program from the current population using the pre-defined selection method, perform mutation by changing a function or terminal randomly to another function or terminal, and then place the mutant into the new population.

6. Repeat steps 4 and 5 until the size of the new population of computer programs becomes equal to the size of the initial population, N.

7. Replace the current population with the new population.

8. Go to step 3 and repeat the process until the termination criterion is satisfied. Usually, the termination criterion is that the fitness of best-so-far program I_{bsf} exceeds some predetermined threshold or that a predetermined number of generations have been processed [35].

The flowchart of GP is shown in Figure 2.6. The numbers in the flowchart correspond to the steps of GP procedure described above.

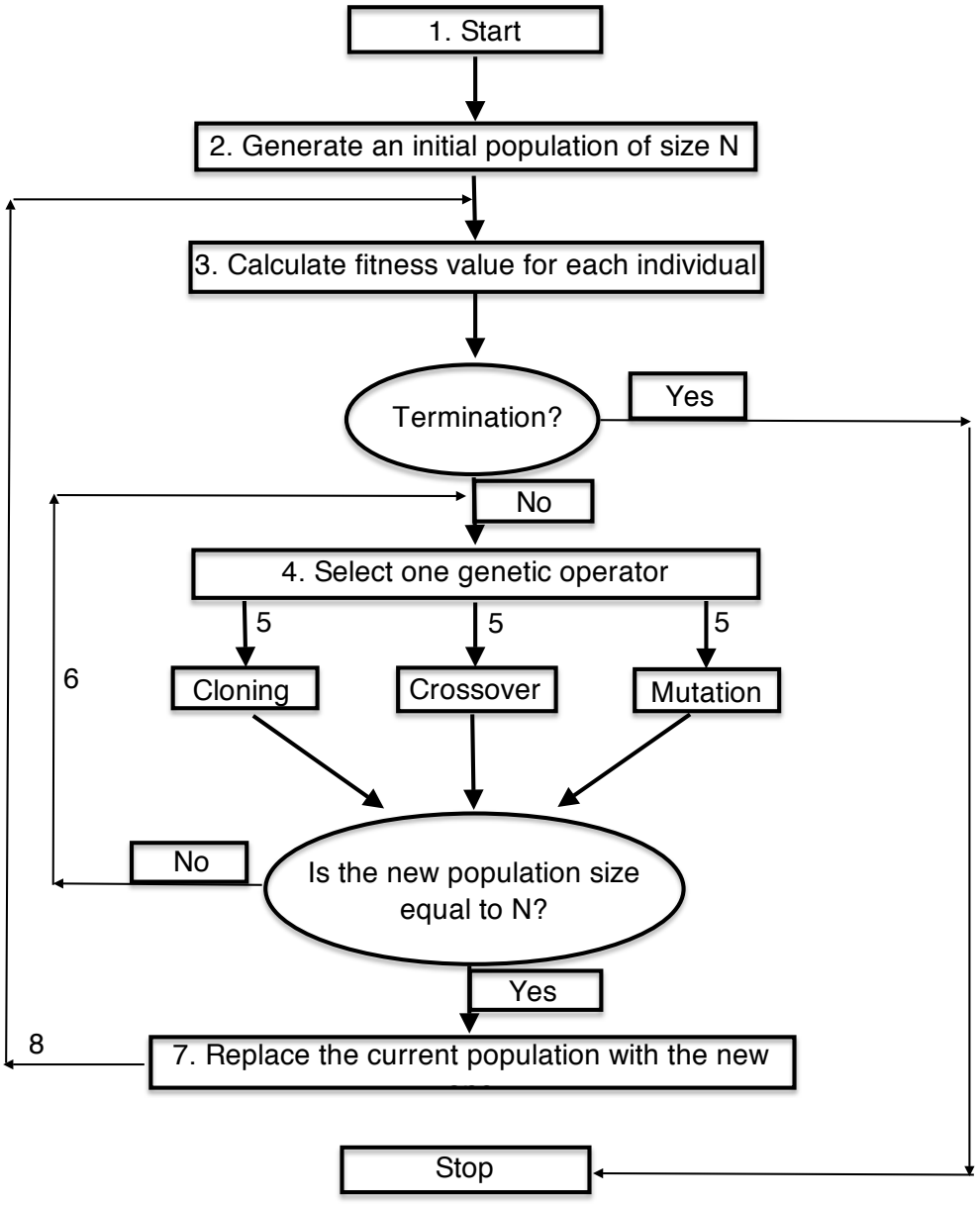


Figure 2.6: The flowchart of GP.

2.3.3 GP Performance Graph

During the procedure of GP illustrated in the section 2.3.2 the fitness values of chromosomes usually vary from generation to generation. As a result, using a performance graph is a good way to examine the behavior of GP over the chosen number of generations. There are two kinds of performance graphs. One plots a curve showing the average performance of the entire population of chromosomes, that is, plot the average values of fitness values of all the program trees. The other kind of performance graph plots a curve showing the performance of the best individual in the population, which is a curve of best-so-far fitness over the generations. The latter one is used in our experiment, since we will take use of the best-so-far individual as the final output classifier.

An example of a performance graph is given in Figure 2.7 which plots the best and average values of the fitness function across 400 generations. The horizontal axis of the performance graph indicates how many generations have been created and evaluated at the particular point in the GP run, and the vertical axis displays the value of the fitness function at that point. The "best" and "average" curves represented here are typical for GP. As can be seen from the figure, the "best" curve rises rapidly at the beginning of the run, but then as the population converges on the nearly optimal solution, it rises more slowly, and finally flattens at the end. The "average" curve was much lower than the "best" curve at the beginning, since there are only a few individuals with relatively high fitness due to the randomness of the initial population. Then the "average" curve will rise faster than the "best" curve, which indicates that the overall fitness of the whole population is increasing rapidly, and at last the average fitness will converge at a value close to the best fitness.

2.4 Local Binary Patterns

2.4.1 Local Binary Pattern Operators

Original Local Binary Patterns

The original LBP operator first introduced by Ojala et al. [36] was based on a 3×3 local neighborhood representing the local texture around a central pixel. The value of each peripheral pixel of this neighborhood is thresholded using the value of the central pixel. Figure 2.8 (a) depicts a 3×3 neighborhood, where p_i $(0 \leq i \leq 7)$ corresponds to the pixel locations and p_{center} indicates the central pixel. The gray value of each pixel p_i is denoted as g_i, and g_{center} is the gray value for the central pixel. Therefore, after thresholding each peripheral pixel by the central pixel's gray value g_{center} according to Equation 2.1, the 3×3 neighborhood can be characterized by a set of binary values d_i $(0 \leq i \leq 7)$, which is shown in Figure 2.8 (b). Therefore, based on the binary values in Figure 2.8 (b) and the corresponding weights for each bit in Figure 2.8 (c), the LBP code can be computed using Equation 2.2 which is the sum of the value of each bit shown in Figure 2.8d. Thus, the local texture information is represented by a 8-bit binary number with an integer value. The

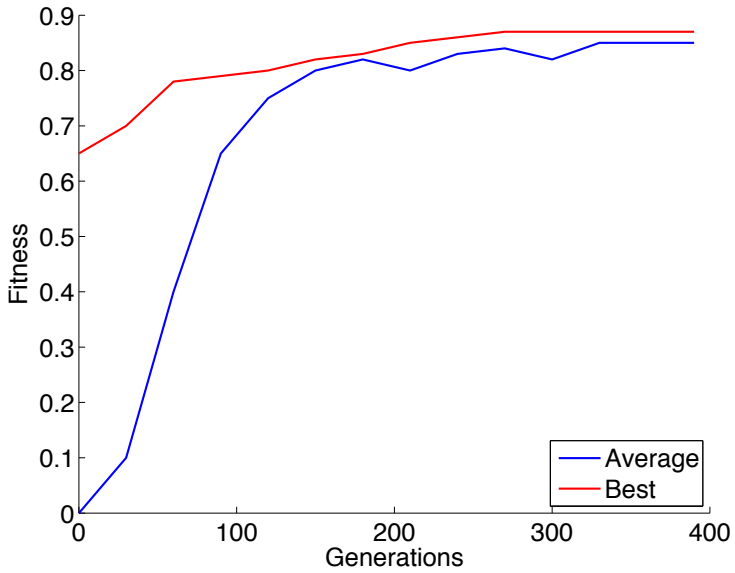


Figure 2.7: An example of GP performance graph. The x-axis indicates the number of processed generations and the y-axis displays the value of the fitness at that point.

possible number of LBP codes is 256, from 0 to 255. The name "Local Binary Patterns" reflects the nature of the operator, namely a local neighborhood is thresholded at the gray value of the center pixel into a binary pattern [37].

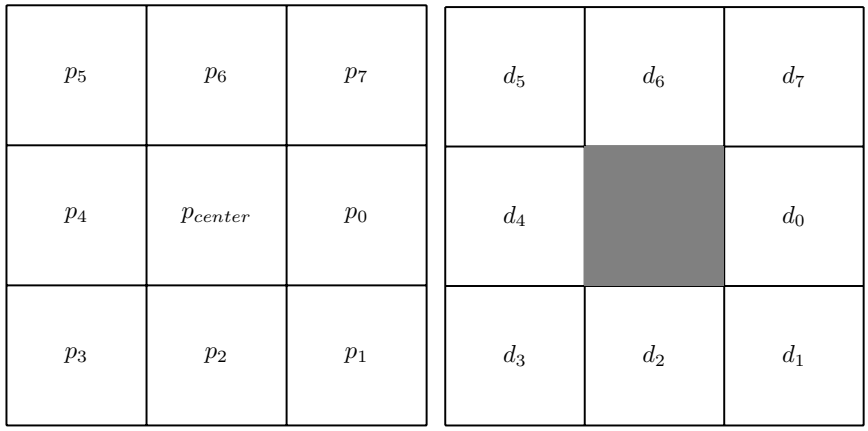
$$d_i = \begin{cases} 1, & g_i \geq g_{center} \\ 0, & g_i < g_{center} \end{cases} \tag{2.1}$$

$$LBP = \sum_{i=0}^{7} d_i * 2^i \tag{2.2}$$

However, since the distance between the central pixel and diagonal pixels (p_1, p_3, p_5 and p_7) is not the same as the distance from the central pixel to the horizontal or vertical pixels (p_0, p_2, p_4, p_6), a circular layout is used in many studies. Figure 2.9 illustrates a circular layout of a 3×3 local neighborhood. The g_i for each p_i in the circular layout is determined by bilinear interpolation [37]. This circular layout is used in this thesis.

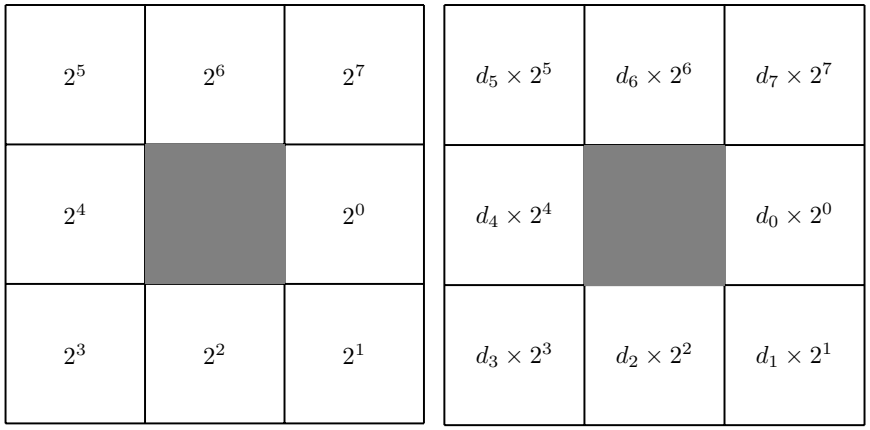
Rotation Invariance

The original LBP operator has 256 (2^8) different output values, corresponding to the 256 different binary patterns that can result from thresholding the eight interpolated intensities within the 3×3 neighborhood. When an image is rotated, interpolation positions p_i will also move along the perimeter of the circle around the center pixel, so rotating a particular pattern will result in a different LBP value. In the case of rotation by



(a) A 3×3 local neighborhood, p_i is the location of each pixel.

(b) Binary local neighborhood after thresholding peripheral pixels by central pixel.



(c) The weight of each bit.

(d) The integer value of each bit.

Figure 2.8: LBP computation scheme.

an increment of 90°, the result is a circular permutation of the bit pattern for the unrotated neighbourhood. Only two patterns, 00000000_2 and 11111111_2, will always remain constant against any rotation.

To overcome this deficiency, rotationally invariant 8-bit local binary patterns [37], denoted LBP_8^{ri}, have been defined:

$$LBP_8^{ri} = \min\{\text{ROR}(LBP_8, i) \mid i = 0, 1, ...7\}, \tag{2.3}$$

where $\text{ROR}(x, i)$ performs i circular bitwise right shifts on the 8-bit number x. There are 36 different possible values for LBP_8^{ri}, corresponding to 36 unique rotation invariant local binary patterns. An example of the computation for rotation invariant LBP is shown in Figure 2.10.

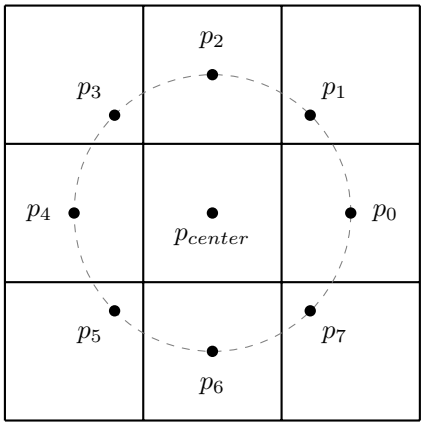


Figure 2.9: 3×3 circular neighbor set. The gray values of diagonal pixels are determined by bilinear interpolation.

"Uniform" Patterns

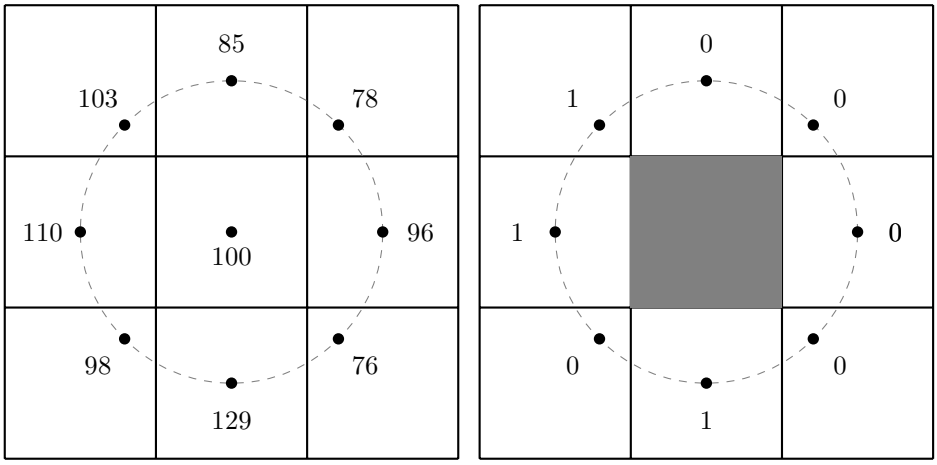
However, according to Ojala et. al [37], the performance of the 36 LBP_8^{ri} patterns varies greatly in discrimination of rotated textures: some patterns sustain rotation quite well while others do not and only confuse the analysis. Consequently, using all 36 patterns leads to a suboptimal result. Thus, a uniformity measure U is defined, which corresponds to the number of transitions (bitwise 0/1 changes) in the bit pattern when read from left to right. For example, $U(00000000_2) = U(11111111_2) = 0$, and $U(00001000_2) = 2$. The smaller the uniformity value U of a pattern is, that is, the smaller number of spatial transitions occurs in the pattern, the better the pattern will sustain rotation [37]. Therefore, the patterns that have U value of at most 2 are designated as "uniform" patterns. The rotation invariant LBP uniform patterns, LBP_8^{riu2}, are defined as follows [37]:

$$LBP_8^{riu2} = \begin{cases} \sum_{i=1}^{8} d_i & U(LBP_8) \leq 2, \\ 9 & \text{otherwise.} \end{cases} \tag{2.4}$$

The "nonuniform" patterns are all represented by the same value of 9, so the possible values of LBP_8^{riu2} are integers in the interval $[0, 9]$. All the 8-bit rotation invariant uniform patterns are displayed in Figure 2.11 with their corresponding LBP_8^{riu2}. Equation 2.4 assigns a label corresponding to the number of 1-bits in each uniform pattern illustrated in Figure 2.11, and all the other patterns are grouped into one class with the label 9 [37].

16-bit Rotation Invariant Local Binary Patterns

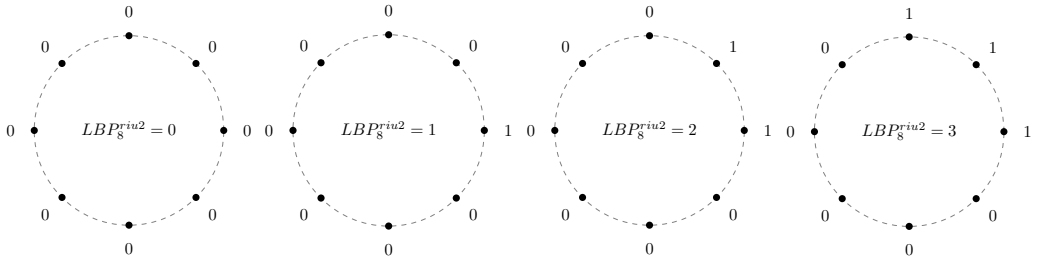
A 8-bit LBP is obtained from a 3×3 neighborhood with a 45° quantization of the angular space provided by the eight interpolation points. However, in order to get better classification accuracy, a more precise resolution of 22.5° is used to quantize the angular space. Figure 2.12 shows the circular neighbor set of 16



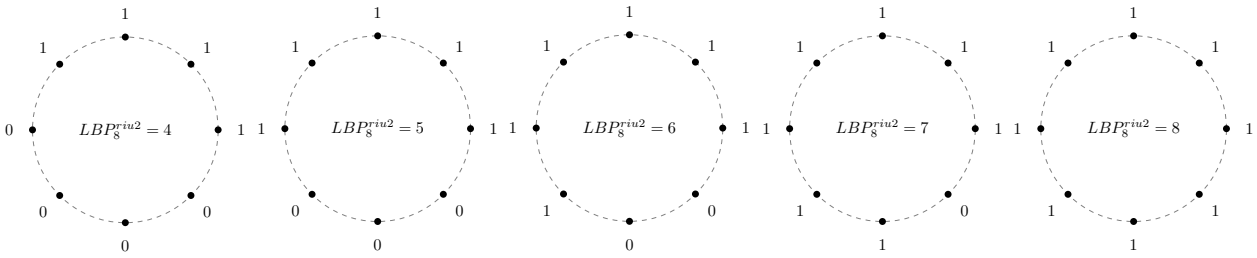
(a) A 3×3 circular local neighborhood with pixel values.

(b) The binary value for each bit obtained from Equation 2.1. The order of all the bits are the same as in Figure 2.9, so the pattern here is 01011000_2. The corresponding rotation invariant local binary pattern is 00001011_2, which is obtained using Equation 2.3.

Figure 2.10: An example of computing the rotation invariant LBP in a 3×3 neighborhood.



(a) Uniform pattern 0 (b) Uniform pattern 1 (c) Uniform pattern 2 (d) Uniform pattern 3



(e) Uniform pattern 4 (f) Uniform pattern 5 (g) Uniform pattern 6 (h) Uniform pattern 7 (i) Uniform pattern 8

Figure 2.11: The 8-bit rotation invariant uniform patterns with their corresponding LBP_8^{riu2} values. All the other patterns have a LBP_8^{riu2} value of 9.

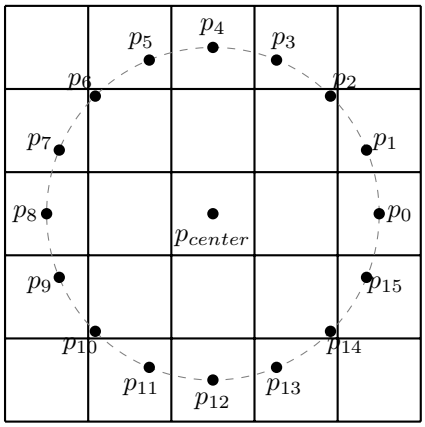


Figure 2.12: 5×5 circular neighbor set. The gray values of p_i which do not fall exactly in the center of pixels are obtained by bilinear interpolation.

pixels. Similarly, p_i indicates the location of the pixels and g_i is the gray values of each pixel. The gray values of p_i which do not fall exactly in the center of pixels are still obtained by interpolation. One detail to note is that the size of the local neighborhood is increased from 3×3 to 5×5, because the 8 newly added neighbors will not provide too much new information if they are inserted into the 3×3 neighborhood [37].

The definition of 16-bit original LBP is similar with the 8-bit one. The local neighborhood in Figure 2.12 is thresholded by the center pixel into a 16-bit binary pattern, while the value of LBP_{16} operator is defined as follows:

$$LBP_{16} = \sum_{i=0}^{15} d_i * 2^i, \tag{2.5}$$

where

$$d_i = \begin{cases} 1, & g_i \geq g_{center} \\ 0, & g_i < g_{center} \end{cases} \tag{2.6}$$

The LBP_{16} operator has 2^{16} different values and the rotation invariant patterns are also defined by performing a circular bit-wise right shift (ROR) on the 16-bit binary number. The uniform patterns are, again, those with at most two 0/1 transitions. The definitions of rotation invariant 16-bit LBP LBP_{16}^{ri} and uniform patterns LBP_{16}^{riu2} are as follows:

$$LBP_{16}^{ri} = \min\{\text{ROR}(LBP_{16}, i) \mid i = 0, 1, ...15\}, \tag{2.7}$$

$$LBP_{16}^{riu2} = \begin{cases} \sum_{i=1}^{16} d_i, & U(LBP_{16}) \leq 2 \\ 17 & \text{otherwise.} \end{cases} \tag{2.8}$$

Thus, the LBP_{16}^{riu2} operator has 18 output values. The values from 0 to 16 correspond to the numbers of 1-bits in the 17 uniform patterns, and value 17 denotes nonuniform patterns [37]. In our experiments, the histograms of LBP_{16}^{riu2} values computed from a subimage are used to encode local texture descriptions.

Chapter 3

CL Segmentation and Detection Algorithms

3.1 Image Preprocessing

3.1.1 Reference Grayscale Bar Detection

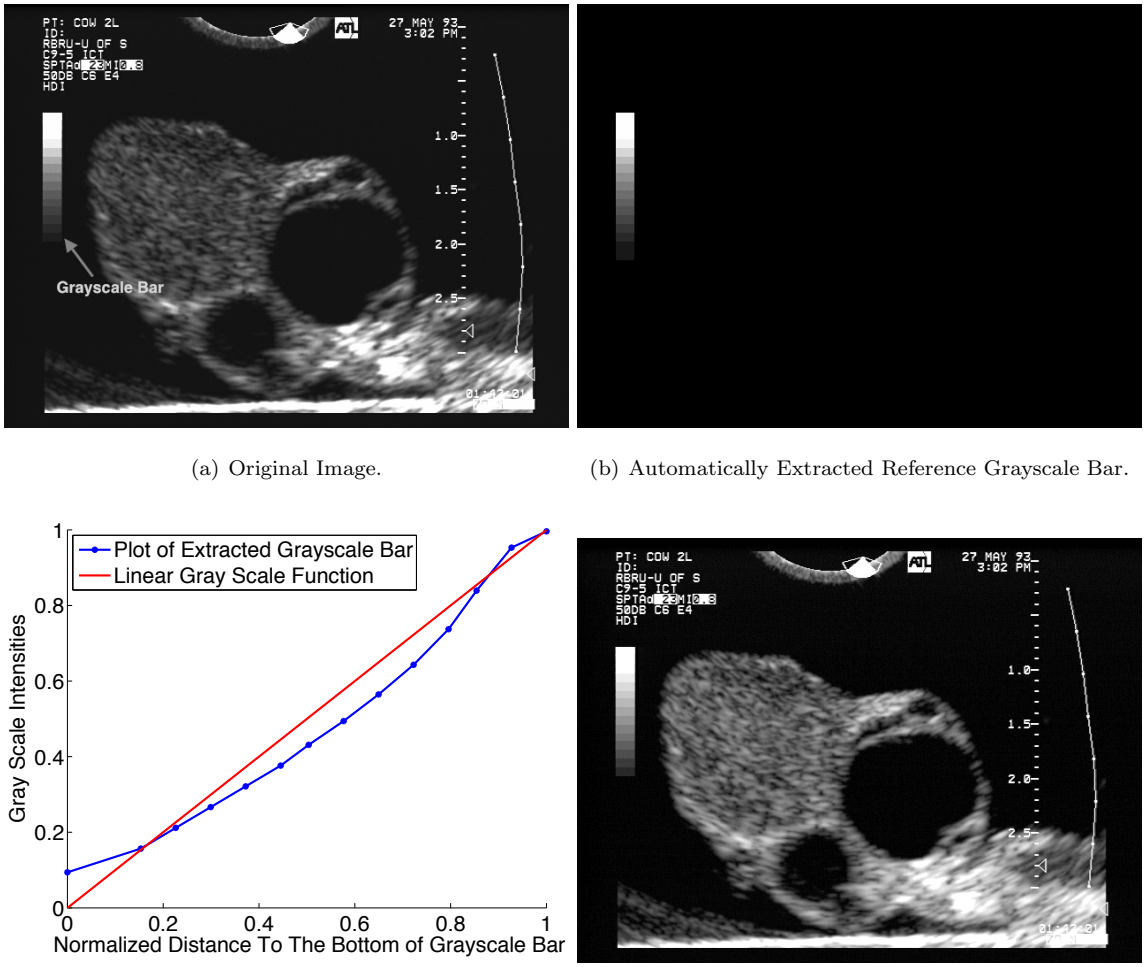
Grayscale standardization is performed as the preprocessing on all the images in our dataset (described in Section 4.1). Each ultrasound image in our dataset contains a reference grayscale bar which appears as steps of decreasing intensity levels. An example image with grayscale bar is given in Figure 3.1 (a). In order to perform grayscale standardization, we have to first extract the reference grayscale bar from the ultrasound images, using the algorithm proposed in [3]. The grayscale bar is detected by searching for consecutive pixel columns with long sequences of monotonicity in graylevels. Therefore, the reference grayscale bar detection algorithm consists of smoothing, computing an monotonicity measure for each column and extracting the reference grayscale from high-monotonicity columns [3].

Smoothing and Downsampling

Some ultrasound images are subject to digitization errors. The reference grayscale in such images is not always monotonic due to extrema within individual steps resulting from the noise. Smoothing is the way to deal with noise, but standard filters may produce a non-monotonic function when applied to a monotonic function corrupted by noise. Therefore, smoothing the function with simultaneous downsampling is used to solve this problem. Every w pixels were replaced by the mean value of their neighbourhood. Given an input function x of length L, the output y of length L/w is computed as follows [3]:

$$y[n] = \frac{1}{w} \sum_{i=nw}^{nw+w-1} x[i], n = 0, 1, 2, ...L/w \tag{3.1}$$

The kernel window is moved in steps of w pixels. The averaging at each kernel location produces a satisfactory estimate of true intensity. Downsampling guarantees that each new value is either equal to or less than the previous one if w exceeds the step width and the noise levels are smaller than the step height, since the underlying function is a monotonic step function corrupted by noise [3]. The value of w is set to be 5 which always produced monotonic results for our dataset.



(a) Original Image.

(b) Automatically Extracted Reference Grayscale Bar.

(c) Standardization Curve for Grayscale Remapping.

(d) Standardized Image.

Figure 3.1: The image grayscale standardization process.

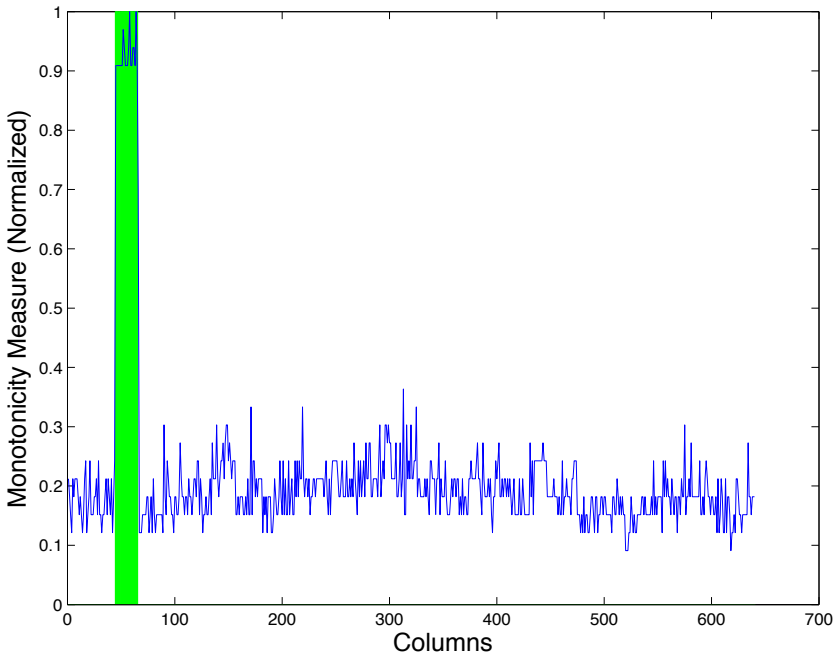


Figure 3.2: Monotonicity measure for all columns in the image displayed in Figure 3.1 (a). The reference grayscale is detected from consecutive columns of high monotonicity measure which is highlighted in green.

Monotonicity Measure

After smoothing and downsampling the image, the column intensities are used to get the longest sequence of monotonically decreasing intensities in each column. Such longest sequence of each column is normalized to the longest sequence in the whole image, which will produce a monotonicity measure between 0 and 1. This normalization makes the monotonicity measure insensitive to different length of reference grayscale. Besides, the start and end coordinates of each longest sequence in every column are recorded. The plot of monotonicity measure for all the columns in Figure 3.1 (a) is displayed in Figure 3.2.

Grayscale Bar Extraction

A threshold for monotonicity measure of 0.7 is used to select the columns containing the reference grayscale, so the algorithm selects the longest contiguous set of columns with monotonicity measure larger than 0.7 as those containing the reference grayscale. Then, the leftmost and rightmost selected columns are the left and right borders of the reference grayscale, while the top and bottom coordinates of reference grayscale are already recorded from the last step. Therefore, we can extract the reference grayscale bar accordingly using the coordinates. The extracted grayscale bar from Figure 3.1 (a) is shown in Figure 3.1 (b).

This reference grayscale detection algorithm can be summarized as follows [3]:

1. Smoothing and downsampling the image using Equation 3.1.

2. Find the maximum pixel distance L_i where intensities are monotonically decreasing for the i_th column. Normalize the distance values by the maximum L_i.

3. Thresholding the sequence of normalized L_i and then the longest contiguous sequence of columns are selected as those containing the reference grayscale.

4. Extract the reference grayscale using the bounding columns as horizontal coordinates and endpoints of monotonic decreasing sequence as vertical coordinates.

3.1.2 Grayscale Standardization

Each image in our data set contains a reference grayscale bar which appears as steps of decreasing intensity levels. Two examples are given in Figure 3.3 and 3.4, including the original and standardized images displayed side by side. Grayscale standardization was performed on all images in our dataset as follows:

1. The grayscale bar was localized and the unique intensities in the grayscale bar identified using the algorithms described in Section 3.1.1 (Figure 3.1(a) and (b));

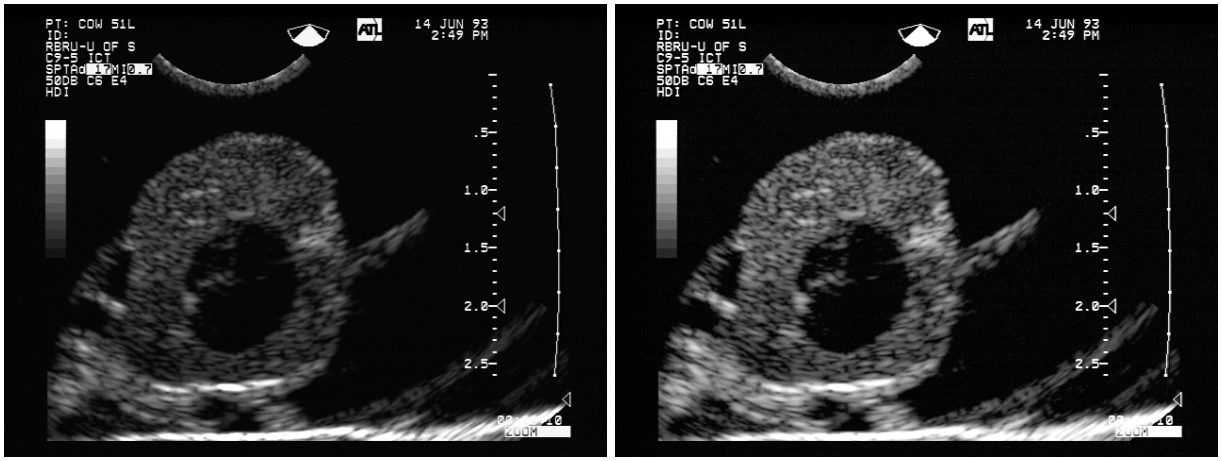
2. The intensities obtained from Step 1 were remapped to an increasing linear function of pixel distance from the bottom of the grayscale bar. Figure 3.1(c) plots the curve for grayscale remapping of Figure 3.1(a). The horizontal axis indicates the normalized distance to the bottom of the grayscale bar; the vertical axis is the intensity represented as real numbers between 0 and 1;

3. Since intensities that do not appear in the grayscale bar may appear elsewhere in the image, mappings for missing intensities were determined by linearly interpolating the result in Step 2; and the intensity of each pixel in the image was mapped to its new value according to the established linearizing mapping.

3.2 Learning CL Texture Classifiers by Genetic Programming

In this study, CL texture was treated as one class and all the other textures were considered as the other class. Program trees were used to represent solutions and the tree output indicated the decision of whether a local texture description was CL or not. The function set and terminal set are listed in Table 3.1 and Table 3.2, respectively.

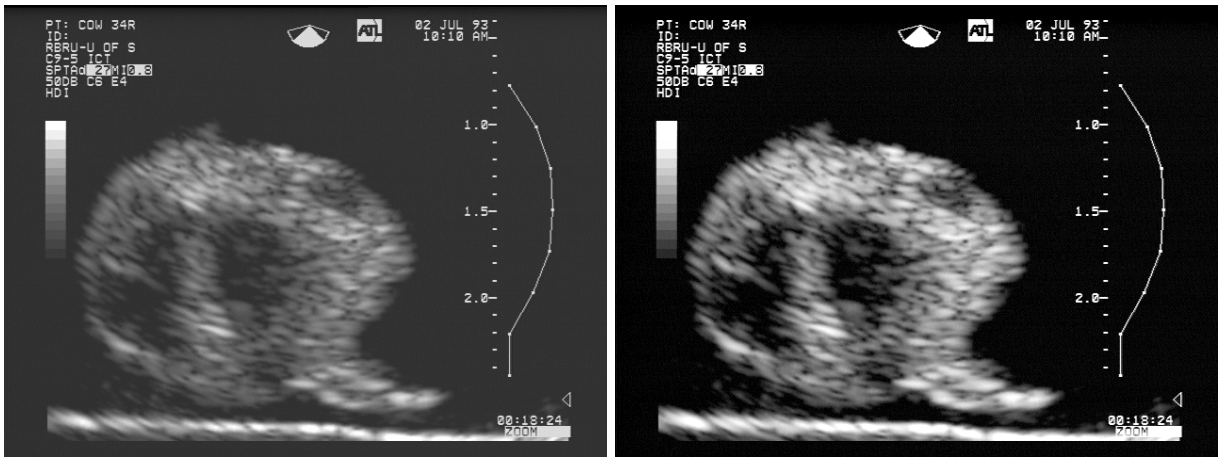
The function set includes basic arithmetic and logical operators. The terminal set consists of the histogram bin values of LBP_{16}^{riu2} and random numbers. The random numbers in the terminal allow construction of programs that can not only compare the histogram bin values with each other but also compare them with constant numbers. For similar textures, the LBP_{16}^{riu2} histograms may have similar trend but very different values in each bin, so just comparing the bin values with each other may result in a non-optimal classifier.



(a) The original CL image

(b) Standardized image

Figure 3.3: An example of grayscale standardization for CL image.



(a) The original non-CL image

(b) Standardized image

Figure 3.4: An example of grayscale standardization for non-CL image.

Table 3.1: Function Set

Function Set	Description
$+$	Arithmetic addition
$-$	Arithmetic subtraction
$*$	Arithmetic multiplication
$/$	Arithmetic division
&&	Logical and operation
$\|\|$	Logical or operation
$if(arg1)then(arg2)else(arg3)$	Conditional. If arg1 is true then return arg2, otherwise return arg3
$>$	True if arg1 is greater than arg2
$<$	True if arg1 is lesser than arg2
exp	The exponential operation
log	The natural logarithm
pow	The power operation
max	Return bigger one of two values
min	Return smaller one of two values
abs	Return absolute value of a number
$sqrt$	Return square root of a number

Because the rotation invariant LBP histograms are computed from 32×32 subimages in our experiment, the possible largest bin values in the LBP_{16}^{riu2} histogram is 1024. Accordingly, the upper boundary for constant numbers is set to be 1024 which is large enough for comparison even though very large bins exist in the histogram. RILH(i) is used to denote the i-th bin value in the histogram of rotationally invariant LBP values.

The fitness function in this study was defined as the number of misclassifications. Since the training process is supervised learning, the number of misclassifications from generated program trees can be determined directly by comparing the program output with the actual class labels. Therefore, the GP process should generate a program tree to minimize the fitness function.

Highly fit programs were learned from the texture descriptions of small subimages sampled from both CL and non-CL images from our training set. Since CL texture was to be classified against the others, one hundred subimages were sampled from each CL area of each CL image to form Class 1, and two hundred subimages were sampled from each non-CL image to form Class 2. Because the source of CL samples is only the CL texture while the source of non-CL samples is the non-CL image with various other textures, a larger

Table 3.2: Terminal Set.

Terminal Set	
$RILH(i)$	The i-th bin value of LBP_{16}^{riu2} histogram, i ∈ [0, 17].
Constant numbers in [0, 1024]	Floating-point number chosen randomly from [0, 1024]

number of non-CL samples was used for training to ensure diversity of samples in this class.

Tournament selection was used for choosing individuals for crossover and mutation operations. The tournament size was 4. The termination criteria for each run was either perfect classification (zero misclassifications), or after some number of generations, NoG, had been processed. The program in the final population, C, with the greatest fitness was the output classifier. An example of a highly fit program, which was used in our experiment described in Section 4.3, was given in Appendix A.

3.3 CL Segmentation Algorithm

In order to perform CL segmentation, a sliding window of size $n \times n$ pixels scans the image in raster order, with horizontal and vertical step sizes of δ_x and δ_y respectively. The subimage within the window is classified by the CL texture classifier C obtained from genetic programming. Each subimage pixel is then assigned the class label output from C. For step sizes where $\delta_x < n/2$ or $\delta_y < n/2$, successive window positions overlap, and pixels receive multiple labels. Final labels are determined by majority vote, as in [49]. The steps of this segmentation algorithm are described in Algorithm 1.

3.4 CL Detection Algorithm

The CL detection algorithm begins by running the segmentation algorithm (Algorithm 1). Subsequently a region property classifier, also trained using genetic programming, is applied to the binary image representing the segmentation and attempts to determine if the image contains a CL or not.

To learn the region property classifier C_{rp}, the images in the training set are segmented by the CL texture classifier C. A set of region properties are computed for each image's output region; these region properties are summarized in Table 3.3. The region property classifier is learned by GP with the same parameters used when learning C, as described in Section 3.2, except that the region properties are used instead of LBP_{16}^{riu2} histograms in the terminal set. The output of this region property classifier is true or false which indicates whether the segmented region is a CL or not. The process of this CL detection algorithm is given in Algorithm 2.

Input : A CL image I,

Step size for moving the sliding window δ_x and δ_y,

Sliding window size n,

A CL texture classifier C whose input is an $n \times n$ subimage and output is label "CL" or "non-CL", respectively indicating that the subimage is CL texture or non-CL texture learned by GP from training data.

Output: A binary image O_b in which foreground pixels correspond to the segmented CL region.

1. Slide an $n \times n$ window about the input image I in raster order with step sizes δ_x and δ_y. For each windowed subimage J:
   (a) Classify the texture J as "CL" or "non-CL" using C.
   (b) Add the output label of C to all pixels in J. A pixel may receive either label multiple times if windows overlap.
2. Determine the final label of each pixel by the majority vote among the labels assigned in step 1(b).
3. Construct a binary image I_b where

$$O_b(x,y) = \begin{cases} 1 & \text{if the final label of } I(x,y) \text{ is CL;} \\ 0 & \text{if the final label of } I(x,y) \text{ is non-CL.} \end{cases}$$

4. Remove all but the largest connected component of O_b.

Algorithm 1: CL Segmentation Algorithm

Table 3.3: Region properties computed for each candidate CL region.

Area	The number of pixels in the region.
Convex Area	The number of pixels in the convex hull of the region.
Equivalent Diameter	The diameter of a circle with the same area as the region.
Filled Area	A scalar specifying the number of on pixels in "FilledImage". A "FilledImage" is a binary image of the same size as the bounding box of the region. The pixels with value "1" in "FilledImage" correspond to the region, with all holes filled in.
Major Axis Length	The length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region.
Orientation	The angle (in degrees ranging from -90 to 90 degrees) between the the major axis and the horizontal.
Solidity	The proportion of the pixels in the region's convex hull that are also in the region.

Input : A bovine ultrasound ovarian image I,

A CL texture classifier C whose input is an $n \times n$ subimage and output is label "CL" or "non-CL", respectively indicating that the subimage is CL texture or non-CL texture learned by GP from training data,

A region property classifier C_{rp} whose input is a vector of region properties computed from segmented region of input image I and output is true or false which indicates whether the segmented region is a CL or not.

Output: True or False which indicates the presence or absence of CL in the input image

1. Segment the input image I using the Algorithm 1 with classifier C. The output binary image I_{bw} from this segmentation will be used in the following steps.

2. Compute the region properties in Table 3.3 for the binary image I_{bw} from step 1. Region properties were computed by using the algorithms within the MATLAB (The Mathworks, Inc., Natick, MA, USA) function `regionprops` in the Image Processing Toolbox.

3. Determine the presence or absence of a CL in I by classifying the resulting vector of region properties from step 2, using the region property classifier C_{rp}. As a result, I is a CL image if the output of C_{rp} is true, otherwise I is a non-CL image.

Algorithm 2: CL Detection Algorithm.

Chapter 4

Experiments

4.1 Bovine Image Dataset

Images were selected from a dataset obtained during a previous study of bovine ovaries collected at defined times of the estrous cycle [47]. Pairs of bovine ovaries were excised at three different time points during the estrous cycle: day 3 of wave 1 (during metoestrus, $n = 6$), day 6 wave 1 (early dioestrus, $n = 8$), and after the onset of pro-oestrus at ≥ 17 days (D$\geq$17, $n = 8$), 22 animals in total. Since the previous study [47] was also a related study on follicular characteristics, the days of ovariectomy were chosen on the basis of follicular wave status and satisfied the objectives of allowing the acquisition of corpora lutea.

Day 0 was the day of the last observed ovulation, "wave" refers to waves of follicle growth [47]. The ovaries were placed in degassed phosphate-buffered saline and imaged at $0.5mm$ increments in parallel planes with a broad-band, convex-array ultrasound transducer interfaced with an ATL Ultra Mark 9 HDI ultrasound machine (Advanced Technology Laboratories, Bothell, WA, USA). All of the images are 640×480-pixel grayscale images [13, 53]. For each animal, one or two images containing CL were selected from one ovary and the same number of images without CL were selected randomly from the other ovary to obtain a total of 60 images, 30 of which contain a CL.

4.2 Experiment on Images From Each Time Point

The first experiment was conducted on images from a single time point. The experiment was repeated three times for time points day 3, day 6 and day ≥ 17 respectively. Details of the whole experiment are given in the rest of this section.

4.2.1 Experimental Settings

Our detection and segmentation algorithms are evaluated on images from each time point in our dataset: day 3 of wave 1 (12 images), day 6 of wave 1 (32 images), day ≥ 17 (16 images), which resulted in 3 sub-experiments. There are equal number of CL and non-CL images from each time point. The images from each time point are split into training and test set by a half-and-half method, and each half includes equal number of CL and non-CL images. The overall performance of our algorithm on each time point was computed as

Table 4.1: Parameter Settings For The Experiment

Parameters	Descriptions
$n = 32$	The size of sliding window
$\delta_x = \delta_y = 2$	Step size for moving the window.
$h = 480, w = 640$	Height and width of images.
$P_c = 0.8,$	Probability of crossover.
$P_m = 0.1$	Probability of mutation.
$sizeP = 600$	Population size.
$D = 20$	Maximum depth of program trees.
$NoG = 500$	Number of generations for GP runs

the average performance of the two halves.

The algorithm parameters used are shown in Table 4.1. In [49], a sliding window size of 16 was used to perform segmentation on 256×256 images containing Brodatz textures. However, the image size in our experiment is 640×480 and a 16×16 sliding window is too small to characterize the local texture features, so the window size was set to be 32×32. The typical ranges for crossover and mutation probabilities are $[0.7, 1)$ and $[0.001, 0.1]$, respectively [35]. We choose a relatively large mutation probability (0.1) to avoid getting trapped in a local optimum.

In order to justify the decision of choosing 500 as the maximum number of generations, a GP performance graph from our experiment is given in Figure 4.1. Since the fitness in our experiment is the number of misclassifications, but we want to plot an ascending curve for GP performance, so *Performance* is used for y axis and defined as follows:

$$Performance = 1 - \frac{Fitness}{Total\ Number\ of\ Training\ Samples} \tag{4.1}$$

where Fitness is the number of misclassifications.

In Figure 4.1, the performance curve rises steeply at the beginning of the GP run, but stops after around generation 500 and little further benefit is gained by computing additional generations suggesting that the fitness is at or very near the global optimum.

4.2.2 Performance Measures

In order to evaluate the GP training results, the training accuracy is defined as the fraction of correct classifications of the training samples (the subimages used for training) by the learned classifier.

The accuracy for the CL detection algorithm is given by the number of correct decisions over the number of testing images.

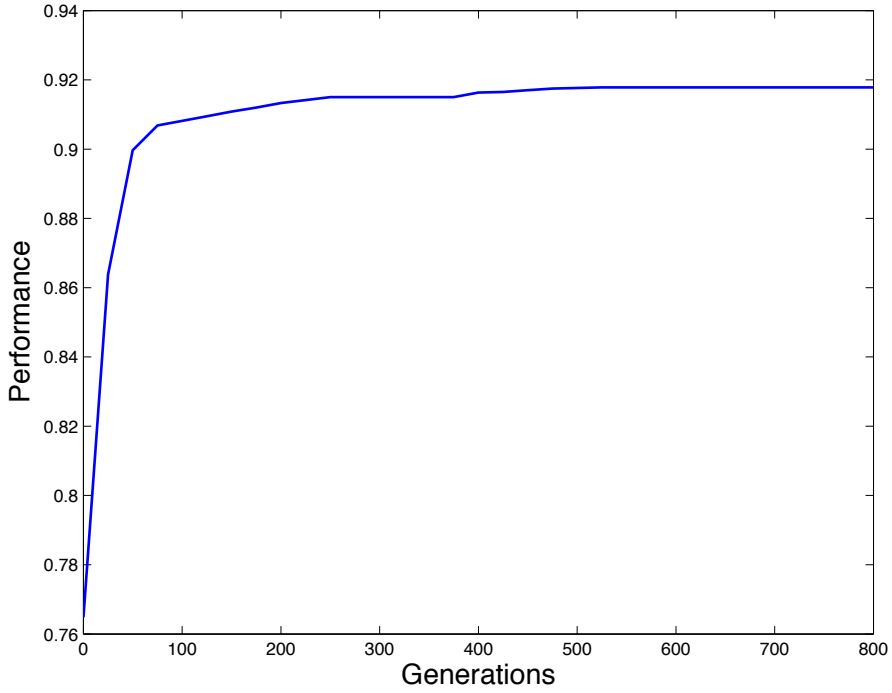


Figure 4.1: GP Performance Graph. X axis is the number of generations, Y axis is the *Performance* value of best-so-far classifier.

For the segmentation algorithm, the manual segmentations obtained for this study are used as the basis for comparison when validating the automatic segmentations. Let TP denote the set of true positive pixels, that is, those pixels that are correctly classified as belonging to a CL region. Let TN denote the set of true negative pixels correctly identified as not belonging to a CL region. Similarly define FP and FN to be the set of false positive and false negative pixels. The sensitivity and specificity of a segmentation are defined as follows:

$$\text{sensitivity} = \frac{|TP|}{|TP| + |FN|}, \tag{4.2}$$

$$\text{specificity} = \frac{|TN|}{|TN| + |FP|}, \tag{4.3}$$

where $|\cdot|$ represents the number of elements in a set.

The sensitivity represents the proportion of CL texture pixels which were correctly identified as such. Specificity represents the percentage of pixels which are not part of the CL region which were correctly identified as such. A perfect segmentation with respect to the ground truth will have both a sensitivity and a specificity of 1.0.

4.2.3 CL Detection and Segmentation Results for Each Time Point

The results of CL detection experiment are shown in Table 4.2 which includes the training accuracy the CL detection accuracy for each time point. Our detection algorithm works well on each time point with accuracy above 90%.

Table 4.2: CL Detection Performances For Each Time Point

Time point	Training accuracy for CL classifiers	CL detection accuracy
Day 3	97.8%	100.0%
Day 6	95.1%	93.8%
Day 17	94.9%	93.8%

The results of CL segmentation for each time point are shown in Table 4.3. The performance of our segmentation algorithm varied a little on different time points. Mean specificity is very high for both CL and non-CL images from each time point, while mean sensitivity for CL images differentiate from each other a little but the overall accuracy is still high. Sensitivity is not reported for non-CL images since $|TP| + |FN|$ is always 0 in such images making sensitivity undefined. In order to test whether the results on different time points are significantly different, a Welch's t-test was used on each pair of segmentation results.

Table 4.3: CL Segmentation Performances For Each Time Point

Time point	Mean sensitivity (CL image)	Mean specificity (CL image)	Mean specificity (non-CL image)
Day 3	0.92 ± 0.09	0.90 ± 0.04	0.99 ± 0.01
Day 6	0.85 ± 0.18	0.93 ± 0.05	0.98 ± 0.03
Day 17	0.82 ± 0.14	0.95 ± 0.05	0.98 ± 0.02

Firstly, a Lilliefors test was performed to test whether the distribution of each time point's segmentation results was from a normally distributed population. As a result, the segmentation results (sensitivity and specificity) on CL images passed the normality test, while the results (specificity) on non-CL images did not because the specificity for every segmented non-CL image was larger than 0.95. Thus, the t-test was only used to test the segmentation results of CL images.

The formula for the Welch's t-test is given in Equation 4.4 where M_i, V_i and N_i are sample mean, sample variance and sample size of the i^{th} distribution. The degrees of freedom (df) was determined for the t-test using Equation 4.5. Once the values of t and df were computed, we have to look up in the table of significance to find the critical t-value and decide whether the ratio obtained from Equation 4.4 is large enough to say that the means are significantly different. To find the significance, a risk level called the alpha level has to

be set. In most researches, the "rule of thumb" is to set the alpha level at 0.05 and this is also the value we chose for our t-test.

$$\mathrm{t} = \frac{M_1 - M_2}{\sqrt{\frac{V_1}{N_1} + \frac{V_2}{N_2}}} \tag{4.4}$$

$$\mathrm{df} = \frac{(\frac{V_1}{N_1} + \frac{V_2}{N_2})^2}{\frac{(\frac{V_1}{N_1})^2}{N_1 - 1} + \frac{(\frac{V_2}{N_2})^2}{N_2 - 1}} \tag{4.5}$$

The results of this t-test are given in Table 4.4 and 4.5 which correspond to the t-test of sensitivity and specificity on segmented CL images respectively. As can be seen from these two tables, all the computed t-values are smaller than the critical t-value obtained from the table of significance, so the segmentation results on CL images from different days are not significantly different.

Table 4.4: T-test Results for Sensitivity of Segmented CL Images from Each Pair of Time points.

Days for Testing	t-value	Degrees of Freedom (df)	Critical t-value
Day 3 and Day 6	1.14	18	2.10
Day 6 and Day $\geq$ 17	0.53	18	2.10
Day 3 and Day $\geq$ 17	1.65	12	2.18

Table 4.5: T-test Results for Specificity of Segmented CL Images from Each Pair of Time points.

Days for Testing	t-value	Degrees of Freedom (df)	Critical t-value
Day 3 and Day 6	1.18	10	2.23
Day 6 and Day $\geq$ 17	0.96	14	2.15
Day 3 and Day $\geq$ 17	1.84	12	2.18

Since good results were achieved from experiment on each time point, a more challenging experiment was conducted on images mixed together from all the time points. Details about this experiment are included in the following section.

4.3 Experiment on Images Mixed Together from Different Time Points

4.3.1 Experimental Settings

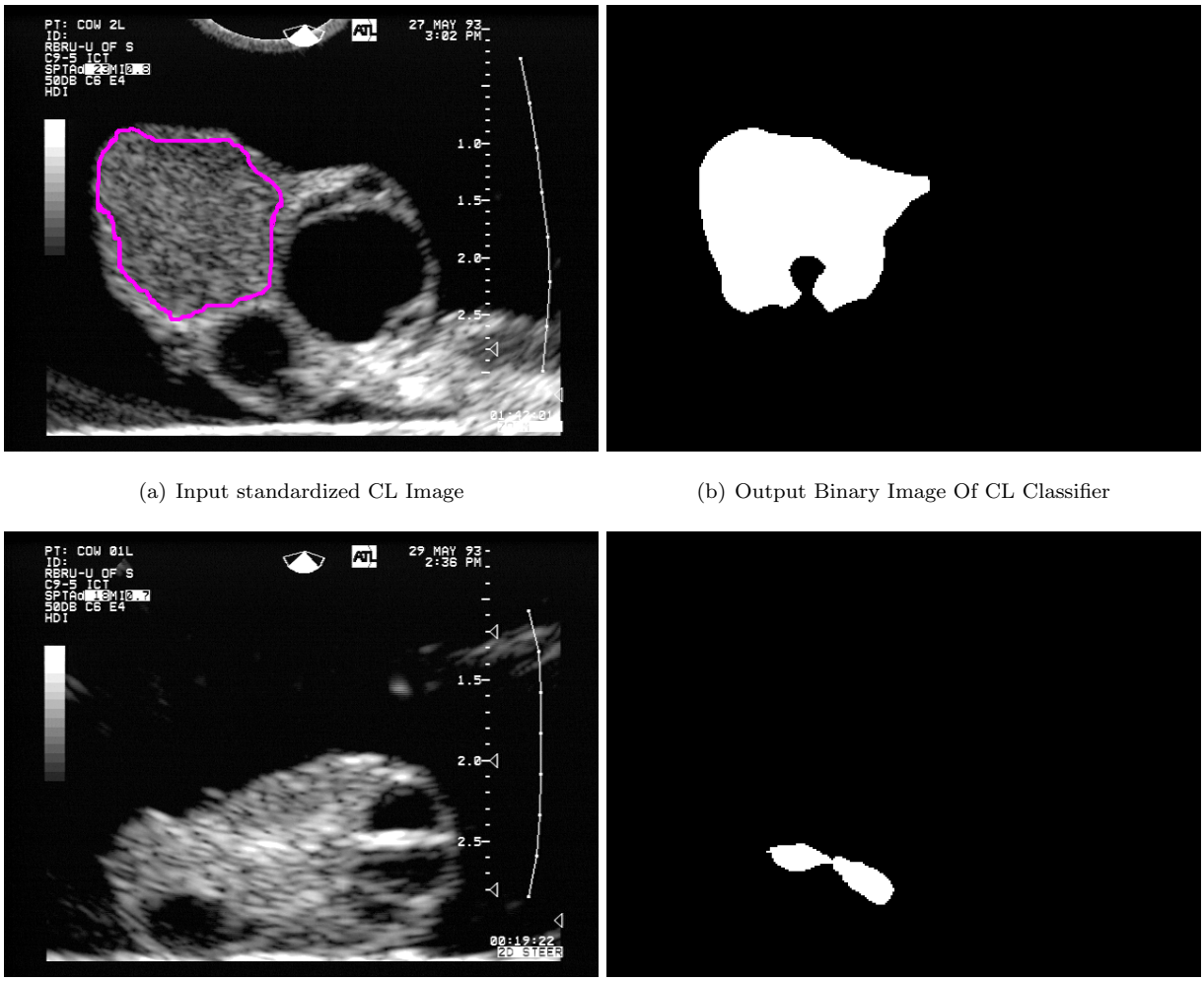
The experimental settings were the same as described in Section 4.2.1, except that all the images from different time points were mixed together and a cross-validation methodology was performed to evaluate the performance of our segmentation and detection algorithms.

An N-fold cross-validation is a generalization of the leave-one-out methodology. A ratio of testing to training data was first determined based on the value of N. For example, if N is equal to 10, 90% of the data will be used for training and 10% for testing. Then N different random splits of the data are created based on this ratio. The N test set must be disjoint but the training sets need not be. The cross-validation process is repeated N times (the folds), with each of the N disjoint test sets used only once. The overall performance is the average of the N tests.

A three-way cross-validation was used in our experiment. Therefore, the whole dataset, including 30 CL and 30 non-CL images, was split into three disjoint subsets randomly, each with 20 images. In each subset, there were an equal number (10) of CL images and non-CL images. Moreover, each subset has same number of images from each estrous cycle time point. The overall performance was computed as the average performance of the three data folds. The algorithm parameters used were the same as shown in Table 4.1. The performance measures are also the same as given in 4.2.2.

4.4 CL Detection Results

The results of the CL detection experiment are shown in Table 4.6 which, for each fold of the cross-validation, includes the training accuracy and the CL detection accuracy. The average accuracy of final decisions over all three folds was 93.3%. For most of the images with a CL, the CL classifier is able to find the region of CL, although there may be a few false positive and false negative pixels. For most of the images without a CL, the CL classifier just claims some isolated small regions which are false positive. An example is shown in Figure 4.2. The ultrasound image after standardization is shown in Figure 4.2(a). Figure 4.2(b) shows the output binary image of the largest connected region claimed by CL classifier for Figure 4.2(a). An image without CL and its output binary image with largest connected region claimed by CL classifier are displayed in Figure 4.2(c) and Figure 4.2(d). The white area in Figure 4.2(d) is the CL claimed by the classifier.



(a) Input standardized CL Image

(b) Output Binary Image Of CL Classifier

(c) Input standardized No-CL Image

(d) Output Binary Image Of CL Classifier

Figure 4.2: Examples of CL Classifier's Binary Output for CL and No-CL Images

Table 4.6: CL Detection Performances

Number of test	Training accuracy for CL classifiers	CL detection accuracy
1	93.7%	85.0%
2	93.4%	100.0%
3	91.8%	95.0%
Average	93.0%	93.3%

4.5 CL Segmentation Results

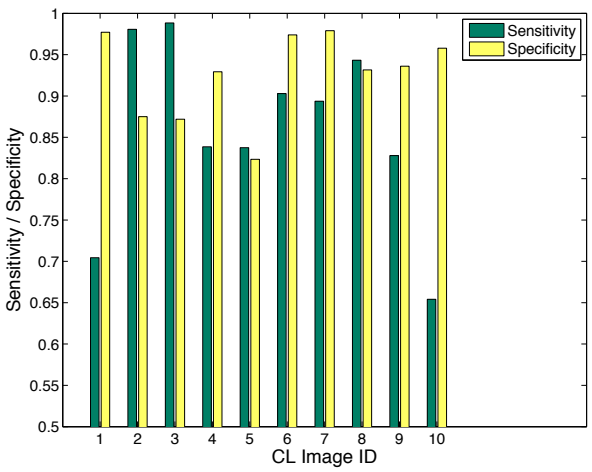
The results of CL segmentation for the CL test images in each data fold are shown in Figure 4.3. The mean ($\pm$ standard deviation) sensitivity and specificity for all the 30 CL images are 0.87 ± 0.14 and 0.91 ± 0.05. Figures 4.5, 4.6, 4.7, 4.8 and 4.9 give some examples of the output segmented CL regions. In each figure, the pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marked the boundary of central cavity. It can be seen from the figures that this method is able to find the CL regions in the CL images. Even though there is a central cavity inside the CL texture, the generated CL classifier can still find the CL regions in the image.

The results of the segmentation algorithm for the non-CL images in each data fold are shown in Figure 4.4. Since there is no CL region in non-CL images, the ground truth of the CL segmentation algorithm should be a black image with no foreground pixels (TP and FN). Thus, only specificity is shown in Figure 4.4. The mean ($\pm$ standard deviation) specificity for all the 30 non-CL images is 0.97 ± 0.04. Most of segmented non-CL images only include small false positive foreground regions as in Figure 4.2(d).

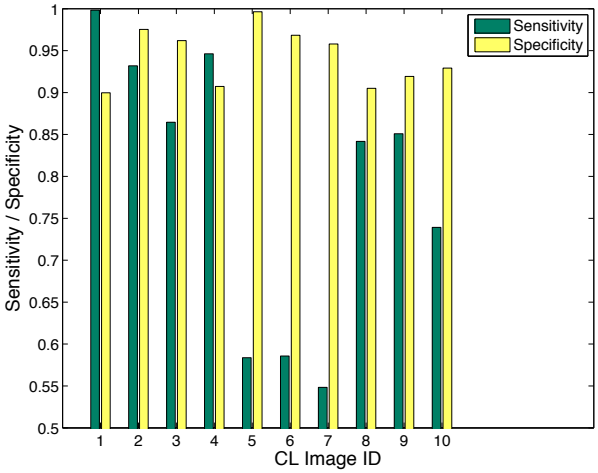
4.6 Discussion

From the results of our CL detection experiments, it is clear that the proposed method works well on the bovine CL dataset including images from different time points of the estrous cycle. The detection accuracy on images from each time point was higher than 90%. After mixing all the images together, the average detection accuracy was still as high as 93.3%. Therefore, these two experiments proved the generality of the proposed CL detection algorithm on our bovine dataset.

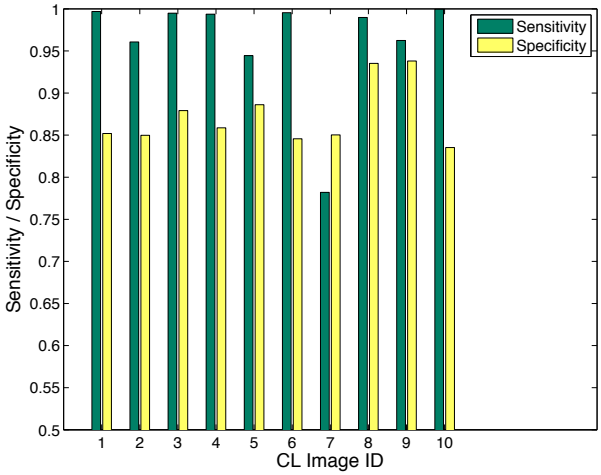
The segmentation experiment on images from each time point achieved high accuracy shown in Table 4.3. The mean specificity was higher than 0.90 for both CL and non-CL images from every separate time point. The sensitivity values for Day 3 and Day 6 were higher than 0.85 but the value on images from Day ≥ 17 was lower (0.82). However, the results of Welch's t-test indicated that the segmentation results on CL images from different days are not significantly different. The reason for the lower sensitivity from Day ≥ 17 is that luteal gland diameter (measured ultrasonographically) decreased with regression during pro-oestrus



(a) Segmentation results for the first fold

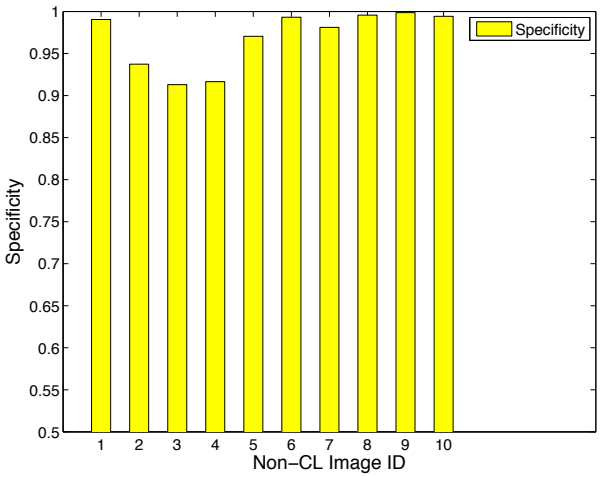


(b) Segmentation results for the second fold

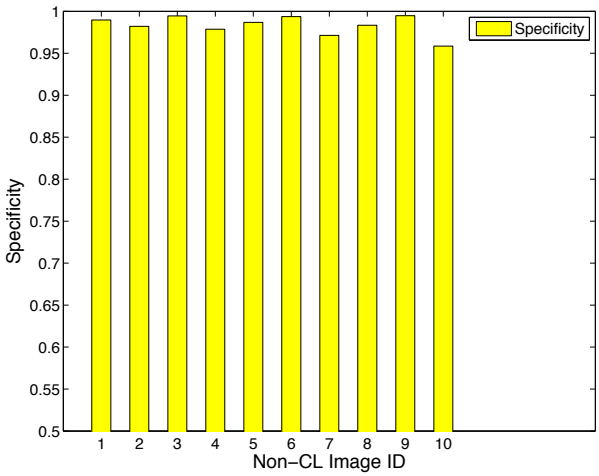


(c) Segmentation results for the third fold

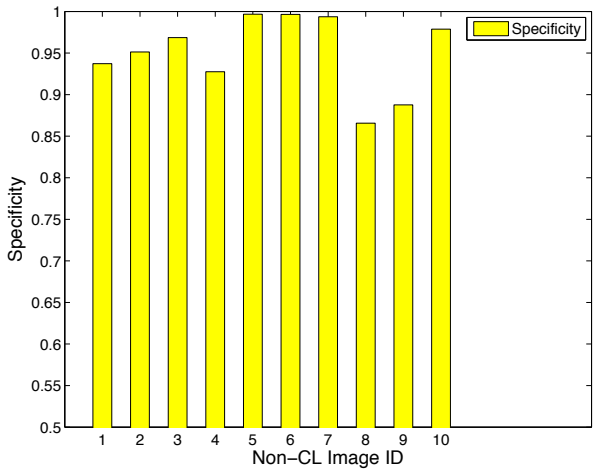
Figure 4.3: CL segmentation performances for the CL images in each data fold. For each image, the green bar indicates the sensitivity and the yellow bar indicates the specificity.



(a) Segmentation results for the first fold

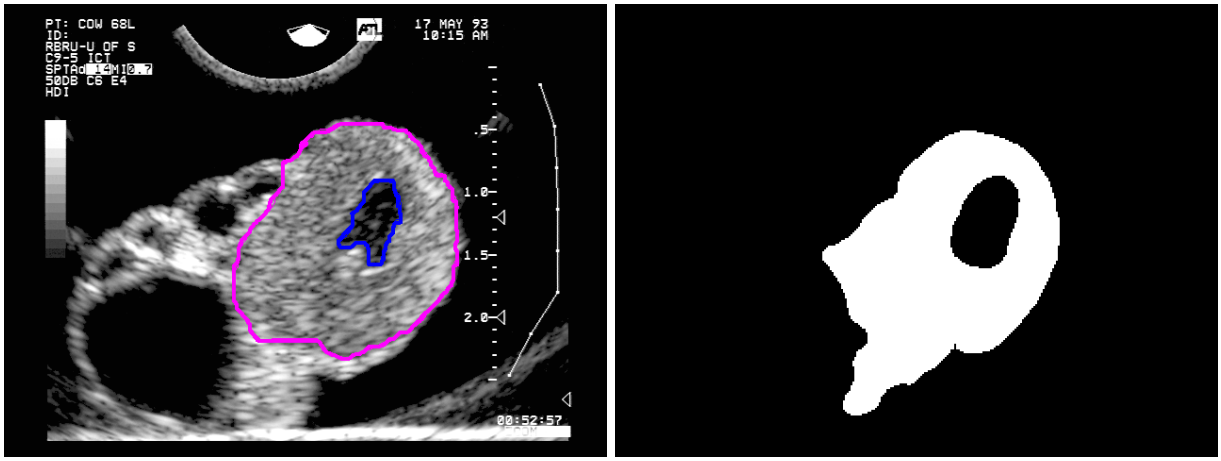


(b) Segmentation results for the second fold



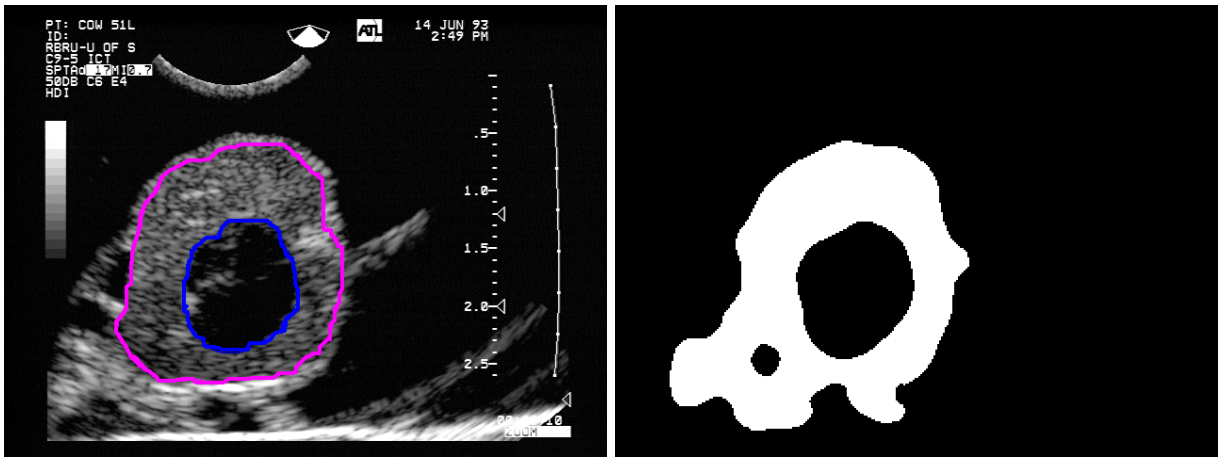
(c) Segmentation results for the third fold

Figure 4.4: CL segmentation performances for the non-CL images in each data fold. Only the specificity values are shown by the yellow bars.



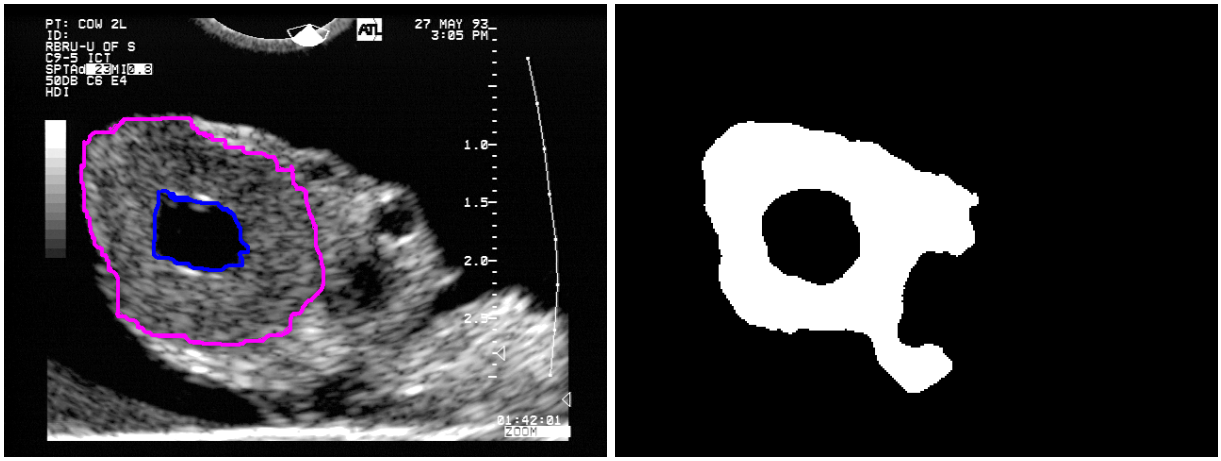
(a) CL Image (b) Output Binary Image

Figure 4.5: An Example of CL Segmentation Results. The pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marks the boundary of central cavity.



(a) CL Image (b) Output Binary Image

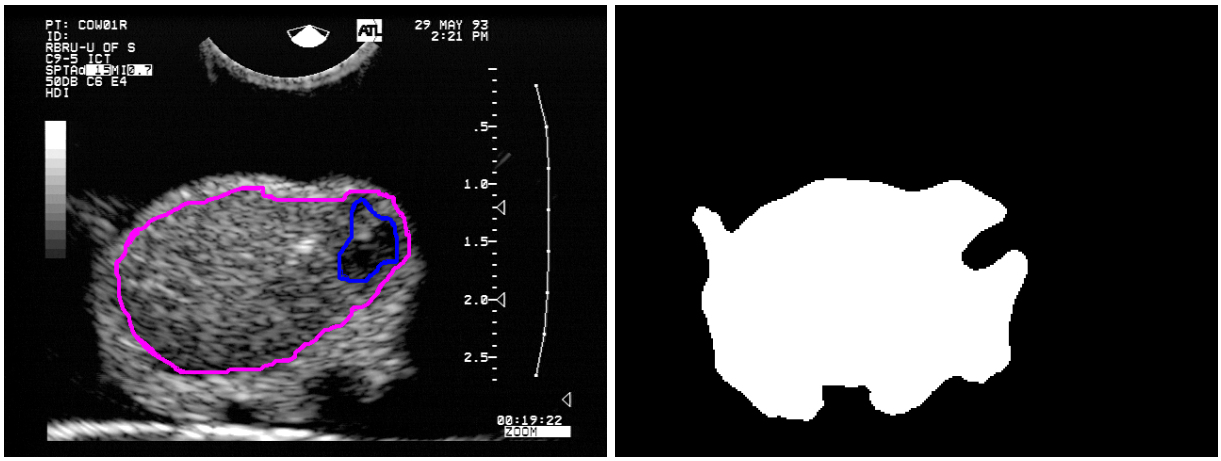
Figure 4.6: An Example of CL Segmentation Results. The pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marks the boundary of central cavity.



(a) CL Image

(b) Output Binary Image

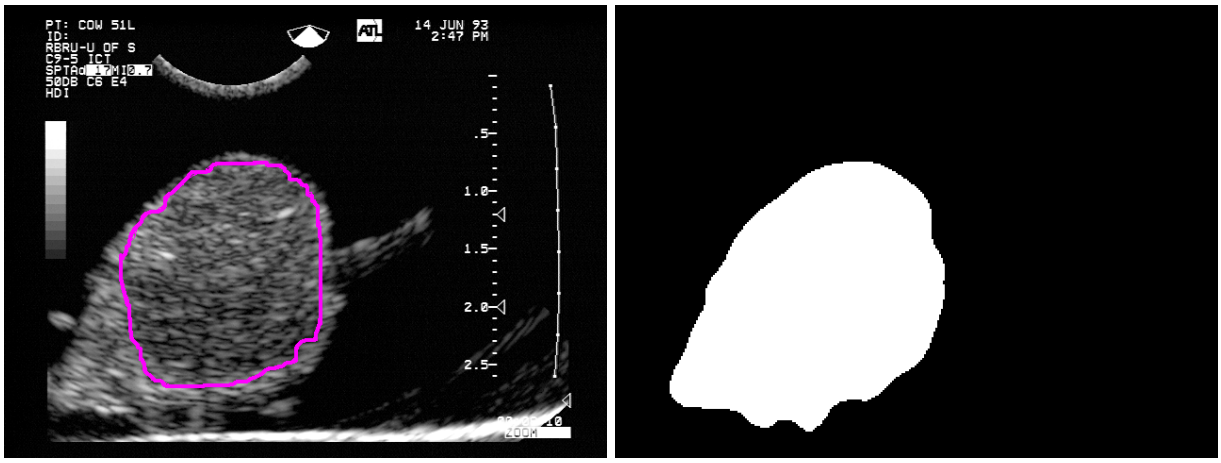
Figure 4.7: An Example of CL Segmentation Results. The pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marks the boundary of central cavity.



(a) CL Image

(b) Output Binary Image

Figure 4.8: An Example of CL Segmentation Results. The pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marks the boundary of central cavity.



(a) CL Image

(b) Output Binary Image

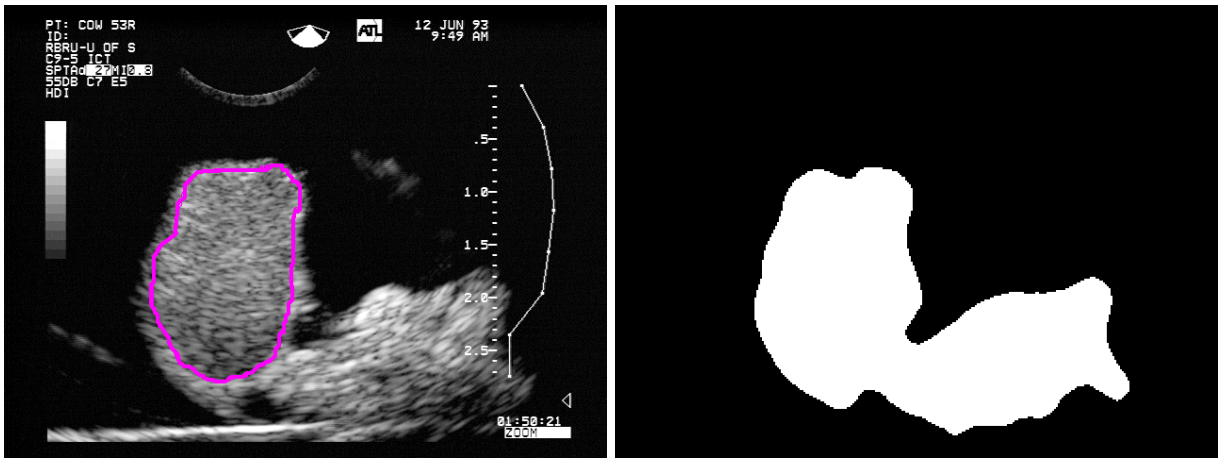
Figure 4.9: An Example of CL Segmentation Results. The pink contour in the CL image indicates the boundary of CL which is used as the ground truth, and the blue contour marks the boundary of central cavity.

(Day $\geq$ 17). The regressing CL decreased in mass, in volume densities of luteal cells and blood vascular components, and in progesterone production, while there was a concurrent increase in connective tissue and hyalinization of blood vessels [47]. Therefore, the CL regions became harder to segment from surrounding textures due to the changes of this regression process.

The second segmentation experiment on all the images from 3 time points also achieved a high accuracy as shown in Figure 4.3 and 4.4. Compared with the previous work using level set segmentation in [43], our CL segmentation method is totally automatic and does not require placement of an initial contour. The mean ($\pm$ standard deviation) sensitivity and specificity of the CL segmentations produced by their algorithm were 0.81 ± 0.17 and 0.99 ± 0.01 over 8 images, while we achieved a mean sensitivity and specificity of 0.87 ± 0.14 and 0.91 ± 0.05 over 30 images from different time points in the oestrous cycle.

Their algorithm of [43] employs a segmentation algorithm based on level-set methods. It tends to under-segment the CL, that is, specificity is generally excellent while sensitivity is much lower and varies from very good to quite poor. This is due to the fact that an initial contour is placed manually within the CL and then evolved outward under an expansive force. The level set segmentation algorithm halted contour evolution too soon in many instances, which produces a large number of FN pixels and few FP pixels. In contrast, our algorithm obtained a higher sensitivity with lower standard deviation. Although our mean specificity is lower, but it is still high, at 0.91 ± 0.05 and our algorithm achieves this without prior knowledge of the location of the CL within the image. Using our approach, we can automatically determine the presence of a CL with an accuracy of 93.3%, and achieve a segmentation with the aforementioned mean accuracy.

Comparing runtime, we find that the mean runtime reported for segmentation of a single image using the

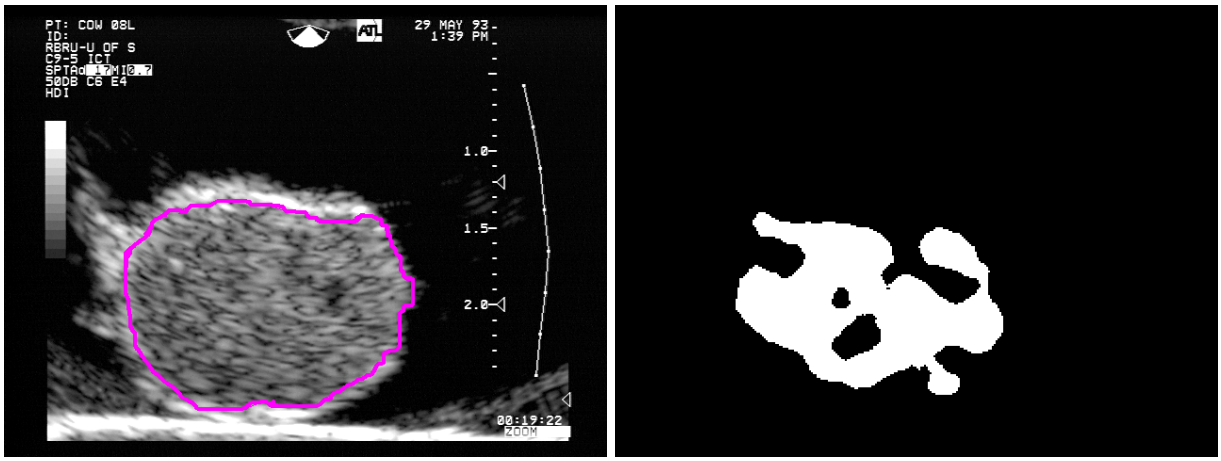


(a) Input image – magenta contour denotes CL boundary.

(b) Output binary image.

Figure 4.10: A segmentation result with many false positive pixels.

algorithm of [43] was 5min 54s on an 2.8GHz Pentium 4 Processor with 2 GB of RAM running Mandriva Linux, using an implementation which was accomplished by using MATLAB (The Mathworks Inc, Natick, MA, USA) version 7.4.0.336 [43]. The mean runtime for segmenting one image using our method was only 61 seconds using a 2.66GHz Intel Core 2 Duo Processor with 2GB of RAM running Mac OS X (Version 10.6.4) implemented in MATLAB version 7.7.0.471 (R2008b). However, this time does not include the one-time cost of the GP training process for learning the classifier. This was implemented using the Java Genetic Algorithm Package (JGAP), which took around 110 minutes to finish on the same Mac machine.



(a) Input image – magenta contour denotes CL boundary.

(b) Output binary image.

Figure 4.11: A segmentation result with many false negative pixels.

Although the CL detection and segmentation results are very good, there are still some limitations of this method. For images of CL obtained only a few days after ovulation, our method does not work well because the area of echotexture resulting from luteal tissue is fairly small. Since the size of our sliding window has to be large enough to capture the local patterns of CL texture, there are often few windows in such images that contain only CL texture. Moreover, some segmentation results exhibit many FP and FN pixels. Figure 4.10 shows an example of segmentation result with a large number of FP pixels and Figure 4.11 shows an example of a segmentation result which contains many FN pixels. In addition, an alternate choice of fitness function for the GP process might be the precision ($TP/(TP+FP)$) because this measures the chance that a point in the segmented region is actually part of the object of interest.

The objective of producing a high-performance, fully automated CL detection algorithm was met; though we do not know of an algorithm to which we can directly compare our algorithm's 93.3% decision accuracy. Our segmentation algorithm is automatic, requires no initial contour, achieves a high accuracy over images from different time points and has a faster speed, compared with the only previous CL segmentation [43]. Based on the proposed method in this thesis, LBP or some other statistical texture features, which may be capable of distinguishing normal tissues from abnormal ones, can also be experimented with GP in the future.

After getting successful results from the experiment on the bovine dataset, another experiment to perform human CL detection and segmentation on images of *in vivo* ovaries were conducted, which is more challenging than our bovine CL images which were acquired *in vitro* under ideal imaging conditions. The details are described in the next section.

4.7 Experiment On Human CL Images

Due to the ethical and logistical limitations in human subjects, humans possess few of the characteristics of a good model. However, the bovine model for research on ovarian dynamics in humans was well established [2]. The bovine ultrasound model is the most well developed model with regard to characteristics of ovarian follicular and luteal dynamics. The size and morphology of the ovaries in cows and women are similar.

The ovaries of each species are approximately $3 \times 2 \times 1.5$ cm, mature follicles are 15 to 20 mm in diameter, and the mature CL is approximately 2.5 to 3.0 cm in diameter. In both species, the cortical region of the ovary (containing the follicles) is outermost and ovulation can occur at any point over the ovarian surface; both species are monovular and polycyclic. Similar pathologic conditions occur in both species such as follicular cysts, luteinized anovulatory follicles, and lactation- or stress-related suppression of follicle growth and ovulation. Therefore, the following experiment was performed on human ovarian ultrasound images using similar methodology as on the bovine dataset.

4.7.1 Human CL Detection and Segmentation Using The Same Algorithm

The human CL dataset used here included images of *in vivo* human ovaries from three different time points: day 3 ($n = 20$), day 6 ($n = 40$) and day 10 ($n = 16$), where n is the number of the images. The images were collected from the Royal University Hospital at the University of Saskatchewan, and some low-quality images which were not appropriate for this study were eliminated from the dataset by an expert. For each time point, there are equal number of CL images and non-CL images. All of the images are also grayscale images with a size of 640×480. All the images were also standardized before any further processing, using the algorithm described in Section 3.1.

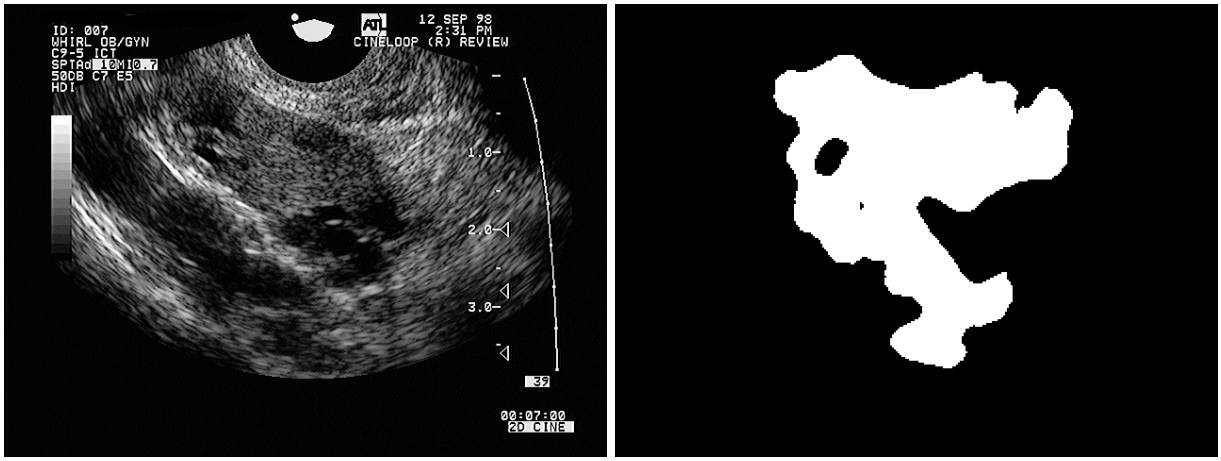
The first experiment on human CL was conducted using the same detection and segmentation algorithms as in Section 3.4 and 3.3. We performed the experiment on images from each time point. There are equal number of CL and non-CL images from each time point. The images from each time point were split into training and test set by a half-and-half method, and each half includes equal number of CL and non-CL images. One half was used as training set and the other half was tested, and vice versa. The overall performance of our algorithm on each time point was computed as the average performance of the two halves.

The parameter settings are the same as in Section 4.2.1. The overall detection accuracy, which was computed as the mean accuracy over all the three time points, was only 41.7%. There were usually a lot of FP in the binary result of non-CL image, which confused the region property classifier. Figure 4.12 (a) displays one non-CL image and the output binary image by CL classifier is shown in Figure 4.12 (b) where a large region of FP exists. The segmentation results were a mean ($\pm$ standard deviation) sensitivity and specificity of 0.77 ± 0.24 and 0.88 ± 0.10. The sensitivity is fairly low with a large variance, which indicates that many CLs are not located precisely in the image. The specificity is acceptable. However, since there are usually intensive "other" textures in *in vivo* human CL images and the CL region is fairly small, the absolute number of FP pixels in the segmentation result is still fairly large. Figure 4.13 gave one segmentation result for human CL. The true CL region was not properly located and a large FP region also exists.

4.7.2 Human CL Detection Without Segmentation

A second experiment on human CL detection was performed on our human CL dataset without making use of the binary segments from CL texture classifier. The images from each time point were still split into training and test set by a half-and-half scheme as in Section 4.7.1.

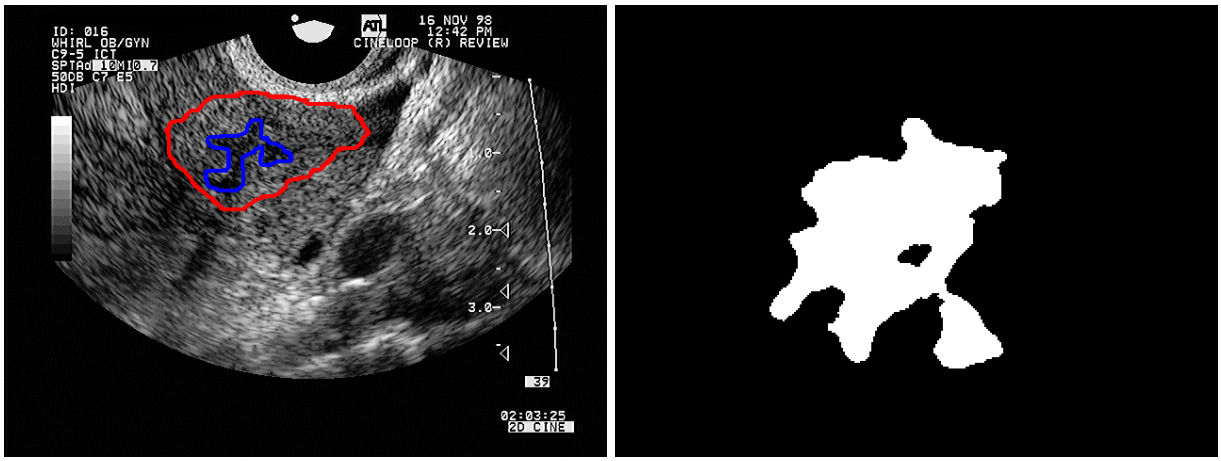
For every human *in vivo* image in our dataset, only one histogram was built for the texture part of the image, which means that we need to clip the margins around the image boundaries and only preserve the "middle" texture part. Thus, the top, bottom, left and right sides are removed and only the middle 480×280 pixels were preserved for each image in our dataset. The choice of the middle 480×280 pixels can make sure that the margins are removed and the CL region can be preserved for every CL image in our dataset. A clipping example is given in Figure 4.14.



(a) A non-CL image of human ovary

(b) The output binary image

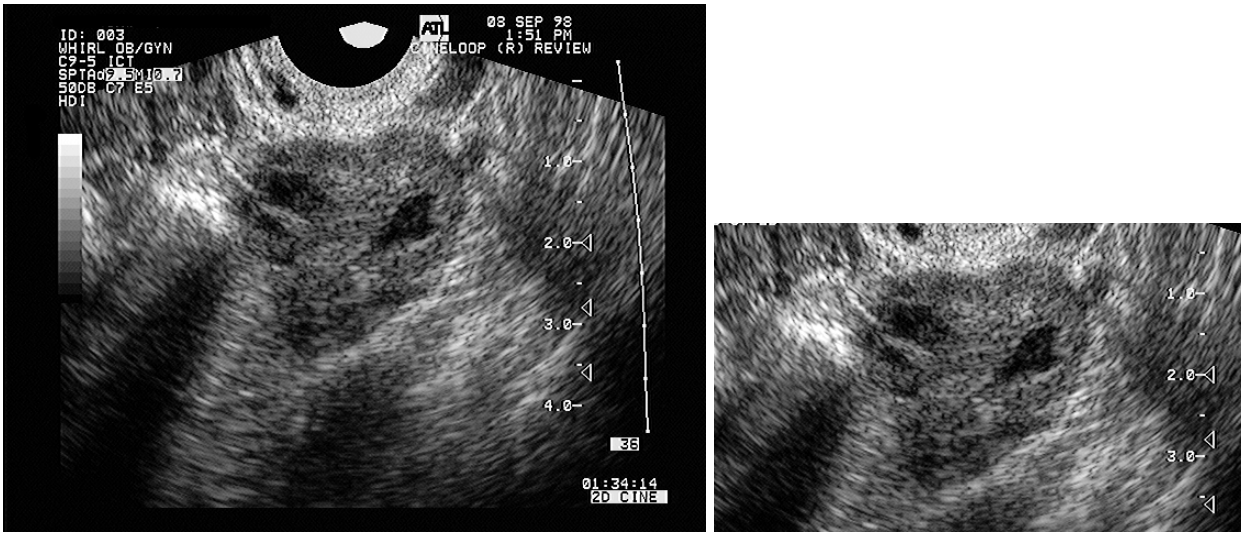
Figure 4.12: An output binary result of human non-CL image with many FP pixels.



(a) A non-CL image of human ovary

(b) The output binary image

Figure 4.13: An example of human CL segmentaion. The ground truth CL region is the area between red and blue contour in the original image.



(a) A human CL image

(b) Clipped image with only the middle texture part

Figure 4.14: An example of clipped human CL image

Afterwards, only one histogram was built for a whole clipped image. In order to make the histogram informative, we build the histogram of all the rotation invariant LBP instead of only the uniform patterns (9 uniform patterns for 8-bit LBP and 17 uniform patterns for 16-bit LBP). However, since there are more than 4000 rotation invariant patterns for 16-bit LBP, which will make the histogram too sparse, we choose to compute the histogram of all the 8-bit rotation invariant LBP and thus the histogram will have 36 bins.

A classifier was evolved by GP using the histograms of clipped training images. The function set is the same as in Table 3.1 and the terminal set consists of only the input histogram bin values. The other parameters for GP training process are the same in the previous section. Then the classifier will be used on the test set. The input is the histogram with all the 8-bit rotation invariant LBP for the clipped test image, while the output is 0 or 1 indicating whether there is a CL in the test image. However, the overall detection accuracy is only 54.58%.

4.7.3 Discussions

Our method worked well on the bovine dataset but was not successful on the human CL dataset. This is because the bovine dataset was imaged *in vitro* under ideal imaging conditions while the human images were imaged *in vivo*. Thus, the outlines of bovine CL are more obvious than human CL. Moreover, due to the larger field of view in the human ultrasound ovarian images, the size of human CL is usually much smaller and there are more surrounding textures present in the images, so it is difficult to detect CLs because we need to use a sliding window with a fairly large size (32×32) to sample homogeneous CL texture.

For the human CL detection experiment described in Section 4.7.2, the reason for the low accuracy is that

many different kinds of textures exist in the clipped image, which made the histogram represent a mixture of textures. Therefore, when we plot the curve for the 8-bit rotation invariant LBP histograms for clipped images, all the curves, for both CL and non-CL images, have very similar shapes like in Figure 4.15, so the classifier can not distinguish the CL and non-CL images accurately. Moreover, blood flow exists and changes over time in human *in vivo* images, which is not a factor in *in vitro* images. During the human menstrual cycle, a blood loss happens during the menstruation phase and the velocity of blood flow is high during the follicular phase. Then the blood flow will slow down and only some sudden changes in hormones at the time of ovulation will cause light mid-cycle blood flow. Afterwards, during the luteal phase, the velocity of blood flow in the thyroid becomes even lower [29]. All these blood flow changes will affect the appearance of human *in vivo* ovarian images in brightness. Thus, future work to develop a robust *in vivo* CL detection and segmentation method should deal with the issue of changing blood flow over time. Some image preprocessing techniques may be used to remove the effect of blood flows and make CL more distinct from the surrounding textures, or even eliminate some other textures in the preprocessing step. Based on the proposed method in this thesis, some other statistical texture features, which may be capable of distinguishing normal textures from abnormal textures, can also be experimented with GP in the future.

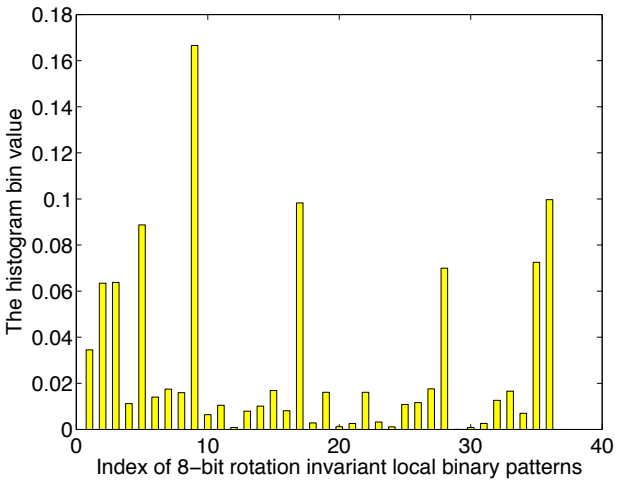


Figure 4.15: An example of the histogram plot for clipped image.

Chapter 5

Conclusions

5.1 Conclusion

This project proposed an automatic method to perform CL detection and segmentation, based on the classifiers trained by genetic programming. Rotation invariant local binary patterns were computed to encode local texture features which were used to evolve the classifiers. The performances of detection and segmentation algorithms were evaluated on a bovine dataset consisting of 60 ultrasonographic images of ovaries imaged *in vitro* from different time points in the estrous cycle. The CL detection algorithm was robust with an overall accuracy of 93.3%. Our CL segmentation algorithm was also successful with a high accuracy. The mean (standard deviation) sensitivity and specificity were 0.87 ± 0.14 and 0.91 ± 0.05. Even though there is a central cavity inside the CL texture, the generated CL classifier can still find the CL regions in the image. The algorithms can assist with monitoring the development of CL over time, facilitating the interpretation of the ovarian ultrasonography and the diagnosis of ovarian diseases. In conclusion, the contribution of this thesis is summarized as follows:

1. The decision can be made about the presence or absence of CL and an automatic detection of evidence of ovulation can be achieved, so the ultrasonographic ovarian images can be classified into CL and non-CL images accurately without any prior knowledge.

2. The CL regions can be obtained from the CL images using the proposed automatic CL segmentation algorithm. No initial placement of contour or points are required.

3. The algorithms were tested on a dataset containing 60 images from different time points during the estrous cycle. Our algorithms were proved to work well on not only the images from the same time point, but also all the images mixed together from different time points.

5.2 Potential Future Work

Some potential future extensions of this work are listed as follows:

1. Develop a method based on this work to perform human CL detection and segmentation on images of *in vivo* ovaries. Some more image pre-processing techniques may be explored to make the images more

distinct or eliminate some other textures in the pre-processing step.

2. Some other statistical texture features can be experimented with the combination of GP to detect and segment human CL, such as some higher-order statistical features like run-length matrix.

3. The method proposed in this work has potential applications on other kinds of texture analysis problems. Rotation invariant LBP may be used to encode local neighborhood in other texture analysis problems like ovarian follicles. The texture classifier learned by GP, based on the chosen texture features which are application dependent, may achieve good performance on classification and segmentation of other kinds of textures.

References

[1] G. P. Adams, R. L. Matteri, J. P. Kastelic, J. C. H. Ko, and O. J. Ginther. Association between surges of follicle-stimulating hormone and the emergence of follicular waves in heifers. *Reproduction and Fertility*, 94:177–188, 1992.

[2] G. P. Adams and R. A. Pierson. Bovine model for study of ovarian follicular dynamics in humans. *Theriogenology*, 43(1):113–120, 1995.

[3] Waqas Ahmed and M. G. Eramian. Automated detection of grayscale bar and distance scale in ultrasound images. In Benoit M. Dawant and David R. Haynor, editors, *Proc. of SPIE*, volume 7623, 2010.

[4] T. Ahonen, A. Hadid, and M. Pietikainen. Face recogniton with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[5] A. R. Baerwald and R. A. Pierson. Ovarian follicular development during the use of oral contraception: A review. *Journal of Obstetrics and Gynaecology Canada*, 26(1):19–24, 2004.

[6] A. U. Bako, S. Morad, and W. A. Atiomo. Polycystic ovary syndrome: An overview. *Reviews in Gynaecological Practice*, 5:115–122, 2005.

[7] J. M. Bland and D. G. Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, 1:307–310, 1986.

[8] P. Brodatz. *Textures: A Photographic Album for Artists and Designers.* Dover Publications, 1966.

[9] M. Broome, J. Clayton, and K. Fotherby. Enlarged follicles in women using oral contraceptives. *Contraception*, 52:13–16, 1995.

[10] C. H. Chen, L. F. Pau, and P. S. P. Wang, editors. *The Handbook of Pattern Recognition and Computer Vision*, pages 207–248. World Scientific Publishing Co, 1998.

[11] A. B. Dahl, P. Bogunovich, and A. Shokoufandeh. *Graph-Based Representations in Pattern Recognition*, pages 342–352. Springer Berlin / Heidelberg, 2009.

[12] L. Davis. *Handbook on Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[13] M. G. Eramian, G. P. Adams, and R. A. Pierson. Enhancing ultrasound texture differences for developing an *in vivo* "vitual histology" approach to bovine ovarian imaging. *Reproduction, Fertility and Development*, 19:910–924, 2007.

[14] O. J. Ginther. *Ultrasonic imaging and reproductive events in the mare*, page 378. Cross Plains, WI, USA: Equiservices, 1986.

[15] O. J. Ginther. *Ultrasonic imaging and animal reproduction: Fundamental Book 1*, page 225. Cross Plains, WI: Equiservices, 1995.

[16] O. J. Ginther, J. P. Kastelic, and L. Knopf. Composition and characteristics of composition and characteristics of follicular waves during the bovine estrous cycle. *Animal Reproduction Science*, 20:187–200, 1989.

[17] O. J. Ginther, K. Kot, L. J. Kulick, and M. C. Wiltbank. Sampling follicular fluid without altering follicular status in cattle: Oestradiol concentrations early in a follicular wave. *Reproduction and Fertility*, 109:181–186, 1997.

[18] R. M. Haralick and K. Shanmugam. Textural feature for image classification. *IEEE Transactions On Systems, Man, And Cybernetics*, 6:610–619, 1980.

[19] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[20] X. Huang, S. Z. Li, and Y. Wang. Shpae localization based on statistical method using extended local binary pattern. In *Proceedings of the Third International Conference on Image and Graphics*, pages 184–187, 2004.

[21] D. K. Iakovidis, E. G. Keramidas, and D. Maroulis. Fuzzy local binary patterns for ultrasound texture characterization. In *Proceedings of the 5th international conference on Image Analysis and Recognition*, pages 750–759, 2008.

[22] J. K. Jain, F. Ota, and D. R. Mishell. Comparison of ovarian follicular activity during treatment with a monthly injectable contraceptive and a low-dose oral contraceptive. *Contraception*, 61(3):195–198, 2000.

[23] J. P. Kastelic, R. A. Pierson, and O. J. Ginther. Ultrasonic Morphology of Corpora Lutea and Central Luteal Cavities During the Estrous Cycle and Early Pregnancy in Heifers. *Theriogenology*, 34(3):487–498, 1990.

[24] K. G. Kim, S. W. Cho, S. J. Min, J. H. Kim, B. G. Min, and K. T. Bea. Computerized scheme for assessing ultrasonographic features of breast masses. *Academic Radiology*, 12(1):58–66, 2005.

[25] N. Kim, V. Amin, D. Wilson, G. Rouse, and S. Udpa. Ultrasound image texture analysis for characterizing intramuscular fat content of live beef cattle. *Ultrasonic Imaging*, 20:191–205, 1998.

[26] S. Kita, K. Okuda, K. Miyazawa, and K. Sato. Study on the appearance of the cavity in the corpus luteum of cows by using ultrasonic scanning. *Theriogenology*, 1985.

[27] J. R. Koza. *Genetic Programming: On the Programming of the Computers by Means of Natural Selection.* MIT Press, 1992.

[28] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[29] J. Krejza, A. Nowacka, A. Szylak, M. Bilello, and L. Y. Melhem. Variability of thyroid blood flow doppler parameters in healthy women. *Ultrasound in Medicine and Biology*, 30(7):867–876, 2004.

[30] G. Layer, I. Zuna, A. Lorentz, H. Zerban, U. Haberkorn, P. Bannasch, G. V. Kaich, and U. Rath. Computerized ultrasound b-scan texture analysis of experimental fatty liver disease: Influence of total lipid content and fat deposit distribution. *Ultrasonic Imaging*, 12(171-188), 1990.

[31] Zhang M. Improving object detection performance with genetic programming. *International Journal on Artificial Intelligence Tools*, 16(5):849–873, 2007.

[32] G. Mailloux, M. Bertrand, R. Stampfler, and S. Ethier. Local histogram information content of ultrasound b-mode echographic texture. *Ultrasound in Medicine and Biology*, 11:743–750, 1985.

[33] G. Mailloux, M. Bertrand, R. Stampfler, and S. Ethier. Computer analysis of echographic textures in hashimoto disease of the thyroid. *Journal of Clinical Ultrasound*, 14:521–527, 1986.

[34] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. Technical report, University of Illinois at Urbana-Champaign, 1995.

[35] M. Negnevitsky. *Artificial Intelligence.* Addison-Wesley, 2005.

[36] T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[37] T. Ojala, M. Pietikainen, and T. Maenpaa. Gray scale and rotation invariant texture classification with local binary patterns. In *Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 404–420. Springer-Verlag, 2000.

[38] R. A. Pierson and O. J. Ginther. Ultrasonography of the bovine ovary. *Theriogenology*, 21:495–504, 1984.

[39] R. A. Pierson and O. J. Ginther. Follicular populations during the estrous cycle in heifers : Influence of day. *Animal Reproduction Science*, 14:165–176, 1987.

[40] R. A. Pierson and O. J. Ginther. Follicular populations during the estrous cycle in heifer iii: Time of selection of ovulatory follicle. *Animal Reproduction Science*, 16:81–95, 1988.

[41] R. A. Pierson and O. J. Ginther. Basic principles and techniques for transrectal ultrasonography in cattle and horses. In *WCVM June conference*, pages 1–9, 1992.

[42] Poli R. Genetic programming for image analysis. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 363–368, 1996.

[43] B. J. Rusnell, R. A. Pierson, J. Singh, G. P. Adams, and M. G. Eramian. Level set segmentation of bovine corpora lutea in ex situ ovarian ultrasound images. *Reproductive Biology and Endocrinology*, 6, 2008.

[44] J. L. Schwartz, M. D. Creinin, H. C. Pymar, and L. Reid. Predicting risk of ovulation in new start oral contraceptive users. *Obstetrics and Gynecology*, 99(2):177–182, 2002.

[45] J. Singh and G. P. Adams. Histomorphometry of dominant and subordinate bovine ovarian follicles. *The Anatomical Record*, 258:58–70, 2000.

[46] J. Singh, G. P. Adams, and R. A. Pierson. Promise of new imaging technologies for assessing ovaian function. *Animal Reproduction Science*, 78:371–399, 2003.

[47] J. Singh, R. A. Pierson, and G. P. Adams. Ultrasound image attributes of the bovine corpus luteum: structural and functional correlates. *Journal of Reproduction and Fertility*, 109(1):35–44, 1997.

[48] J Singh, R. A. Pierson, and G. P. Adams. Ultrasound image attributes of bovine ovarian follicles: endocrine and functional correlates. *J Reprod Fertil*, 112:19–29, 1998.

[49] A. Song and V. Ciesielski. Texture segmentation by genetic programming. *Evolutionary Computation*, 16(4):461–481, 2008.

[50] E. A. Stewart, W. M. W. Gedroyc, C. M. C. Tempany, B. J. Quade, Y. Inbar, T. Ehrenstein, A. Shushan, J. T. Hindley, R. D. Goldin, and M. David. Focused ultrasound treatment of uterine fibroid tumors: Safety and feasibility of a noninvasive thermoablative technique. *American Journal of Obstetrics and Gynecology*, 189(48-54), 2003.

[51] T. Sutton. *Introduction to animal reproduction*, page 336. Vermilion, AB, Canada: E.I.Sutton Consulting, 2000.

[52] J. M. Thijssen, B. J. Oosterveld, P. C. Hartman, and G. J. E. Rosenbusch. Correlations between acoustic and texture parameters from rf and b-model liver echograms. *Ultrasound in Med and Biol*, 19(1):13–20, 1993.

[53] J. W. Tom, R. A. Pierson, and G. P. Adams. Quantitative echotexture analysis of bovine corpora lutea. *Theriogenology*, 49(7):1345–1352, 1998.

[54] D. H. Townson and O. J. Ginther. Duration and pattern of follicular evacuation during ovulation in the mare. *Animal Reproduction Science*, 15:131–138, 1987.

[55] P. N. T. Wells. *Biomedical ultrasonics*, page 635. UK: Academic Press, 1977.

[56] J. Woo. A short history of the development of ultrasound in obstetrics and gynecology.

[57] B. C. Zhang, Y. S. Gao, S. Q. Zhao, and J. Z. Liu. Local derivative pattern versus local binary pattern: Face recognition with high-order local pattern descriptor. *IEEE Transactions on Image Processing*, 19(2):533–544, 2010.

[58] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):915–928, 2007.

Appendix A

An example of CL texture classifier

An example of a highly fit CL classifier learned by GP is given as follows:

$result = (((RILH7 < max(RILH3,\ RILH15)) \,||\, (0 \,||\, ((847 + max((RILH7 - RILH4),\ max(RILH15,\ RILH17))) < (max((RILH15 + RILH1), max((RILH15 + RILH2),\ (RILH1 + RILH3))) * RILH17)))) \,\&\&\, ((max(RILH1, RILH17) - RILH4) < RILH3)) \,\&\&\, ((RILH17 - RILH1) < max(RILH14,\ (max(RILH14,\ (RILH2 + RILH2)) - RILH1)))$,

where $RILHi$ is the i-th bin value of LBP_{16}^{riu2} histogram and $result$ is the output of this program which is either 0 or 1. The program includes many parentheses which specify the order of the computation. The whole program can be read from left to right with the computation performed first inside the parentheses. In order to make the program more readable, it is presented in pseudocode style as follows:

- Start;
- $a = (RILH6 < (max(RILH2,\ RILH14)))$;
- $b = 847 + max((RILH6 - RILH3),\ max(RILH14,\ RILH16))$;
- $c = max((RILH14 + RILH0),\ max((RILH14 + RILH1),\ (RILH0 + RILH2))) * RILH16$;
- $d = (max(RILH0,\ RILH16) - RILH3) < RILH2$;
- $e = (RILH16 - RILH0) < max(RILH13,\ max(RILH13,\ (RILH1 + RILH1)) - RILH0)$;
- $result = ((a \,||\, (0 \,||\, (b < c))) \,\&\&\, d) \,\&\&\, e$;
- End;

Since the classification in our experiment is a binary problem, the "improper" trees which doesn't produce a boolean value were eliminated during the GP training process implemented by using Java Genetic Algorithm Package (JGAP).