Automatic Detection And Segmentation of Bovine Corpora Lutea In Ultrasonographic Ovarian Images Using Genetic Programming And Rotation Invariant Local Binary Patterns

Meng Dong* Mark G. Eramian*† Simone Ludwig‡ Roger A. Pierson§

Abstract

In this paper we propose a fully automatic algorithm to detect and segment corpora lutea (CL) using genetic programming (GP) and rotationally invariant local binary patterns (LBP). Detection and segmentation experiments were conducted and evaluated on 30 images containing a CL and 30 images with no CL. The detection algorithm correctly determined the presence or absence of a CL in 93.33% of the images. The segmentation algorithm achieved a mean ($\pm$ standard deviation) sensitivity and specificity of 0.8693 ± 0.1371 and 0.9136 ± 0.0503, respectively, over the 30 CL images. The mean root mean squared distance of the segmented boundary from the true boundary was 1.12 ± 0.463mm and the mean maximum deviation (Hausdorff distance) was 3.39 ± 2.00mm. The success of these algorithms demonstrates that similar algorithms designed for analysis of *in vivo* human ovaries are likely viable.

Keywords: Texture Analysis, Ultrasonography, Corpora lutea, Local Binary Patterns, Genetic Programming.

Total word count:

Abstract Word Count: 99

Number of Figures: 12

Number of Tables: 6

*Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada.
†Corresponding Author, eramian@cs.usask.ca, Tel: 306-966-4028, Fax: 306-966-4884
‡Department of Computer Science, North Dakota State University, Fargo, North Dakota, USA
§Department of Obstetrics, Gynecology and Reproductive Sciences, University of Saskatchewan, Saskatoon, Saskatchewan, Canada

1 Introduction

The ovaries of all mammalian species, including humans and cattle, contain follicles and corpora lutea (CL). The corpus luteum is a temporary endocrine gland that forms from the follicular tissue after ovulation. Its primary function is to produce the steroid hormone progesterone, which is necessary for maintaining pregnancy [13] and for regulation of folliculogenesis [4]. For background on follicular dynamics in cattle and humans, the reader is referred to Adams and Pierson [1] and Baerwald et al. [3], respectively.

Singh et al. [18] demonstrated that mean pixel intensity in ultrasonographic images of bovine ovaries was significantly different between the metoestrus, early/mid-dioestrus, and pro-oestrus phases. In the same study several correlations between mean pixel intensity of luteal tissues in ultrasonographic images and histological and biochemical observations were shown, specifically: that mean pixel intensity was strongly correlated with volume density of luteal cells, connective tissue, and stromal components; and that mean pixel intensity was strongly correlated with plasma and luteal progesterone concentration.

Similar variations in mean intensity of luteal pixels was observed in ultrasonographic images of human ovaries through the interovulatory interval (IOI) and the variations were not significantly different between women with cycles of two and three waves of follicular growth [4]. In the same study, mean luteal area was found to be positively correlated with serum progesterone and estradiol concentrations, again with no differences between women with two and three-wave cycles.

These observations suggest the possibility of predicting the ovarian physiological status of an individual from luteal brightness and luteal area measurements based on a single day's ultrasonographic examination. Indeed, it has been shown using data collected from Singh et al.'s experiments [18] that CL size, and number and sizes of ovarian follicles are sufficient information to predict bovine oestrus phase [10]. Part of the process of automating such a determination is developing the capacity for automatic localization of the CL. Presently, CL detection and segmentation in ultrasonographic images are typically performed manually which is time consuming and subject to variance in human interpretation of images.

The objective of our study was to design a high-performance, fully automated computer algorithm for detection of presence or absence of a CL in a single image and the segmentation CL regions. We have developed CL detection and segmentation algorithms for bovine ovaries imaged *ex situ* under ideal conditions [18] which demonstrates the potential feasibility for doing the same with *in vivo* human CL images. The bovine model for studying human ovarian function has been well established [1, 17].

Validation of our algorithm was performed using the same images obtained by Singh et al. [18] which were used to correlate luteal pixel brightness with the aforementioned histological and biochemical features. These

algorithms could be used as part of a larger automated system to automatically detect, segment, and sample the luteal brightness in ultrasonographic images, and predict the ovarian physiological status based on a single ultrasonographic examination.

No previous studies of automated CL detection in a single image were found in the literature. The only previously published study on CL segmentation was limited in size and scope and the algorithm presented was not completely automatic [16]. 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. The algorithm requires an initial contour to be manually placed within the CL region to be segmented. Moreover, for CL's exhibiting a central cavity, the initial contour must enclose and not intersect with the central cavity.

In contrast, our algorithm uses genetic programming (GP) 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 CLs.

2 Methods

Our CL segmentation and detection algorithms work by learning computer programs that are highly fit for the task of distinguishing the ultrasonographic echo texture of corpora lutea from other textures in the image. We begin this section with some background on genetic programming and local binary patterns, followed by a detailed description of our use of genetic programming, our algorithms, and experimental designs.

2.1 Genetic Programming

Genetic Programming (GP) offers a solution to user-defined tasks through the evolution of computer programs by methods inspired by the mechanisms of "natural selection" [7, 8]. GP searches the space of possible computer programs for one program that is highly fit for solving the problem at hand [12]. GP has been demonstrated to be a flexible method of problem solving for a diverse range of complex problems, such as classification [19]. Classifiers learned via GP are proven capable of classifying Brodatz textures [19] (a standard data set for evaluating texture analysis and classification algorithms [5]).

GP can operate on programs given by various representations, such as program trees, graphs, and linear representations. After an initial population is generated using the chosen representation, the population is evolved from one generation to the next using operations called *crossover*, *mutation*, and *cloning*. Cloning a copies a member of the current population to the next generation unchanged. Crossover creates two members

of the new population that result from a recombination of two members of the current population. Mutation adds a member to the new population that is the product of some small, but random change in a member of the current population. A new population is generated from a sequence of these three events, where each event in the sequence is independently chosen at random according to some predefined probabilities of each event occurring. When the new population is equal in size to the current population, the new population becomes the current population and the process repeats until some stopping criterion occurs.

2.2 Rotationally Invariant Local Binary Patterns

The Local Binary Pattern (LBP) operator was first introduced by Ojala et al. [14], as a non-parametric, gray-scale invariant texture analysis model, which summarizes the local spatial structures of an image [6]. The use of LBP for ultrasound texture description is motivated by the Ojala et al.'s report of improved classification rates for Brodatz textures described by rotationally invariant LBP compared with previously reported methods [15].

In a 3×3 or 5×5 pixel neighbourhood, a circular sampling of 8 or 16 subpixels, respectively, is computed using bilinear interpolation of the original pixel intensities. This process is illustrated in Figure 4, where p_i corresponds to the subpixel locations at which bilinear interpolation occurs. The size of the local neighbourhood is increased from 3×3 to 5×5 for the 16-bit LBP code because the eight additional subpixels would not provide much additional information if constrained to the 3×3 neighbourhood [15].

The interpolated intensity, g_i, at each interpolation point p_i is thresholded by the intensity of the central pixel g_{center}. In this way, a local neighbourhood can be changed into a binary pattern (a sequence of binary digits) by concatenating the thresholded (on or off) interpolated intensities in index order [15].

Rotational invariance is achieved by finding the circular permutation of a pattern that results in the smallest value when it is interpreted as an integer in base 2. Uniform local binary patterns are formed by grouping of patterns according to their discriminatory power and considering them as the same pattern. Uniformity of a pattern is defined as the number of 0/1 transitions as the pattern is read from left to right. Patterns of higher uniformity (fewer such transitions) have been found to be more discriminatory, thus all non-uniform patterns (with a uniformity greater than 2) are grouped and considered to be the same pattern. The reader is referred to [15] for details on uniform rotationally invariant binary patterns.

2.3 CL Detection and Segmentation

2.3.1 Images Data Sets

The images of *ex situ* bovine ovaries used for training and validation of our algorithm were taken from the previous study performed by Singh et al. [18], some results of which were summarized in the introduction. We used images of pairs of bovine ovaries which were excised at three different time points during the estrous cycle: day 6 wave 1 (D6W1, early dioestrus, $n = 8$), day 1 of wave 2 (D1W2, during metoestrus, $n = 6$), and after the onset of pro-oestrus at $\geq$ 17 days (D$\geq$17, $n = 8$), 22 animals in total. Day 0 was the day of the last observed ovulation, "wave" refers to waves of follicle growth [18]. The ovaries were placed in degassed phosphate-buffered saline and imaged with an ATL Ultra Mark 9 HDI ultrasound machine (Advanced Technology Laboratories, Bothell, WA, USA) interfaced with a broad-band (5–9MHz), convex-array ultrasound transducer. Both ovaries of each animal were scanned sequentially at 0.5mm increments in parallel planes. All of the images obtained were 640$\times$480-pixel grayscale images. After being imaged, the corpora lutea were dissected, processed, and stored pending the biochemical and histological analysis [18] which were described briefly in the introduction herein.

For each animal, one or two images containing CL were selected randomly from among those images of the ovary containing the CL where the CL area present in the image plane were near maximal for the images of that ovary. The same number of images without CL were selected randomly from images of the other ovary known not to exhibit a CL. In total, 60 images, 30 of which contain a CL, were selected.

Although this data set is 14 years old and the ultrasound machine used for acquisition is older, we justify its use by noting that the dataset is unique in that the images are accompanied by the biochemical and histological analysis allowing CL regions in images to be identified with certainty. The dataset is also unique in that ovaries were collected at specific time points in the estrous cycle allowing the estimation of our algorithms' performance on ovaries imaged at different points in the cycle with different average brightness [18]. Moreover, the images were acquired using a high-frequency transducer similar to those still in use today.

2.3.2 Preprocessing

Each image in our data set contains a reference grayscale bar which appears as steps of decreasing intensity levels. Grayscale standardization was performed on all images in our training and testing sets as follows:

1. the grayscale bar was localized and the unique intensities in the grayscale bar identified using the algorithms of Ahmed and Eramian [2];

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 5(b) shows the result of standardizing the grayscale intensity scale of the image in Figure 5(a). Figure 5(c) plots the curve for grayscale remapping of Figure 5(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 linearized mapping.

An expert in ovarian ultrasound interpretation, guided by the ovarian morphology documented during surgery and histologic examination in [18], delineated CL boundaries in each image manually to serve as a "ground truth". Accuracy of our segmentation algorithm was determined by comparing segmentations resulting from our algorithm with ground truth. Ground truth for the detection problem was similarly informed by data recorded during the work of Singh et al. [18] and was recorded as a simple "yes/no" for each image in our data set.

2.3.3 Texture Description

Our algorithms attempt to classify the texture of local neighbourhoods in an image as either CL or non-CL. Texture descriptions of such a neighbourhood are formed by constructing a histogram of the 16-bit rotationally invariant uniform local binary patterns of pixels in the neighbourhood; the histogram bin values essentially form a feature vector. Programs that are learned by genetic programming are defined in the next section to use one or more features from these texture descriptions as inputs.

In the sequel, 16-bit rotationally invariant uniform local binary patterns are denoted LBP_{16}^{riu2} following the notation of [15]. The notation $\mathrm{RILH}(i)$ (short for Rotationally Invariant LBP Histogram) refers to the i-th feature (or histogram bin) in such a texture description.

2.3.4 Learning CL Texture Classifiers by Genetic Programming

A genetic programming solution was used to learn a classifier for distinguishing between CL texture and other textures. Program trees were used to represent the members of our population of programs undergoing "evolution". An example of a small program tree is displayed in Figure 3. Each *node* of the tree contains a symbol representing either data or operations on data performed by the program. Leaf nodes, which have no children, are found at the bottom of the tree and represent inputs to the program. The set of symbols representing input

data is the *terminal set*. Internal nodes, which have one or more children, represent operations on data. The operands for an operation in an internal node are the outputs of the operations represented by its children, or simple data inputs in the case of children which are leaf nodes. The set of symbols representing operations in the internal nodes is called the *function set*. The output of the operation at the root node of the tree is the output of the program; in our case the output is binary, indicating whether the input texture was classified as CL or non-CL.

For our program trees, the function set consisted only of some basic arithmetic and logical operators (Table 1). The terminal set consists of the histogram bin values of LBP_{16}^{riu2} ($\mathrm{RILH}(i)$ for all i for the input texture) and random numbers. The random numbers in the terminal set allow construction of programs that can not only compare the LBP histogram bin values with each other but also compare them with constant numbers. For similar textures, the LBP_{16}^{riu2} histograms may have similar shape but quite absolute values in each bin, so just comparing the bin values with each other may result in a non-optimal classifier.

The crossover operation for program trees is defined on two operand programs p_1 and p_2. Two randomly chosen subtrees of p_1 and p_2 are exchanged. That is, if t_1 and t_2 are the roots of randomly chosen subtrees from p_1 and p_2 respectively, and n_1 and n_2 are respectively the parents of t_1 and t_2, then the crossover is completed by making n_1 the parent of t_2 and n_2 the parent of t_1 and adding the modified p_1 and p_2 program trees to the new population.

The mutation operation requires one operand p_1 and is completed by changing a randomly chosen function or terminal node within p_1 to another randomly chosen function or terminal respectively. The modified p_1 is then added to the new population.

The clone operation requires one operand p_1 and simply copies p_1 to the new population unmodified.

The GP fitness function was defined as the number of misclassifications on the training set. Thus, for each new generation of programs, the fitness of each program was computed by using it to classify each texture sample in the training set and comparing the result to the known texture classes of each sample.

Highly fit programs were learned from the texture descriptions of 32×32 subimages sampled from both CL and non-CL images from our training set. 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 number of non-CL samples was used for training to ensure diversity of samples in this class.

The termination criteria for each run was either perfect classification (zero misclassifications over the whole

training set), or after some number of generations had been processed. The output of the GP learning process was the program, C, in the final population with the greatest fitness. The classifier program C was used in our CL segmentation and detection algorithms, discussed in the next sections.

2.3.5 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 which was learned by GP. Each image pixel in the window 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 Song and Ciesielski [19]. The steps of this segmentation algorithm are described in Algorithm 1.

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 LBP^{riu2}_{16} histogram.
and output is label "CL" or "non-CL".

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

1. Slide an $n \times n$ window over the input image I in raster order with step sizes δ_x and δ_y. For each windowed subimage J:
   (a) Compute the LBP histogram for the window.
   (b) Classify the texture J as "CL" or "non-CL" using C.
   (c) 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

2.3.6 CL Detection Algorithm

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

To learn the region classifier, 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 4. The region classifier is learned by GP with the same parameters used when learning C, except that the region properties are used as inputs 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 : An ovarian ultrasound image I,
A CL texture classifier C whose input is an LBP_{16}^{riu2} histogram 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 to produce an output image I_{bw}.
2. Compute the region properties in Table 4 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.

3 Results

3.1 Experimental Design

A three-way cross-validation experiment was performed to evaluate the performance of our segmentation and detection algorithms using our data set of 60 images (30 CL images and 30 non-CL images). The whole dataset was split into three disjoint subsets, each with 20 images. In each subset, there were an equal number (10) of CL images and non-CL images. Moreover, each subset has the same number of images chosen randomly from each estrous cycle time point. The overall performance was computed as the average performance of the three data folds.

For the GP learning process, operands for the three genetic operations were chosen using tournament selection [11] using a tournament size of $k = 4$. The probability of crossover was $P_c = 0.8$, the probability of mutation was $P_m = 0.1$ and the probability of cloning was $P_{cl} = 0.1$. Typical ranges for crossover and

mutation probabilities are $[0.7, 1)$ and $[0.001, 0.1]$, respectively [12]. We choose a relatively large mutation probability (0.1) to avoid getting trapped in a local optimum. The population size was 600. The initial generation of program trees were randomly generated subject to a maximum depth $D = 20$. Because the rotation invariant LBP histograms are computed from 32×32 subimages in our experiment, the possible largest bin value in the LBP_{16}^{riu2} histogram is 1024. Accordingly, the random numbers in the terminal set are chosen from the interval $[0, 1024]$. The maximum number of generations was set to 500; the GP performance graph from our experiment, given in Figure 6, justifies this choice. Since the fitness function is being minimized, *Performance*, defined as follows, is used for vertical axis of Figure 6.

$$\textit{Performance} = 1 - \frac{\textit{Misclassifications}}{\textit{Total Number of Training Samples}} \tag{1}$$

where *Misclassifications* is the total number of classification errors. In Figure 6, the performance curve rises steeply at the beginning of the GP run, but flattens after around generation 500 and little further benefit is gained by computing additional generations suggesting that fitness is at or very near the global optimum. See Table 2 for summary GP learning parameters.

For our CL segmentation and detection experiments we used a window size of $n = 32$, and step sizes $\delta_x = \delta_y = 2$. In Song and Ciesielski [19], 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 32×32. Segmentation parameters are summarized in Table 3.

3.2 Performance Measures

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

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

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:

$$\text{sensitivity} = \frac{|TP|}{|TP| + |FN|}, \tag{2}$$

$$\text{specificity} = \frac{|TN|}{|TN| + |FP|}, \tag{3}$$

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.

We also computed the boundary based accuracy measures of mean absolute distance (MAD), root mean squared distance (RMSD), and Hausdorff distance for each segmentation. The boundary of a segmented region is the inner boundary of the region. MAD is the average distance from a pixel in the segmented region boundary to the closest ground truth region boundary. RMSD is similar to MAD, except it is the square root of the averaged squared distance. Hausdorff distance is the largest shortest distance between the segmented region boundary and the ground truth region boundary.

3.3 CL Detection Results

The results of CL detection experiment are shown in Table 5 which, for each fold of the cross-validtion, includes the training accuracy and the CL detection accuracy. The average accuracy of final decisions over all three folds was 93.33%. The most distinguishing region features were the area of the output region and the major axis length of the output region. Box plots of the area and major axis length region properties from the 30 CL images and the 30 non-CL images are shown in Figure 7. These show graphically that the medians of the features are significantly different between CL and non-CL images with 95% confidence (Figure 7).

3.4 CL Segmentation Results

The results of CL segmentation for all the 30 CL test images in our dataset are shown in Figure 8. The mean ($\pm$ standard deviation) sensitivity and specificity are 0.8693 ± 0.1371 and 0.9136 ± 0.0503. The mean RMSD over all 30 test images was 1.23 ± 0.58mm. The mean Hausdorff distance was 4.1 ± 1.5mm.

Figures 9, 10, 11 and 12 show four 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, which is shown in Figure 9 and Figure 10.

Though we cannot fairly compare runtime of our algorithm to that of Rusnell et al. since the same hardware was not available, we state the algorithms' runtimes here, nonethless. The mean runtime reported for segmentation of a single image using the algorithm of Rusnell et al. [16] was 5min 54s on an 2.8GHz Pentium 4 Processor with 2 GB of RAM running Mandriva Linux, using an implementation in MATLAB (The Mathworks Inc, Natick, MA, USA) version 7.4.0.336 [16]. The mean runtime for segmenting one image using our method was 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), in our experiment took around 110 minutes to finish on the same Mac machine.

4 Discussion

4.1 CL Segmentation

The mean ($\pm$ standard deviation) RMSD and HD for Rusnell et al.'s algorithm [16] were 1.12 ± 0.463mm and 3.39 ± 2.00mm, respectively, while our algorithm achieved 1.23 ± 0.58mm and 4.1 ± 1.5mm. Welch's t-test, which assumes unequal sample variances, was used to determine whether the means of our segmentation results differed significantly from those of [16]. For RMSD there was a significant difference between the mean RMSD for our algorithm and the mean RMSD from [16] at the 5% confidence level, but not at the 1% confidence level ($p = 0.03$). The confidence interval was very large ([0.054, 0.954], suggesting while there is evidence supporting a significant difference, it is not particularly strong evidence. For HD, no significant difference was observed ($p = 0.34$).

The mean sensitivity and specificity of the CL segmentations produced by Rusnell et al.'s algorithm were 0.814 ± 0.171 and 0.990 ± 0.00786 over 8 images, while we achieved a mean sensitivity and specificity of 0.8693 ± 0.1371 and 0.9136 ± 0.0503 over 30 images from different time points in the estrous cycle. For sensitivity, Welch's t-test reported no significant difference ($p = 0.41$). For specificity, a significant difference was observed ($p = 3.3 \times 10^{-09}$). Summary results from all four t-tests are given in Table 6.

The reason for the higher specificity of Rusnell et al. [16] is that their algorithm employs a segmentation algorithm based on level-set methods where 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 larger number of FN pixels and few FP pixels. In contrast, although our mean specificity is lower, but it is still high, at 0.9136 ± 0.0503 and, unlike Rusnell et al.'s algorithm, our algorithm achieves this automatically, without user intervention or prior knowledge of the CL's location. Moreover, our segmentation algorithm was tested over 30 images from different time points in the estrous cycle while the previous results were produced over 8 images demonstrating effectiveness over a wider range of images.

4.2 CL Detection

From the results of our CL detection experiment, it is clear that Algorithm 2 works well on the bovine CL dataset including images from different time points of the estrous cycle. Moreover, at the 5% significance level, a Welch's t-test showed that there was no significant difference in CL detection rates between ovaries collected from each time point.

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. Therefore, the decision can be made about the presence or absence of CL and an automatic detection of evidence of ovulation can be achieved. 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 [16].

4.3 Limitations and Future Work

Although the CL detection and segmentation results are very good, there are still some limitations of this method. Singh et al.'s data [18] also contains images from ovaries collected on the third day since ovulation during metoestrous. For these images, 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, the segmentation algorithm would benefit from improvements to enhance the sensitivity.

Future experiments will develop a methodology based on this work to perform human CL detection and segmentation on images of *in vivo* ovaries, which is more challenging than our bovine CL images which were acquired *ex vivo* under ideal imaging conditions. This success of our algorithms demonstrate the promise of this method for solving CL segmentation and detection problems in images of *in vivo* human ovaries. Such

algorithms could assist with monitoring the development of CL over time, facilitating the interpretation of ovarian ultrasonography, and the diagnosis of ovarian diseases.

4.4 Conclusion

Herein we presented a new CL segmentation and detection algorithm. The accuracy of the segmentations were found not to be dramatically worse or better than the previous method, however, the current solution is fully automatic and more than five times faster which is a major advantage over the previous method. The other main contribution of this work is a CL detection algorithm which correctly determined the presence or absence of a CL in an image in 93% of cases.

5 Acknowledgments

The authors would like to express their sincere thanks and appreciation to Dr. Jaswant Singh, Western College of Veterinary Medicine, University of Saskatchewan, for permitting the use of his images in this study, and for valuable advice rendered.

This research was supported, in part, by the Natural Sciences and Engineering Research Council of Canada, grant number RGPIN262027-03.

References

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

[2] Waqas Ahmed and Mark 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.

[3] Angela R. Baerwald, Gregg P. Adams, and Roger A. Pierson. A new model for ovarian follicular development during the human menstrual cycle. *Fertil Steril*, 80(1):116–122, July 2003.

[4] Angela R. Baerwald, Gregg P. Adams, and Roger A. Pierson. Form and function of the corpus luteum during the human menstrual cycle. *Ultrasound Obst Gyn*, 25:498–507, 2005.

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

[6] Dimitris K. Iakovidis, Eystratios G. Keramidas, and Dimitris 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.

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

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

[9] Henning Lategahn, Sebastian Gross, Thomas Stehle, and Til Aach. Texture classification by modeling joint distributions of local patterns with gaussian mixtures. *IEEE Transactions on Image Processing*, 19(6):1548–1557, 2010.

[10] Maryruth J. Lawrence, Mark G. Eramian, Roger A. Pierson, and Eric Neufeld. Computer assisted detection of polycystic ovarian morphology in ultrasound images. In *Proceedings of the 4th Canadian Conference on Computer and Robot Vision*, pages 105–112. IEEE Computer Society, 2007. (56/103 submissions accepted).

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

[12] Michael Negnevitsky. *Artificial Intelligence*. Addison-Wesley, 2005.

[13] Jimmy D. Neill, editor. *Knobil and Neill's Physiology of Reproduction*. Academic Press, 2005.

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

[15] Timo Ojala, Matti Pietikainen, and Topi 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.

[16] Brennan J Rusnell, Roger A Pierson, Jaswant Singh, Gregg P Adams, and Mark G Eramian. Level set segmentation of bovine corpora lutea in ex situ ovarian ultrasound images. *Reproductive Biology and Endocrinology*, 6, 2008.

[17] 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.

[18] J. Singh, Roger A. Pierson, and Gregg P. Adams. Ultrasound image attributes of the bovine corpus luteum: structural and functional correlates. *J Reprod Fertil*, 109(1):35–44, 1997.

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

6 Figure Captions

Figure 1: An example of a CL image. The pink contour denotes the boundary for the CL region. The blue contour identifies the boundary of central cavity inside the CL.

Figure 2: An example of an image not containing a CL.

Figure 3: An example of program tree. Terminal nodes contain program inputs from the *terminal set*, internal nodes represent program functions from the *function set*.

Figure 4: 8-bit and 16-bit circular neighbour set: p_i is used to denote the subpixel locations at which bilinear interpolation of the original pixel intensities occurs.

Figure 5: The image grayscale standardization process. a) Original image; b) after standardization; c) linearization of intensities.

Figure 6: GP Performance Graph. X axis is the number of generations, Y axis is the Performance value of best-so-far classifier.

Figure 7: Examples of Box Plot for Region Properties. In each subfigure, column 1 corresponds to the box plot of the region property values from 30 CL images while column 2 is the box plot for the values from 30 non-CL images.

Figure 8: CL segmentation performances for all the 30 CL images. For each image, the green bar indicates the sensitivity and the yellow bar indicates the specificity.

Figure 9: An Example of CL Segmentation Results.

Figure 10: An Example of CL Segmentation Results.

Figure 11: An Example of CL Segmentation Results.

Figure 12: An Example of CL Segmentation Results.

7 Tables

Table 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

Table 2: GP Classifier Learning Parameters.

Parameters	Descriptions
$P_c = 0.8,$	Probability of crossover.
$P_m = 0.1$	Probability of mutation.
$sizeP = 600$	Population size.
$D = 20$	Max. depth of initial program trees.
$NoG = 500$	Max. number of generations.

Table 3: Segmentation/Detection Algorithm Parameters.

$h = 480, w = 640$	Height and width of images.
$n = 32$	The size of sliding window
$\delta_x = \delta_y = 2$	Step size for moving the window.

Table 4: 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 on pixels 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.

Table 5: CL Detection Performances.

Test ID	Training accuracy for CL classifiers	CL detection accuracy
1	93.70%	85.0%
2	93.37%	100.00%
3	91.78%	95.00%
Average	92.95%	93.33%

Table 6: T-test Results

Samples for t-statistic	t-value	Degrees of Freedom	Confidence Interval	p-value
RMSD	2.35	18.3	$[0.054, 0.954]$	0.0301
HD	0.986	9.36	$[-0.959, 2.46]$	0.35
Sensitivity	0.846	9.54	$[-0.0913, 0.202]$	0.418
Specificity	-7.94	33.4	$[-0.0956, -0.0566]$	3.40×10^{-9}