University of Bordeaux

Internship Report

Master of Software Engineering (2013 - 2015)

Automatic morphology: landmarks estimation in biological images

Student:
LE Van Linh

Supervisor:
Marie BEURTON-AIMAR

November 30, 2015

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Mrs Marie BEURTON-AIMAR for her agreement, guide and support during the planning and developing of my internship.

I would like to thank Mr Jean-Pierre SALMON for his generous help and comment during my work. I would like to thank the staffs, students in LaBRI, who helped, supported with the technique and providing me a professional working environment.

I would also like to thank all the professors in the University of Bordeaux and the PUF-HCM, who imparted a lot of knowledge about learning and researching. Last but not least, I would like to thank the dean of the IT-DLU, who gave me the opportunity to join in this course. Finally, I would like to thank my family and colleagues for their support and encouragement through my study.

Abstract

Image processing is a field that has many applications in life. It could be the usual application or the applications in medicine or cosmology. To obtain the result best,most of the applications must follow the two processes: firstly, pre-processing the images with some appropriate operators to enhance the interest and to reduce the noises. Secondly, applying the main operations to obtain the result.

The goal of this project to build the program fully functional program about processing base on the biological images. During my internship at LaBRI, my tasks was developing the algorithm to automatic identify the landmarks of the biological images and to classify the biological images. The method based on the morphometry in image processing.

Finally, I integrated my functions into the Image Processing for Morphometrics (IPM) software, which was developed by NGUYEN Hoang Thao. Besides, we also debug the previous code and write the documentation for the next phases.

Contents

List of Figures

Introduction

.

Morphometric is a concept refers to qualitative analysis of form, it includes the size and the shape of an object. Mophormetric analyses are commonly performed on organism. An objective of morphometric is to statistic hypotheses based on the effect of shape. In image processing, morphometrics can be used to detect the shape, size or changes in form on family organism. Generally, to measure the object's morphometry, we can use one of three morphometric forms: traditional morphometrics, landmark-based geometric morphometrics, outline-based morphometrics.

In this internship, we consider landmark-base geometric morphometric. And our goal is to implement the methods to extract the landmarks automatically from biological images as well as calculate some statistics to comparing results.

The whole report has five chapters. In the first chapter, this is the short introduction about the context working as well as the objectives of my internship. Chapter 2, introduces the necessary preliminaries in image processing field which we use to implement the methods. In the third chapter, this is the mainly chapter of report. We mention method to extract the landmarks automatically from the biological images.In the chapter 4, we will focus on the design of software. This chapter introduces the architecture and explain the model of software. Finally, chapter 5, we will introduce the result and our experimentation. Besides, we give a short summary about the work and describe about the future work.

Chapter 1

Context

1.1 Pôle Universitaire Français

The Pôle Universitaire Français (PUF) was created by the intergovernmental agreement of VietNam and France in October 2004. With ambition is building a linking program between the universities in VietNam and the advanced programs of universities in France. There are two PUF's center in VietNam: Pôle Universitaire Français de l'Universite Nationalé du Vietnam - Ha Noi located in Ha Noi capital (PUF-Ha Noi) and Pôle Universitaire Français de l'Universite Nationalé du Vietnam - Ho Chi Minh Ville located in Ho Chi Minh city (PUF-HCM).

1.1.1 PUF-Ha Noi

PUF-Ha Noi is regarded as a nursery for the linking program, it support on administrative procedure and logistics for the early year of program. Besides, PUF-Ha Noi also implement the training program regularly about Master 2 provided by universities and academies in France. About administration, PUF-HN directly under Institut Francophone International (IFI), which was created by VietNam National University at HaNoi in 2012.

1.1.2 PUF-HCM

PUF-HCM[1] is a department of VietNam National Univeristy at Ho Chi Minh city. From the first year of operations, PUF-HCM launched the quality training programs from France in VietNam. With target, bring the programs which designed and evaluated by the international standards for Vietnamese student. PUF-HCM always strive in our training work.

[1] http;//pufhcm.edu.vn

So far, PUF-HCM have five linking programs with the universities in France, and the programs are organized into the subjects: Commerce, Economic, Management and Informatics. In detail:

- Bachelor and Master of Economics : linking program with University of Toulouse 1 Captiole
- Bachelor and Master of Informatics: linking program with University of Bordeaux and University of Paris 6.

The courses in PUF-HCM are provided in French, English and Vietnamese by both Vietnamese and French professors. The highlight of the programs are inspection and diploma was done by the French universities.

1.2 Laboratoire Bordelais de Recherche en Informatique

The Laboratoire Bordelais de Recherche en Informatique (LaBRI)[2] is a research unit associated with the CNRS (URM 5800), the University of Bordeaux and the Bordeaux INP. Since 2002, it has been the partner of Inria. It has significantly increased in staff numbers over recent years. In March 2015, it had a total of 320 members including 113 teaching/research staff (University of Bordeaux and Bordeaux INP), 37 research staff (CNRS and Inria), 22 administrative and technical (University of Bordeaux, Bordeaux INP, CNRS and Inria) and more than 140 doctoral students and post-docs. The LaBRI's missions are: research (pure and applied), technology application and transfer and training.
Today the members of the laboratory are grouped in six teams, each one combining basic research, applied research and technology transfer:

- Combinatorics and Algorithmic
- Image and Sound
- Formal Methods
- Models and Algorithms for Bio-informatics and Data Visualisation
- Programming, Networks and Systems
- Supports and Algorithms for High Performance Numerical Applications

[2]http://www.labri.fr

Within these team, research activities are conducted in partnership with Inria. Besides that, LaBRI also collaborate with many other laboratories and companies on French, European and the international.

1.3 The Internship

The internship is intended to be a duration to apply the knowledge to the real enviroment. It shows the ability synthesis, evaluation and self-research of student. Besides, the student may study the experience from the real working environment. My internship is done under the guidance of Mrs Marie BEURTON-AIMAR in a period of six months at LaBRI laboratory.
As mentioned in previous part, in this internship, we consider landmark-base geometric morphometric. **Morphometric landmarks are points that can be defined in all spceimens and located precisely**[4]. They are widely used in many biological and medical applications. Presently, the landmarks can be extracted manually or automatically. Automatic extraction of landmarks can be finished by applying other ways such as recognition (cross-correlation), shape analysis, etc. As part of goals, the objectives of this internship is implementing a method to automatic extract landmarks from biological images. This method was proposed by Palaniswamy[4].

Chapter 2

Morphometric

Image processing is a field that used mathematical operation to an input image. The output of the image processing process may be either an image or a set of characteristics related to the image. And most of the image processing techniques are performed on two-dimensional images. In image processing, we have a lot of operators used for measure the form of image. In this chapter, we introduce some basis operations that are often used as the object of this internship.

2.1 Image filtering

Image filtering is a process to modify or enhance the image's quality. This is known as a "neighborhood" operation. The neighborhood is a set of pixels around a selected pixel. In image processing, with a pixel, we may obtain have 4-neighbors or 8-neighbors of it. Image filtering determines the value at the selected pixel by applying some operations with the values of its neighbors. One of the filtering operators is smoothing, also named blurring. This technique is used in preprocessing steps, particularly in noise reduction. With a matrix called kernel. It was sliding over the image. At each position, the output of a value at that position is calculated by meaning its neighborhoods value. In image processing, we have many filtering techniques. But there are 2 main types:

Linear filter: The idea behind this filtering method is replacing the value of every pixel in the image by the average of the gray levels in the neighborhood defined by the filter mask. By this work, this filter sometime are called averaging filter. The result of this process is an image with the sharp edges reduced in gray level, it also reduces the noise because the noise is typically and random in the image. The mask is a matrix and its useful for blurring, sharpening, edge-detection, etc. The output image is accomplished by convoluting between a mask and an image.

Order-Statistics filter: By ordering the pixels in the image and then replacing the values of the center pixel with the value determined by the ranking result. Median filter is an example of this technique.

2.2 Histogram

Histogram is a representation of the distribution of data on the regions (we called bins) in the data range. The bins are the number of sub-ranges when we divide the entire data range into several small intervals (i.e. With the range of [0 - 255] and the size of each sub-range (bin) is 16, the number of bins is $256/16 = 16$ bins. The first bin range is [0 - 15], the second range is 15 - 30, and so on). The value at each bin is the number of data which have value belong to it. Normally, histogram is representing by the columns chart with x-axis represent as for the number of bins, and y-axis represent as for the value of each bin.
Histogram can be used effectively for image enhancement, it's also useful in many image processing applications, such as image compression and segmentation.

Histogram equation: is a method that allows adjusting the contrast using the histogram of

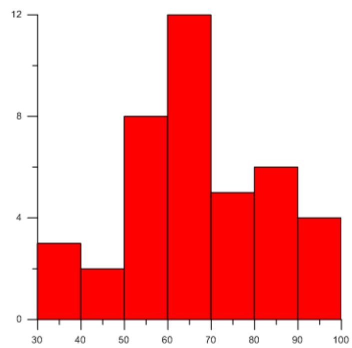


Figure 2.1: An example about histogram

image. It maps the distribution on a histogram to a wider distribution of the intensity values. By applying this, the image could be brighter.

Histogram matching: is the method that adjusts two images using the histogram. This method is finished by calculating the cumulative distribution functions of the two histograms to finding the histogram matching function. Finally, applying the matching function on each pixel of the image to get the result.

2.3 Segmentation

Segmentation subdivides an image into the regions. The size of the regions is depended on the problem being solved. This mean, segmentation should stop when the regions of interest in application have been detected. In real world, the segmentation is applied to many fields such as machine vision, medical imaging, object detection, etc. The most what of segmentation algorithms are based on the basic properties of intensity values: discontinuity and similarity. In the first case, the segmentation based on abrupt changes in intensity. In the second case, the image segmentation based on a predefined criteria. It means the image was segmented into regions those are similar according to a set of criterias. And, we have many the methods to segment an image such as thresholding method, region growing method, clustering method, histogram-based method, etc.

Thresholding is the simplest method of image segmentation. Thresholding uses a particular threshold value "t", which splits the image into two parts: the first part includes pixels which have the value greater than "t", and the second part contains the pixels smaller than "t". With this technique, thresholding can be used to create an binary image from a gray scale image. In fact, we have many type of thresholds, as follows:

- *Global thresholding*, when t is a constant over an entire image
- *Variable thresholding*, when t changes over an image
- *Local or regional thresholding*, is variable threshoding in a region of an image
- *Dynamic or adaptive thresholding*, if t depends on the spatial coordinates.
- *Multiple thresholding*, thresholding on 3 dominant modes (color image)

Canny algorithm is an edge detection algorithm that uses to detect the structure of an image. The process of this algorithm can be broken into the steps as follows [1]:

- Apply the Gaussian filter to smooth the image (remove the noise),
- Find the intensity gradients of the image,

[1] https://en.wikipedia.org/wiki/Canny_edge_detector

- Apply non-maximum suppression to get rid of spurious response to edge detection,
- Apply double threshold to determine potential edges,
- Track edges.

2.4 Color processing[3]

The usage of colors in image processing are not limit in identifying or extracting an object from its scene, it also a factor of image analysis. Color processing can be a effected on each component image individually or works directly with pixels-based on a color model. The color models is a specification of colors in some standard, generally accept way such as BGR, CMY, HSV or Grayscale model.

- BGR model: using blue, green, red as three primary colors. Image presented in this model consists of three components images for each primary color.
- CMY model: used for hardcopy devices. Based on the BGR mode, each value in CMY mode was computed by integrating the 2 primary colors in BGR. Specific, C (cyan) is consist of green and blue, M (magenta) is consist of red and blue and Y (yellow) is consist of red and green.
- HSV model: different from BGR, HSV uses the 3 components which are hue, saturation and brightness to represent image. Hue is a color attribute which describes the pure color (yellow, orange, red) and saturation gives a degree to pick the pure color diluted by white light. Brightness is a notation of intensity for color sensation.
- Grayscale model: The colors in grayscale are black and white because it just carry the intensity information on each pixel. Because that, the image in grayscale mode was called black and white image. The color of each pixels in image from black, where have weakest intensity to white at the strongest intensity.

The most popular color operations in image processing fields is transformation. It is a process to convert image between the color models by using a transform expression such as BGR to HSV, HSV to BGR, BGR to Grayscale.

Besides, by considering a specific characteristic of each color space, allow as us classify the pixels in the image. This idea can be used to segment objects of an image. HSV model is widely used to compare the color because the range of color (Hue value) is specific.

Chapter 3

Analysis

In this chapter, we describe a method to automatic extraction landmarks from biological images. This method was proposed by Palaniswamy[4] and the process includes some steps as follows:

1. Segmentation (includes preprocessing image and extracting the features),
2. Construction and comparison the Pairwise Geometric Histogram,
3. Estimation the global pose of object by the Probabilistic Hough Transform,
4. Refinement the estimated landmarks by template matching.

3.1 Segmentation

Segmentation is a process to extract interested features (lines) from the digital image. The expected result in this process is the list of the approximate lines which are used to construct the pairwise geometric histogram.

The process mainly separate into two stages: firstly, we pre-process image; secondly, we extract the image's features. In first stage, we reduce the noise in image by finding a threshold value and applying the thresholding technique to obtain the interested features. In second stage, we extract the features based on the edge segmentation. By applying the appropriate technique to obtain the step edges and broken the edges into approximate lines.

3.1.1 Preprocess image

In this application, we use the thresholding technique to pre-process the image. In thresholding technique, using a threshold value **"t"**, we can decrease the noise and obtain the interested features. The threshold value can be defined by the histogram analysis.

Based on the histogram of the original image, we compute the mean and the median of this histogram. With the histogram obtained, we split it into two parts: the first part starts from the bin 0 to the limit value (the limit value is the smallest value between mean and median); the second part, starts from the limit value to the end of the histogram. For each part, we find the maximum, minimum value and calculating the mean of it. The value "t" obtained by the mean of the two mean values in two parts of histogram.
With the threshold value "t", we apply the threshold technique to pre-process image in the CV_THRESH_BINARY mode (keep the pixel has value greater than threshold value).

Algorithm 1: Algorithm to preprocess image

Data: inputImage: the input image

Result: outputImage: the image after processing

```
Convert the input image into gray scale image;
Calculate the histogram on gray scale image and store the result in histogram
variable ;
Compute the mean value and median value of histogram;
limit ← (mean > median ? median : mean);
limitSub ← ((limit >= 120) ? (limit − 25) : (limit − 5));
Declare some variables: int imax ← −1, max ← −1;
for i ← 0 to limitSub do
    if histogram[i] > max then
        max = histogram[i];
        imax = i;
Declare some variables: int imin ← −1, min ← max;
for k ← imax to limit do
    if histogram[k] < min then
        min = histogram[k];
        imin = k;
Declare some variables: int max2 ← −1, imax2 ← −1;
for j ← limit to end_of_histogram do
    if histogram[j] > max2 then
        max2 = histogram[j];
        imax2 = j;
middle1 ← (imax1 + imin)/2 ;
middle2 ← (imax2 + imin)/2 ;
middle ← (middle1 + middle2)/2 ;
Apply the threshold with threshold value is middle;
```

3.1.2 Feature extraction

After applying the threshold to pre-process image, we apply the Canny algorithm to detect the step edges, which incorporates non-maximal suppression and hysteresis thresholding. In

Canny, the important parameters are the two threshold values and the aperture size of the Sobel operator, it decides the pixels kept. The threshold value used in Canny algorithm also the value used in the previous step, and the ratio between lower threshold and upper threshold is 1 : 3 (follows the article [4] but have to be modified). In our implementation, the Canny operation used from OpenCV library[1], and the parameters need to be provided into Canny are:

- source: the input image (in grayscale mode)
- destination: the output image,
- low_thresh: the first (lower) threshold value,
- hight_thresh: the second (upper) threshold value,
- kernel_size: size of kernel, aperture for the Sobel operator.

The Canny algorithm is not aware of the actual edges, the edge detecting process was based on the Sobel operator, extracted with non-maximal suppression. To obtain the expecting result, we apply another technique to obtain the step edges. The **findContours** was chosen for this goal, the result is a vector of the edges, and each edge was presented by a vector of the points. Like the Canny, the **findContours** uses OpenCV library [2] and the parameters used in this operation are as follows:

- source: the binary input image,
- contours: the output. Each contours is stored in a vector of points,
- hierarchy: optional output vector, containing information about the image topology,
- mode: contours retrieve mode,
- method: contours approximation method,
- offset: optional offset by which every contour point is shifted.

3.1.3 Edge segmentation

The geometric relation could not be constructed from the edges, it is always constructed from the relation of basic geometric objects, such as the lines. In fact, any arbitrary edge can be

[1] http://docs.opencv.org/modules/imgproc/doc/feature_detection.html#canny

[2] http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#findcontours

represented by a set approximate lines. Instead an edge, we represent a set of approximate lines of it. This method is useful when we want to present the edges or describe the relationship between the lines. From the set of step edges was obtained from find contours (the image structure), in this step, we will segment each step edge into approximated lines. The method to segment the edges is the recursive algorithm[5] but it has some changes in the "stop condition" of the algorithm to simplify, as follows:

- Establish a line *"l"* between two endpoints of the edge.
- For each point on edge, we compute the perpendicular distance from it to the line l and keep the point which has the maximum perpendicular distance.
- If the maximum perpendicular distance from a point on edge to the line l is greater than α, then the edge is splited at this point. The value chosen for α in the program is 3 ($\alpha = 3$).
- Reprocess both parts which was obtained from step 3.
- The algorithm continues until all edges fragments are represented.

The algorithm is presented as follows:

Algorithm 2: Algorithm to segment an edge

Data: listPoints: list of points which presented the edge

Result: Queue of “step” points on the edge

Declare the first endpoint: $p0 \leftarrow listPoints[0]$;

Declare the second endpoint: $pend \leftarrow listPoints[size - 1]$, $size$ is the size of $listPoints$;

Set up a straight line between the two endpoints $p0, pend$ (line d);

Initialization the max value: $maxDistance \leftarrow 0$;

Declare a “split point”: $imax \leftarrow 0$;

Declare a variable: $distance \leftarrow 0$;

for *point p in listPoints* **do**

  $distance \leftarrow$ from p to line d;

  if $distance > max\_distance$ **then**

    $maxDistance \leftarrow distance$;

    $imax \leftarrow$ position of p;

if $maxDistance > 3$ **then**

  split the list of points at $imax$ and put into 2 parts ($part1, part2$);

  Pre-process on $part1$;

  Pre-process on $part2$;

if $p0$ *does not exist in result queue* **then**

  push $p0$ into queue;

  `// queue is a variable of class`

if *pend does not exist in result queue* **then**

  push $pend$ into queue;

  `// queue is a variable of class`

3.2 Pairwise Geometric Histogram

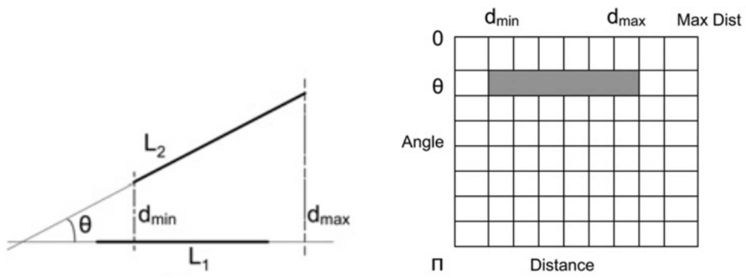
Pairwise Geometric Histogram(PGH)[2] is used to encode the relative information between a line and a set of lines in an object. Therefore, an object can be represented by a set of PGH. From the set of PGH, we can reconstructed the object or compare to another object. In this section, we introduce the construction of a PGH for an object based on the geometrical relationship and compute the similar distance between two objects.

3.2.1 Local pairwise geometric histogram

The PGH is constructed on the geometric features between lines relative. The geometric features are characteristic which can describe the geometric shape such as angle, the length of line, perpendicular between two lines, ... For the shape representation, the relative angle and perpendicular distance is geometrical features useful.

The proceed to construct the PGH between two lines was described in below:

- Choose the reference line (other lines called object lines),
- Compute the angle between the reference line and the object lines,
- Calculate the perpendicular distance from the two endpoints of an object lines to the reference line (assigned dmin and dmax),
- Recording the perpendicular distance and the angle relation between reference line and the object lines into the two-dimensional histogram.



(a) The geometric relationship between two lines

(b) The pairwise geometric histogram

Figure 3.1: Example about geometric features and the pairwise geometric histogram

The local PGH presented the relationship between a reference line with other lines of shape. Thus, for an object, we can construct a local PGH for each line in object. The frequency of the geometric features is recorded as a two dimensional histogram with an angle axis (0 - π) and distance axis (range of perpendicular distance, d_{max} is the maximum distance on all distance of two arbitrary lines). The entries on PGH describes the geometric relationship between the reference line and the object lines. The blurring of entry along the axis regarding the true position and orientation of each object lines for reference line. Following the accuracy, we can indicate the size of histogram and normalize the value to match with size of histogram.

3.2.2 Global pairwise histogram

Based on local PGH constructor, global pairwise histogram is combined of all local PGHs of all lines belong to the object. It means if the object is defined by n lines, the global pairwise histogram will composed of n local PGHs. This method is good when we apply some variants on the image, such as translate or rotate the image because the angle and perpendicular distance between a pair of lines are invariant.

3.2.3 Histogram matching

"The histogram matching enables robust classification of shape features by finding similarity between the scene and reference model"[4]. The similar between two models can obtain via the similar distance, which is computed by comparing their probability distribution on geometric histogram. In our program, each image is represented by a comprises of many geometric histograms and uses the Bhattacharya metric to determine the similar distance between the two models [4]. In general, we have normalize the histograms before comparing. The form of Bhattacharrya metric used to compute the degree of 2 model:

$$d_{Bhattacharyya}(H_iH_j) = \sum_{\theta}^{\pi} \sum_{d}^{d_{max}} \sqrt{H_i(\theta, d)H_j(\theta, d)} \tag{3.1}$$

The significance of parameters in the formula 3.1, as follows:

- θ: angle value, range of θ in angle axis from 0 to π.
- d: the perpendicular distance, range of d in perpendicular distance from 0 to the maximum distance of arbitrary lines of shape.
- $H_i(\theta, d)$ is an entry at row θ and column d in histogram of image i
- $H_j(\theta, d)$ is an entry at row θ and column d in histogram of image j

Based on the accuracy of the program, we can change the range of the angle and distance axis. Besides the Bhattacharya metric, we may choose another metric to matching the histograms, such as: **Chi-squared** metric and **Intersection** metric. The forms are presented as below:

Chi-squared metric:

$$d_{Chi-squared}(H_iH_j) = \frac{\sum_{\theta}^{\pi} \sum_{d}^{d_{max}} \left(\frac{(H_i(\theta,d)-H_j(\theta,d))^2}{(H_i(\theta,d)+H_j(\theta,d))}\right)}{2} \tag{3.2}$$

Intersection metric

$$d_{Intersection}(H_iH_j) = \sum_{\theta}^{\pi} \sum_{d}^{d_{max}} min(H_i(\theta, d), H_j(\theta, d)) \tag{3.3}$$

The significance of parameters in formula (3.2) and (3.3) are similar to formula (3.1). For the Bhattacharyya and Intersection metric, the perfect match is 1 and the total mismatch is 0. The result is opposite to Chi-squared metric (0 for perfect match and 1 for total mismatch).

Hence, depending on the purpose of the comparison subject will choose a suitable comparing method. In this program, we propose three methods to obtain a general result when matching the histograms.

3.3 Probabilistic Hough transform

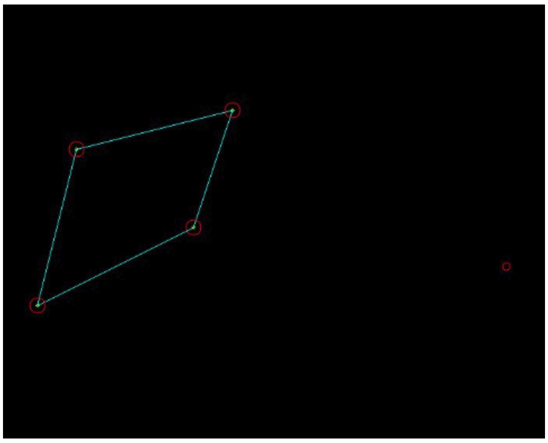
The Probabilistic Hough Transform (PHT) is used to estimate the global pose of shape[1]. Based on a group of features within the scene, identifying the represent of a model image in a scene image. The hypothesised location of the model in the scene is indicated based on the conditional probability that any pair scene lines agreement about a position in model.

Estimating the global shape has two main stages. Firstly, training process starts with recording the perpendicular distance and the angle from a reference point to each pair of model lines. Secondly, predicting the pose of scene different from the model, then we estimate the location of the landmarks. We create a Hough space to store value if exist a pair of scene lines matching to the entry in the training process. The peak in Hough space is assumed as the reference point of the model in the scene image. From this reference point, we can estimate the reference landmarks of reference image on the scene image. The process to estimate the global pose includes the steps as follows:

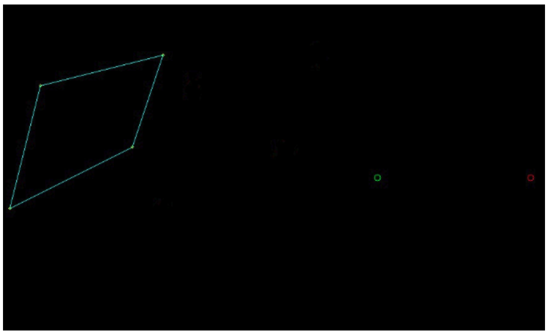
- Choose an arbitrary point in the model as a reference point,
- For each pair lines in the model, calculating and recording the perpendicular distance and angle from the reference point to each line,
- Create an two-dimensional accumulator, one dimension for the angle and the other for the perpendicular distance,

- For each pair lines in the scene, finding the entry correspond to the position, orientation and scale. Increasing the value at correlative cell in the accumulator (indicate by the angle and distance),
- Compute the maximum value in the accumulator,
- Indicating the pair of scene lines and the entry with maximal value of accumulator,
- Extending the perpendicular lines of the pair belong to scene lines at the appropriate position. The intersection of them is the location of the reference point in the scene.

In the example below, we apply the PHT to estimate the landmarks of the model to the scene. The image in figure 5.2a as the model. In the model, the small red circle and large red circles are the reference point and the landmarks in model, respectively. The image in figure 5.2b as scene. By applying the PHT, we estimate the reference point (green circle) in the scene and the location of the landmarks (the yellow circles).



(a) The model image



(b) The scene image

Figure 3.2: The landmarks estimated by probabilistic Hough transform

3.3.1 Training process

In this step, recording the perpendicular distance and angle from each pair model lines to a reference point (called reference table). The reference point can be chosen at arbitrary position on the model image. To reduce the time complexity processing the next step, we consider the "closet pair lines". In this application, the reference point chosen at the center of image and the closet pair lines are pair of lines have all three conditions: the length of each line greater than 60 pixels, the lines are not parallel and the distance between them is less than 5 pixels.

The algorithm considers a pair of closet lines and constructs the reference table as follows:

Algorithm 3: Algorithm to consider the closet lines

Data: line1 (the first line), line2 (the second line)

Result: Two line closet or not (bool)

$distance1 \leftarrow$ distance from the first endpoint of $line1$ to $line2$;
$distance2 \leftarrow$ distance from the second endpoint of $line1$ to $line2$;
if *$line1.length()$ > 60 and $line2.length()$ >60*
and $line1$ not parallel with $line2$
and (distance1 <= 5 or ditance2 <= 5) **then**
    return true;
return $false$;

Algorithm 4: Algorithm to construct the reference table

Data: lines (a list of lines), refPoint (the reference point)

Result: The reference table

Declare the reference table $refTable$;
for *line i in lines.size()* **do**
    for *line j in lines.size()* **do**
        if *i != j and line(i) closet with line(j)* **then**
            Compute the angle and perpendicular distance from $line(i)$ to $refPoint$;
            Compute the angle and perpendicular distance from $line(j)$ to $refPoint$;
            Create an entry to store pair of lines and its information ;
            Add the entry into reference table ;
return $refTable$;

3.3.2 Estimating process

The estimating process is duration that estimating the reference landmarks on the scene image. Firstly, we find the reference point on scene image. Secondly, we estimate the reference landmarks on the scene image from the position of reference point in the scene image.

By finding a pair of scene lines agree with a pair of model lines, we can detect the position of the reference point on scene image. We create an accumulator to store each agreement between the pair of scene lines and the pair of model lines. For each pair of scene lines, we find its exist in the reference table and increase the value at correspondence position in accumulator. At the end, we obtain a pair of scene lines and pair of model lines correspondence with the maximum value in accumulator. By extending the perpendicular lines of the pair of scene lines at the appropriate position, we meet the reference point at the intersection.

In our program, two pair lines are supposed to be agreement if the angle difference between them is less than one degree, the length scale is less than 1 and the distance between two pair of lines is less than two. Two followed algorithms describe the definition between the two pair lines and finding the position of reference point in scene image.

Algorithm 5: Algorithm to check the agreement between two pair lines

Data: line1 (the first reference line), line2 (the second reference line), sline1 (the first scene line), sline2 (the second scene line)

Result: Two pair lines similar or not (boo)

$angle1 \leftarrow$ angle between $line1$ and $line2$;

$angle2 \leftarrow$ angle between $sline1$ and $sline2$;

$mdistance \leftarrow$ sum of perpendicular distance from two endpoints of $line1$ to $line2$;

$sdistance \leftarrow$ sum of perpendicular distance from two endpoints of $sline1$ to $sline2$;

if $abs(ange1 - angle2) < 1$

$and\ abs(line1.length()/sline1.length() - line2.length()/sline2.length()) < 1$

$and\ abs(mdistance - sdistance) < 2$ **then**

  return $true$;

return $false$;

Algorithm 6: Algorithm to find the agreement of pair scene lines in model

Data: refTable (the reference table), slines (pair of scene lines)

Result: The entries in reference table similar with pair of scene lines

Declare the return entries in reference table $entries$;

for $entry\ et\ in\ refTable$ **do**

  if *agree between lines in entry and slines* **then**

    put the entry et into list of entries $entries$;

return $entries$;

Algorithm 7: Algorithm to find the reference point in scene

Data: lines (a list of scene lines), refTable (reference table)

Result: The reference table

Create an accumulator, *acc*;
Declare the reference table $refTable$;
for *line i in lines.size*() **do**
    for *line j in lines.size*() **do**
        if $i\,!= j$ *and line*(i) *closet with line*(j) **then**
            Find the agreement of pair scene lines in mode;
            Increase the value in *acc* with correspondence position;
            Marked the maximum value, pair of scene lines and entry in reference table;
Find the intersection ($intersect$) between two perpendicular lines with pair scene lines at appropriate position;
`// The appropriate position is correct with the distances in reference table.`
return $intersect$;

By finding the reference point, the landmarks in the scene image can be estimated by calculating the relatedness between the reference point and the reference landmarks. Besides, we also record the difference about rotation, orientation and scale between the model image and the scene image.

3.4 Template matching

Template matching is the process to refine the estimated landmarks, which was obtained by PHT of the scene image with an appropriate method.

3.4.1 Cross-correlation

Cross-correlation is a method of estimating the similarity between the two signals. By computing the sum of products between two signals when sliding, and choose the maximal value. It is used for searching a short signal in a longer signal. In image processing, it used to detect the present of an object (template) in a large object (image). The equation of cross-correlation is as follows (equation 3.6):

$$R_{ccorr}(x,y) = \sum_{x',y'} [T(x'.y').I(x+x',y+y')] \tag{3.4}$$

Where:

- T is template which use to slide and find the exist in other image.
- I is image which we expect to find the template image
- (x',y') are coordinates in template where we get the value to compute.
- $(x+x',y+y')$ are coordinates in image where we get the value to compute when template T sliding.

By sliding the template on image by each pixel from left to right and top to down. At each position, we compute the $R_{ccorr}(x,y)$. The position have maximal $R_{ccorr}(x,y)$ is positioned that best similar of template in image.

However, if we use the original image to compute and find the similarity, the brightness of the template and the image might change the conditions and the result. So, we can normalize the image before applying the cross-correlation to reduce the effect of lighting difference between them. The normalization coefficient is:

$$Z(x,y) = \sqrt{\sum_{x',y'} T(x'.y')^2 . \sum_{x',y'} I(x+x',y+y')^2} \tag{3.5}$$

The value of this method when we normalized computation as below:

$$R_{ccorr\_norm}(x,y) = \frac{R_{ccorr}(x,y)}{Z(x,y)} = \frac{\sum_{x',y'} [T(x'.y').I(x+x',y+y')]}{\sqrt{\sum_{x',y'} T(x'.y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \tag{3.6}$$

3.4.2 Template matching

Back to our problem, with a reference image and its set of landmarks. We use the cross-correlation to refine the landmarks. In this case, the template is a region around each landmark in reference image and the image is also a region around the Hough landmark detection in scene image. Hence, to save the processing time, before applying the cross-correlation, the scene image is rotated to match with model using Hough estimate.

For each landmark in the reference image, we create a bounding box around the landmarks with an arbitrary size and use landmark as a center point. When create the bounding box, we need to keep the distance between left corner to the landmarks, because sometimes, with the landmark position, the size of bounding box can be over the size of image. Use this box as a template and do the cross-correlation with each scene image. The results obtained store the location where the template matches the image. From these position, we indicate the position of each landmark of reference image on scene image. The algorithm to create the bounding box around a landmark is described follows:

Algorithm 8: Algorithm to create a bounding box around a landmark

Data: image (reference image), landmark (location of a reference landmark), tsize (size of bounding box), distance (to keep the distance from the landmark to bounding box)

Result: A matrix represented for bounding box of landmark

Get the matrix of image (image presented by matrix):
$Mat matImg = image.getMatrix();$

`// Indicate the top left-corner of bounding box:`

$int\ lx = (landmark.x - tsize/2) < 0\ ?\ 0 : (landmark.x - tsize/2);$

$int\ yx = (landmark.y - tsize/2) < 0\ ?\ 0 : (landmark.y - tsize/2);$

`// Keep the distance from the landmark to bounding box`

$distance.x = landmark.x - lx;$

$distance.y = landmark.y - ly;$

`// Indicate the low right-corner of bounding box`

$int\ lx2 = (landmark.x + tsize/2) > matImg.cols\ ?\ matImg.cols :$
$(landmark.x + tsize/2);$

$int\ yx2 = (landmark.y + tsize/2) < matImg.rows\ ?\ matImg.rows :$
$(landmark.y + tsize/2);$

`// Create the bounding box around landmark`

$Mat\ box(matImg, Rect(lx, ly, lx2 - lx, ly2 - ly));$

return the box;

The belows algorithm describe a method to estimate the reference landmarks on a scene image by using cross-correlation. Before applying the cross-correlation, the scene image is rotated to match with the model. The angle used to rotate is the sum of the difference between the scene line and model line to which it matched and the difference between the two pairs of the similar lines. To apply the cross correlation, we have used the function *matchTemplate*[3] in OpenCV library with matching method is CV_CCORR_NORMED (cross-correlation normalize). This function allows us compare the template overlaping the image and it supports many different matching methods. When the template slide over each pixel on image, the coefficient between them is calculated and stored in a array.

After finished correlation, to get the value and the position of the maximum value when we compute the coefficient, we use a function in OpenCV, *minMaxLoc*[4]. This method is used to detect the minimum and maximum value in an array. Beside that, it also output the location where having the minimum and maximum value.

---

[3] http://docs.opencv.org/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#matchtemplate

[4] http://docs.opencv.org/modules/core/doc/operations_on_arrays.html?highlight=minmaxloc#minmaxloc

Algorithm 9: Algorithm to get the position of reference landmarks in scene image

Data: refImage (reference image), sceneImage (the scene image), lmpath (file path store the reference landmarks)

Result: A list of landmarks on scene image

```
Get the reference landmarks from file and store in list refLandmarks;
Create a variable to store the new landmarks: sceneLandmarks;
Estimate the reference landmarks (refLandmarks) in scene image using probabilistic
  Hough transform and save into a variable: esLandmarks;
// Get the matrix of scene image
sceneMatrix = sceneImage.getMatrix();
Rotate the scene matrix with appropriate angle;
for variable i in esLandmarks.size() do
    // Get the reference landmark
    Point refPoint = refLandmarks.at(i);
    // Create a bounding box of reference landmark refPoint
    Mat template = createTemplate(refImage, refPoint, size);
    // Get the estimate landmark
    Point esPoint = esLandmarks.at(i);
    // Create a bounding box of estimate landmark esPoint
    Mat sceneImg = createTemplate(sceneImage, esPoint, size);
    Create the matrix to store the value when do the cross-correlation: result ;
    // Apply the matching and store the result into matrix result
    cv :: matchTemplate(sceneMatrix, template, result, CV_TM_CCORR_NORMED);
    // Get the maximum value and position in result matrix
    double maxValue, minValue;
    Point maxLoc, minLoc;
    cv :: minMaxLoc(result, &minValue, &maxValue, &minLoc, &maxLoc, Mat());
    Compute the position of landmark from maximum position;
    Push the landmark into the list sceneLandmarks;
Return the list of landmarks;
```

Chapter 4

Conception

This chapter describes the implementation to extract the landmarks from biological images. This work is connected to a previous one which was implemented some operations about image segmentation by NGUYEN Hoang Thao.

4.1 Software architecture

Extending the previous version of software, the functions of morphometry are saved in the **impls_2015** package of the program. Besides the methods created by myself, I use some methods from the OpenCV (library for image processing) and the Qt framework (framework for C++).

The class diagram[1] in 4.1 shows main classes of my task. The *mainly* methods located in the `ImageViewer` class, where contains all functions of the software. To represent the information of image and preprocessing about clear the yellow grid, we use classes such as `Line`, `Edge`, `Landmark`, `YellowGird`, `Image`. For the edge segmentation, construct the pairwise geometric histogram, probabilistic hough transform and landmarks detection, we have `GFeatures`, `LocalHistogram`, `ShapeHistogram`, `EdgeSegmentation`, `PHTransform` `LandmarkDection`. The main functions were inherated from the abstract classes `HistogramMethod`, `SegmentMethod`, `HoughMethod`, `LandmarkMethod` respective and used in `main` class (`ImageViewer`) via the `Sceneario` class.

[1]See the full image in Appendix

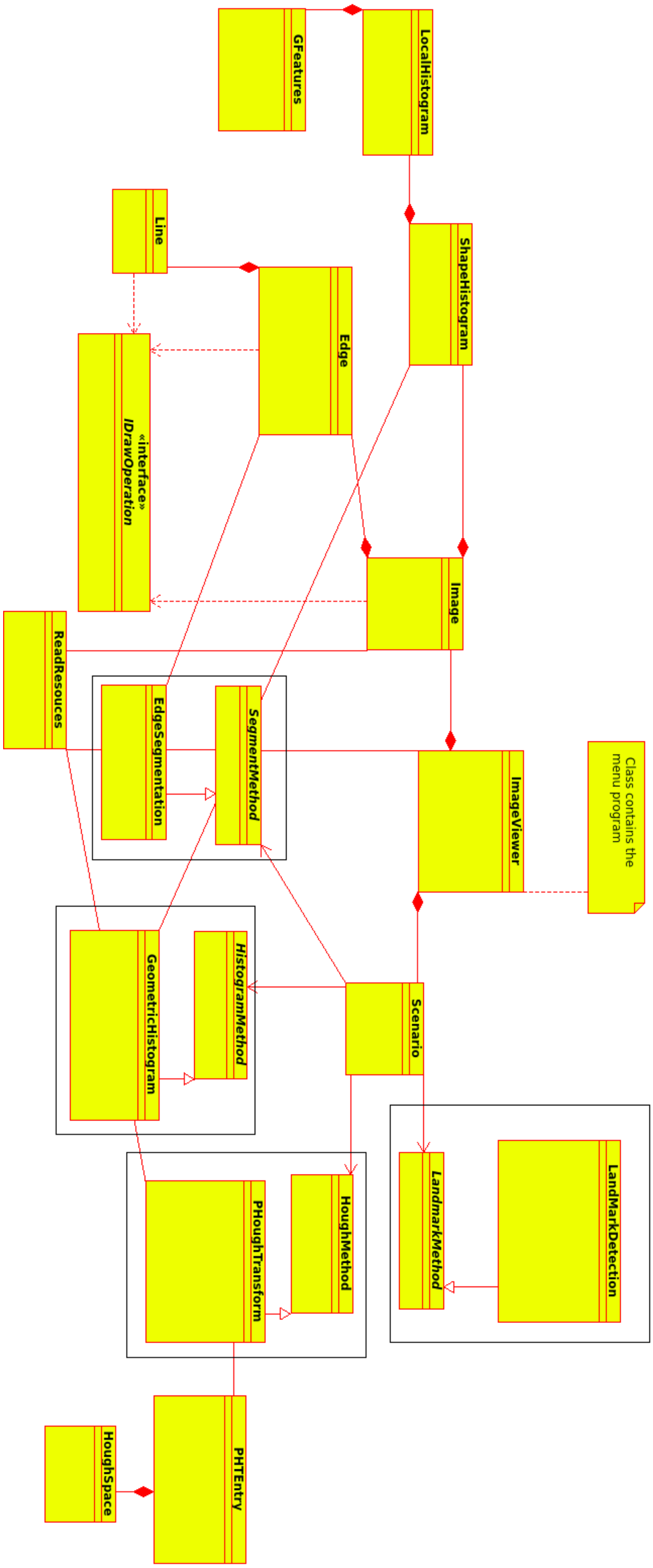


Figure 4.1: The class diagram diagram

4.2 Image preprocessing

The *Image processing* section describes information about the classes which describe the geometric objects that can be represent the image and the method to remove the yellow grid on the images.

The `Line` class descibes the information of a straight line and its method, such as: get the length of line, compute the perpendicular distance from a point to line, find the intersection between two lines, compute the angle between two lines, find the parallel line with this line.

The `Edge` class uses to present a curve and the methods with edge. An edge can be presented by a list of lines or a list of points. The important methods in `Edge` class are `breakEdge()` and `segment()` method. It used to break the edge into approximate lines based on the list of point constructed edge.

The `Image` class presents the information of an image such as file name, list of edge extracted from it. Besides, `Image` class also provides the methods to get the file name of image, compute the histogram of image, remove the yellow grid (if it exist) on image, get the PGHof image, read its landmarks from a file, etc.

4.3 The abstract classes

The abstract classes contains the abstract methods get the actions on image such as segmentation, PGH construction,.... The methods are implemented by the inherit classes, respective and provide the way access to action for other classes. The abstract classes include: `HistogramMethod` class, `SegmentMethod` class, `HoughMethod` class, `LandmarkMethod` class.

4.4 Edge segmentation

Edge segmetation includes classes used to segment an image. Besides, the classes construct the edge which is described in previous section such as `Line, Edge,...`, we also provide the access methods for other classes.

The `EdgeSegmentation` class provides the methods such as obtaining the lines from an image, present the result of segmentation or applying the segmentation on an image folder. The methods in `Edge segmentation` are:

- Extract the approximate lines of object in an image,

- Extract the approximate lines of object in each image in a folder.

4.5 Pairwise Geometric Histogram

This section describes the classes used for PGH construction process.

`GFeatures` class contains the relative information of the objects in PGH such as angle, minimum distance and maximum distance. It provides the methods to get and set the relative information.

`LocalHistogram` class is constructed to contain the information when computing the PGH of a line in object. The chosen lines as reference lines, the local histogram is constructed based on recording the relating between reference line and other lines in object. Besides, it has the methods for the user to change the accuracy, such as the angle accuracy or the distance accuracy.

`ShapeHistogram` class constructs the PGH for an object. It is constructed by on combing all PGH of the lines in an object. It also provides the methods to compute the measured distance between the pairwise geometric histograms by a matching method. The methods in this class includes:

- Construct the PGH for an image,
- Construct the matrix to save the PGH result,
- Compute the measured distance between the two PGHs based on *Bhattacharyya*, *Chi-Squared* or *Intersection* metric.

`GeometricHistogram` class provides the access ways for other classes. By usign this class, user can compute the pairwise geometric histogram of an image and calculate the distance between the pairwise geometric histograms.

4.6 Estimate the global pose (Probabilistic Hough Transform)

This section describes the classes use probabilistic hough transform to estimate the model image from a scene image. In particular, it is estimating the reference landmarks on the scene image.

`HoughSpace` class contains the information about the angle and distance from a line to a reference point. These information is recorded to construct the accumulator when we apply the Probabilistic Hough Transform.

`PHTEntry` class presents each entry when constructing the reference table in the training process. Each entry contains the pair of lines and its information about angle and distance to a reference point.

`PHoughTransform` class describes the main process when we apply the probabilistic hough transform to estimate the landmarks. It includes the methods to construct the reference table, find the reference point in scene image and estimate the landmarks. Besides, it also provides the methods to estimated the landmarks of an image on the directory of images.

4.7 Refine the landmarks

`LandmarkDetection` class provides the methods to refine the landmarks. It uses cross-correlation technique to refine the estimated landmarks. Besides, we might compute the centroid point of an object.

Chapter 5

Realisation and Conclusions

5.1 New interface

The results of this internship are integrated into IPM[1] software, but the software interface was reorganized. Besides the functions of previous version, the clients can be use new functions to segment image or automatic extraction landmarks from image.

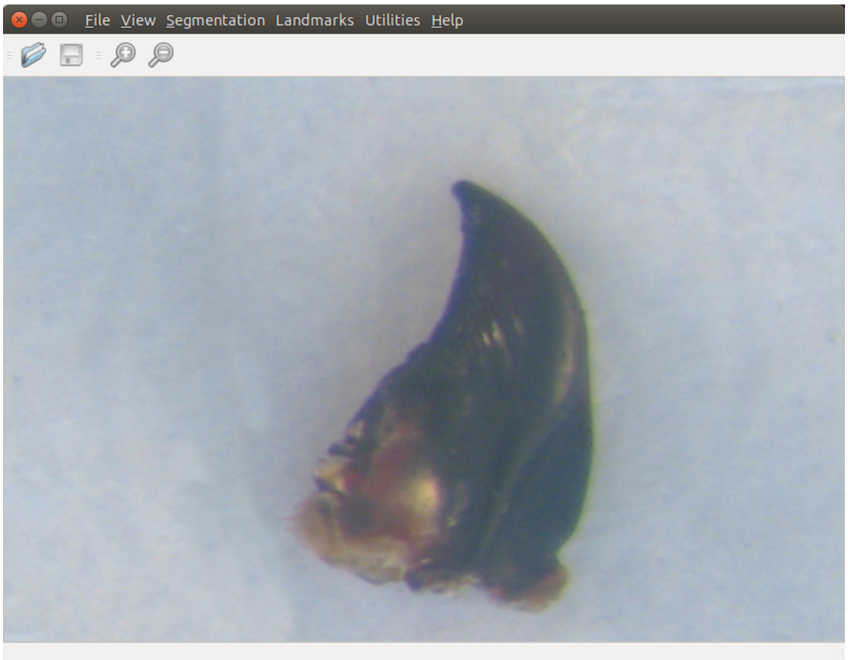


Figure 5.1: The graphic user interface of IPM software

The software provides adequately functions to process on biological image. Specially, the menu **Landmarks** contains the main result of this internship. It provides clients the functions auto-

[1]Image Processing for Morphometrics

matically estimation landmarks and analysis the result from estimated landmarks. The process can be finished on an image or a list of images.
Besides, other functions also developed and integrated into IPM software, as followed:

- Image segmentation by analysing histogram (in **Segmentation** menu)
- Removing the yellow grid on image (in **Utilities** menu)
- Loading the manual landmarks (in **Utilities** menu)
- Computing the pairwise geometric histogram (PGH) of image (in **Utilities** menu)
- Computing the measure metric between PGHs by Bhattacharya, Chi-Squared or Intersection method (in **Utilities** menu)

5.2 Experimentation

The experimentation is deployed on a machine equipped with Intel(R) Core(TM) i7-4790 CPU 3.6GHz, 16 GB of RAM. The testing data is two set of biological images: *Right mandible and left mandible*. Each dataset includes 293 images(3264 x 2448). However, the data was filtered by suppressing the bad images in the both datasets. The bad images includes the empty images and broken images (images contain the broken object). They are showed as below:

- Md 004.JPG
- Md 146.JPG
- Md 238.JPG
- Mg 003.JPG
- Mg 007.JPG
- Mg 040.JPG
- Mg 066.JPG
- Mg 159.JPG
- Mg 248.JPG
- Mg 292.JPG

5.2.1 Parameters

In our program, we use these parameters for the methods:

- The best segmentation obtained from choosing a good threshold value. In the program, Canny algorithm is applied to segment the image. Thus, the ratio between *lower threshold : upper threshold* is important to get a good result. And the ratio is: *1 : 3* (in class `Image`, method `getEdges`), this ratio has been chosen experimentally. The lower value is 1 * *threshold* value and the upper value is 3 * *threshold* value. The *threshold* value is identified by analysing the histogram of image.

- The angle and distance accuracy used in constructing the PGH matrix and calculate the measure distance between PGHs. The angle accuracy can be 90 (0.5 * 180), 180, 360 (2 * 180), 720(4 * 180), 1080(6 * 180), 2160(12 * 180) degree. The distance accuracy can be 250, 500 or 1000 columns. The **default value** in program is **180** degree for angle accuracy, and **250** for the distance accuracy.

- During applying the Probabilistic Hough Transform, to reduce the time complexity during training, we consider the pair of closet lines. And the parameters used to indicate the closet line are (used in method `closetLine`, class `PHoughTransform`):

  - Length of each line greater than **60** pixels
  - Angle between two lines greater than **15** degree
  - Perpendicular distance from one of two endpoints of a line to other line less than **5** pixel.

  The conditions to predicate two pairs of lines are similar (used in method `similarPairLines`, class `PHoughTransform`):

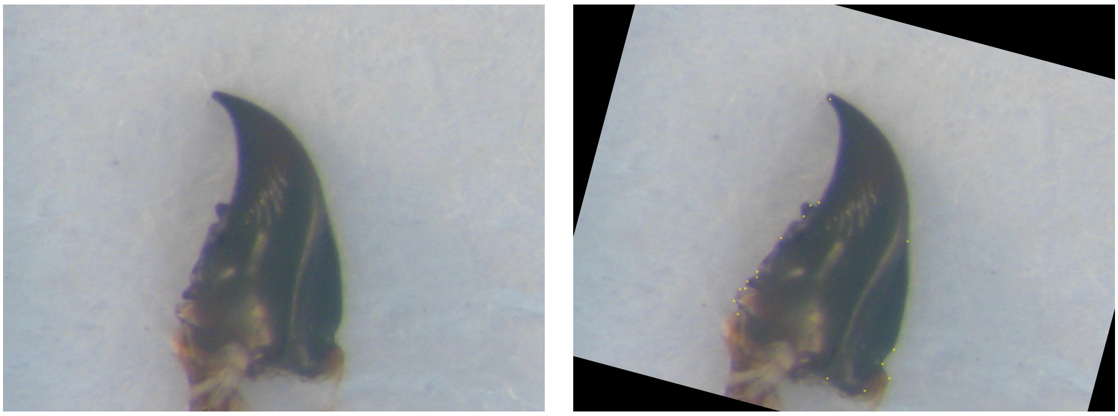
  - Subtraction between angle of two pair of lines is less than **1**
  - Subtraction between ratio couple of scene lines and reference lines is less than **1**
  - Subtraction between distance of two pair of lines is less than **2**

- The size of bounding box around the reference landmarks used for estimating landmarks by cross-correlation method or computes the estimated centroid is *400* pixels (used in method `crossCorrelation and crossCorrelationDistance`, class `ImageViewer`)

- The size of bounding box around reference landmarks and estimated landmarks used to refine the estimated landmarks or compute the estimated centroid are *400* pixels and **1400** pixels, respective.(used in method `getLandmarks and tplMatchingDistance`, class `ImageViewer`) To increase the flexible of program, all parameters was placed in the resources files (**data/resources** folder). For each group of parameters, the parameters are put in a file.

5.3 Results

The automated landmark identification is examined on two data sets: *right mandible and left mandible*. And the landmarks are extracted: 18 landmarks for each *right mandible* image, 16 landmarks for each *left mandible* image.



(a) The scene image (b) The scene image with estimated landmarks

Figure 5.2: Automatic identification of the landmarks

The accuracy of the system can be determined by comparing the differences(in pixels) between the landmarks located by this method and the manual landmarks. This method has to pass several steps, the result of each step will effect on next steps. Thus, to evaluate the accuracy of this method, we can evaluate the result of each step.

5.4 Conclusions

The landmarks are important characteristics used in shape analysis of many biological and medical applications. The method proposed in this internship report can be used to estimate the landmarks on biological images. These estimated landmarks can be compared with the result estimated by cross-correlation.

After finishing the internship, we have implemented the proposed method by using the OpenCV library and the Qt framework on C++. And the test was done on two set of biological images: *right mandible and left mandible*. A feature work could be apply this method on other dataset of biological images.

Bibliography

[1] Anthony Ashbrook, Neil A Thacker, Peter Rockett, and CI Brown. Robust recognition of scaled shapes using pairwise geometric histograms. In *BMVC*, volume 95, pages 503–512, 1995.

[2] Alun Evans, Neil A Thacker, and John EW Mayhew. The use of geometric histograms for model-based object recognition. In *BMVC*, volume 93, pages 429–438. Citeseer, 1993.

[3] Jos Luis Espinosa Aranda Jesus Salido Tercero Ismael Serrano Gracia Noelia Vllez Enano Gloria Bueno Garca, Oscar Deniz Suarez. *Learning Image Processing with OpenCV*. Packt Publishing, 2015.

[4] Sasirekha Palaniswamy, Neil A Thacker, and Christian Peter Klingenberg. Automatic identification of landmarks in digital images. *IET Computer Vision*, 4(4):247–260, 2010.

[5] Neil A Thacker, PA Riocreux, and RB Yates. Assessing the completeness properties of pairwise geometric histograms. *Image and Vision Computing*, 13(5):423–429, 1995.