



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

December 22, 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 useful to extract the information and knowledge about phenomena as medical or biological processes. To obtain the best result, most of the applications must follow two processes: firstly, pre-processing the images with some appropriate operators to enhance the interest and to reduce the noises. Secondly, applying the measures or classification procedures to obtain the main results.

The goal of this project is to build the program fully functional program about processing base on the biological images in the context of Morphometry. During my internship at LaBRI, my tasks was developing the algorithm to automatic identify the landmarks of the biological images and to classify a set of beetle in 2D dimensions. The method based on the probabilistic Hough transform. The article is "**Automatic identification of landmarks in digital images**".

At the end of the internship, I integrated the implementation of the article into the Image Processing for Morphometrics (MorphoEcol) software, which was developed previously by NGUYEN Hoang Thao (another PUF Master Degree). Besides, we also debug the previous code and write the documentation for the next phases.



# Contents

<b>Introduction</b>	<b>5</b>
<b>1 Context</b>	<b>6</b>
1.1 Pôle Universitaire Français . . . . .	6
1.2 Laboratoire Bordelais de Recherche en Informatique . . . . .	7
1.3 The Internship . . . . .	7
<b>2 Analysis</b>	<b>10</b>
2.1 Segmentation . . . . .	10
2.1.1 Pre-process image . . . . .	10
2.1.2 Feature extraction . . . . .	11
2.1.3 Edge segmentation . . . . .	12
2.2 Pairwise Geometric Histogram . . . . .	12
2.2.1 Local pairwise geometric histogram . . . . .	13
2.2.2 Global pairwise geometric histogram . . . . .	13
2.2.3 Histogram matching . . . . .	13
2.3 Probabilistic Hough Transform . . . . .	14
2.3.1 Training process . . . . .	15
2.3.2 Estimating process . . . . .	15
2.4 Template matching . . . . .	16
<b>3 Conception</b>	<b>18</b>
3.1 Features extraction . . . . .	18
3.1.1 Image preprocessing . . . . .	18
3.1.2 Edges extraction . . . . .	20
3.1.3 Edge fragmentation . . . . .	21

3.2	Features description (Pairwise Geometric Histogram) . . . . .	22
3.2.1	Local PGH constructor . . . . .	22
3.2.2	Global PGH constructor . . . . .	22
3.2.3	PGH matching . . . . .	22
3.3	Global pose estimation (Probabilistic Hough Transform) . . . . .	24
3.3.1	Training process . . . . .	24
3.3.2	Estimation process . . . . .	27
3.4	Landmarks refinement . . . . .	27
<b>4</b>	<b>Realisation and Result</b>	<b>30</b>
4.1	General model of software . . . . .	30
4.1.1	Segmentation module . . . . .	31
4.1.2	PGH module . . . . .	31
4.1.3	PHT module . . . . .	31
4.1.4	Landmark detection module . . . . .	31
4.2	Software architecture and the packages . . . . .	32
4.2.1	Edge segmentation . . . . .	34
4.2.2	Pairwise Geometric Histogram . . . . .	34
4.2.3	Estimate the global pose (Probabilistic Hough Transform) . . . . .	35
4.2.4	Refine the landmarks . . . . .	35
4.3	Result . . . . .	36
4.3.1	New interface . . . . .	36
4.3.2	Experimentation . . . . .	37
4.3.3	Parameters . . . . .	38
4.3.4	Results . . . . .	39
4.3.5	Limits and future works . . . . .	41
<b>Conclusion</b>		<b>42</b>
<b>Bibliography</b>		<b>44</b>

# List of Figures

1.1	The parts of beetle . . . . .	9
2.1	Example about noises in image . . . . .	11
2.2	An example about image segmentation . . . . .	12
2.3	Example about geometric features and the pairwise geometric histogram . . . . .	13
2.4	The landmarks estimated by probabilistic Hough transform . . . . .	16
4.1	The modules of program with main classes . . . . .	30
4.2	The packages of program . . . . .	32
4.3	The class diagram of program . . . . .	33
4.4	The graphic user interface of MAELab software . . . . .	36
4.5	The MAELab software with its menu . . . . .	37
4.6	Automatic identification of the landmarks . . . . .	40
7	An example about histogram . . . . .	45

# Introduction

Morphometry is a concept refers to qualitative analysis of form, it includes the size and the shape of an object. Morphometry analyses are commonly performed on organism. An objective of morphometry is to statistic hypotheses based on the effect of shape. Often, it was applied on the organism objects or the animal objects; but in scope of this field, we consider morphometry on biological images. Generally, to measure the object's morphometry, we can use one of three morphometry forms: traditional morphometry, landmark-based geometric morphometry, outline-based morphometry.

In this work, we consider landmark-base geometric morphometry. Our goal is to implement the methods to extract the landmarks automatically from biological images and evaluate the results by statistic method. To archive the landmarks identification, the method of Palaniswamy[9] has been chosen. The goal is to test if the landmarks provided by this method enough good that we can replace the manual landmarks. The application has been done on a set of beetle images which provided by biologists.

The whole report has four chapters. In the first chapter, this is the short introduction about the context working and the objectives of my internship. Chapter 2, introduces about the method which we use to extract the landmarks on biological images. In the third chapter, we mention how to apply the method for our problems. In the chapter 4, we will focus about the software. This chapter introduces the modules, the architecture and the packages used to build the software. Besides, we give our examination, results and the summary 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'Université National du Vietnam - Ha Noi located in Ha Noi capital (PUF-Ha Noi) and Pôle Universitaire Français de l'Université National du Vietnam - Ho Chi Minh Ville located in Ho Chi Minh city (PUF-HCM).

#### **PUF-HCM**

PUF-HCM<sup>1</sup> is a department of VietNam National University 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.

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 Capitole
- Bachelor and Master of Informatics: linking program with University of Bordeaux and University of Paris 6.

---

<sup>1</sup><http://pufhcm.edu.vn>

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)<sup>2</sup> 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

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 environment. It shows the ability synthesis, evaluation and self-research of student. Besides, the student

---

<sup>2</sup><http://www.labri.fr>

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 a part of collaboration between LaBRI and INRA Rennes, the project is working on the agriculture field. The goals of the project is tracking, collecting and classifying the insects based on the morphometry of them, specific beetle. Biologists from Rennes have collected a set of 293 beetles and produced a series of images for the different parts of beetle such as head, body, right mandible, left mandible and pronotum (see in figure 1.1). The main request from the biologist team is: *"How to produce automatically landmarks on each image that could be equivalent to manual landmarks (landmarks set by hand) ?"*.

*"Morphometric landmarks are points that can be defined in all specimens and located precisely"*[9]. They are widely used in many biological and medical applications. Presently, the landmarks are mainly extracted manually. Doing that automatically remains a challenge.

The objectives of this internship is implementing a method to automatic extract landmarks. The method based on the article: **“Automatic identification of landmarks in digital images”**, which proposed by Palaniswamy[9]. Besides, an evaluation about accuracy of method is done based on the comparing between the estimated landmarks and the manual landmarks which provided by biologists.

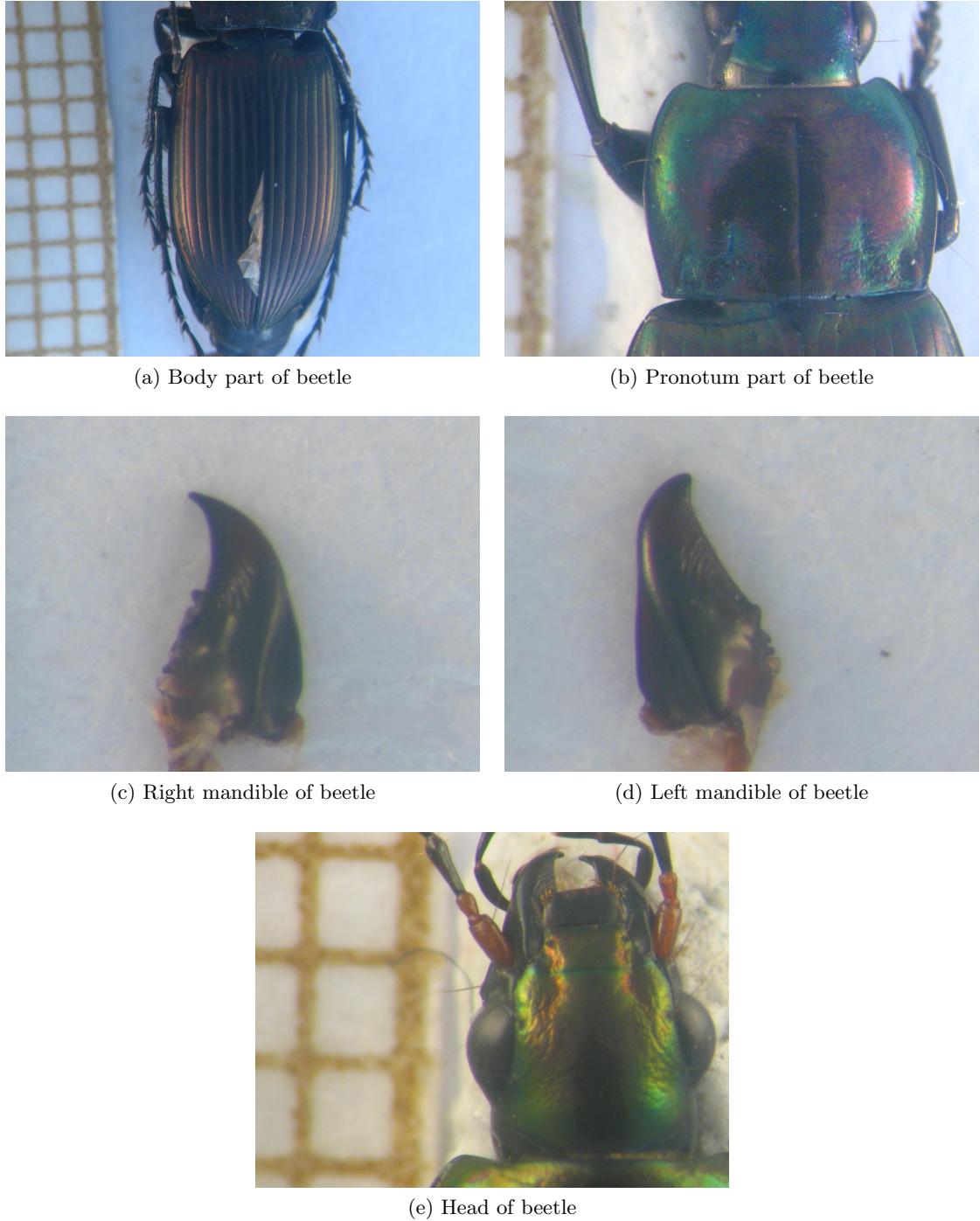


Figure 1.1: The parts of beetle

# Chapter 2

## Analysis

In this chapter, we present some generic concept of image processing and the Palaniswamy's method to automatic extraction landmarks from biological images. The method is based on 4 steps:

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.

### 2.1 Segmentation

Segmentation is a process to extract interested features from the digital image. The expected result in this stage is the list of approximate lines which are presented for the image's features. The process mainly separate into two steps: firstly, we pre-process image to reduce the noise and enhance the quality of features in image. Secondly, we extract the image's features (edges) and present it as list of approximate lines.

#### 2.1.1 Pre-process image

To finish the pre-processing step, we apply the **threshold** method. This way is different with the suggested method in the article[9] (DoG filter) to pre-process image. Usually, the threshold value might indicated follows two ways: manually and automatically. In our case, the threshold

value was indicated by analysing the image histogram. This method will be focused in next chapter.

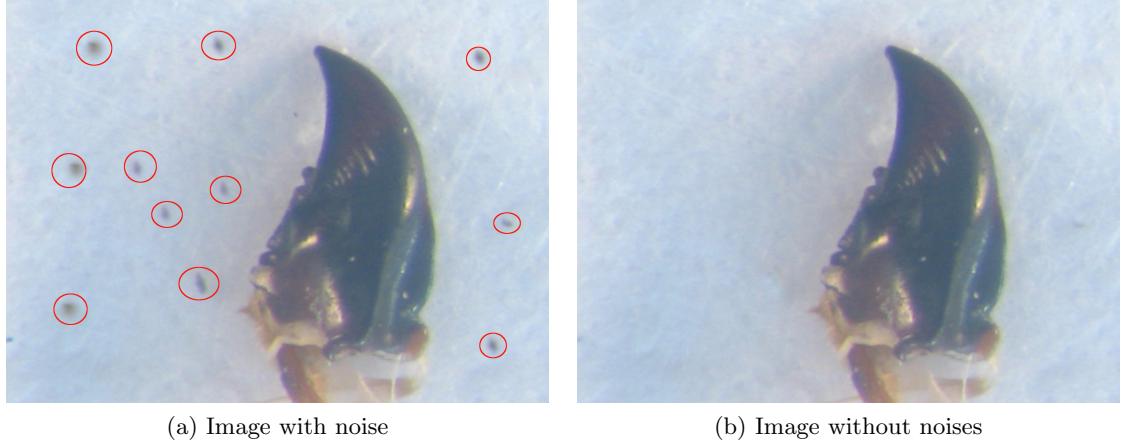


Figure 2.1: Example about noises in image

### 2.1.2 Feature extraction

After reducing the noise from input image in pre-processing step, we can extract the interested in features (edges) from the image. The Canny[3] algorithm, which incorporates non-maximal suppression and hysteresis thresholding, is used to detect the edges. The inputs which we need to know when applying the Canny algorithm as follows:

- *Source*: the input image (in grayscale mode)
- *Destination*: the output image,
- *Lower threshold*: the first (lower) threshold value,
- *Upper threshold*: the second (upper) threshold value,
- *Kernel size*: size of kernel, aperture for the Sobel operator.

In Canny algorithm, the important parameters are the two threshold values (*lower threshold* and *upper threshold*) because it will decide which pixels are kept. Normally, the value of lower threshold and upper threshold are determined according to a certain rate.

Actually, 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 need to apply another technique to obtain the step edges. In this case, we can use the technique to analysis the structure of topology to get the edges. This technique was proposed by Suzuki[10].

### 2.1.3 Edge segmentation

The geometric relation of an object could not be constructed from edges, it is always constructed from the relation of basic geometric objects, such as the lines. In fact, any arbitrary edge can be represented by a set of approximate lines. This presentation is useful when we want to describe the relationship between the edges in an object. The method used to fragment the edge into a list of approximate lines is the recursive algorithm[11], it is similar with the method proposed by Lowe[8] except the stop condition changed. An example about edge segmentation is presented in figure 2.2.

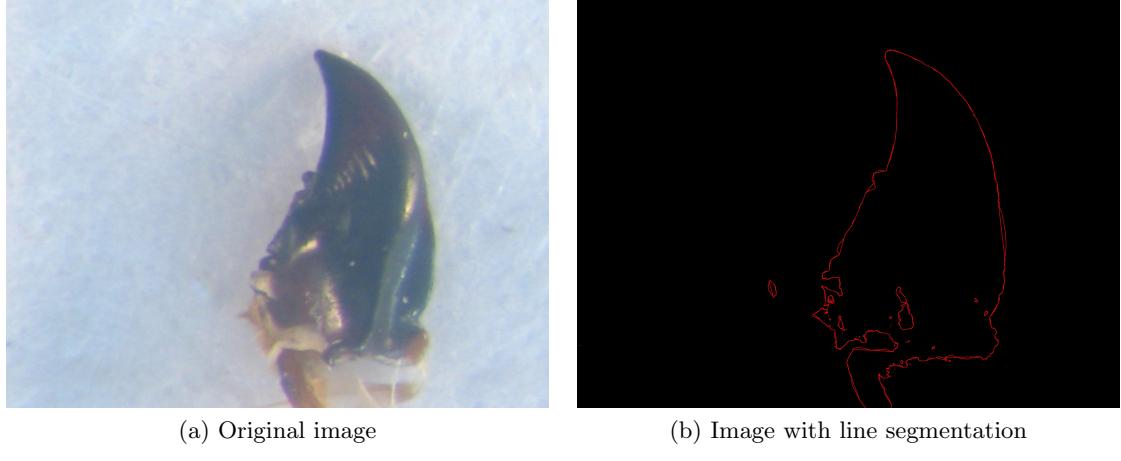


Figure 2.2: An example about image segmentation

## 2.2 Pairwise Geometric Histogram

Pairwise Geometric Histogram(PGH)[5] 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. The object is described by set of approximate lines which were obtained in the previous stage.

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 distance between two lines, etc. For the shape representation, the relative angle and perpendicular distance is geometrical features useful.

As example in figure 2.3<sup>1</sup>, the figure 2.3a represents the geometric relationship between two

---

<sup>1</sup>Image extracts from the article

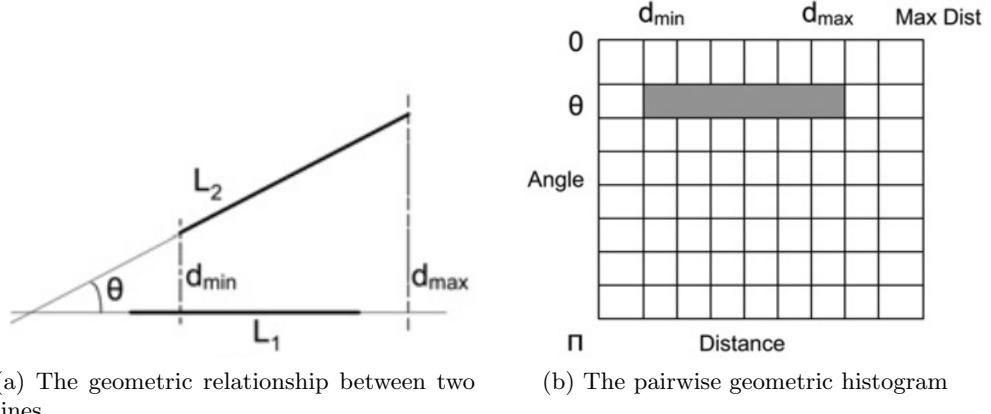


Figure 2.3: Example about geometric features and the pairwise geometric histogram

lines **L1**, **L2**. The figure 2.3b represents the PGH between them when using line **L1** as reference line and line **L2** as object line.

### 2.2.1 Local pairwise geometric histogram

The local PGH presented the relationship between a reference line with other lines of the object. Thus, for each line in object, we can construct a local PGH for it. The frequency of the geometric features is recorded as a two dimensional histogram with an angle axis and distance axis. 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”*[9]. Following the accuracy, we can indicate the size of histogram and normalize the value to match with size of histogram.

### 2.2.2 Global pairwise geometric histogram

Based on local PGH constructor, global PGH 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 PGH 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.

### 2.2.3 Histogram matching

*“The histogram matching enables robust classification of shape features by finding similarity between the scene and reference model”*[9]. The similar between two models can obtain via the

similar distance, which is computed by comparing their probability distribution on geometric histogram. To solve it, each model is represented by a global PGH and uses the Bhattacharya[11] metric to determine the similar distance between the two models [9]. In general, we can be normalize the PGH before comparing. The form of Bhattacharrya metric used to compute the similarities between two models is:

$$d_{Bhattacharyya}(H_i H_j) = \sum_{\theta}^{\pi} \sum_{d}^{d_{max}} \sqrt{H_i(\theta, d) H_j(\theta, d)} \quad (2.1)$$

The significance of parameters in the formula 2.1, as follows:

- $\theta$ : angle value, range of  $\theta$  in angle axis from 0 to  $\pi$ .
- $d$ : 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  $\theta$  and column  $d$  in PGH of image  $i$
- $H_j(\theta, d)$  is an entry at row  $\theta$  and column  $d$  in PGH of image  $j$

### 2.3 Probabilistic Hough Transform

**Hough transform** is a technique used to find the exists of an object in another object. The main idea of this method based on the “vote” procedure. The Hough transform was originally developed to recognize the line[7] and after, it has extended to apply on arbitrary shape[4].

The **generalised Hough transform**[2] (GHT) is the modification of Hough transform. It can used to detect an object described with its model. The problem solved with GHT is finding the model’s position in the image. Like **Hough transform**, GHT also use the vote procedure to solve the problem. Following the similarity metric (by Bhattacharya), the hypothesised matches can used as input in pose estimation algorithm such as the generalised Hough transform. But in this case, it do not apply. Instead, the probabilistic Hough transform (PHT) is used to estimate the global pose of the object[1] in object.

Based on a group of features within the scene image, identifying the represent of a model image in a scene image. The hypothesised location of the model image in the scene image is indicated based on the conditional probability that any pair scene lines agreement about a position in

model image.

Estimating the global pose of the object has two main steps. Firstly, training process starts with recording the perpendicular distance and the angle from a reference point to each pair of model lines, then finding the similar pairs between model image and scene image. Secondly, estimating process starts as predicting the pose of scene image different from the model image, then we estimate the location of the model landmarks on scene image.

### 2.3.1 Training process

The training process begin with choosing a reference point in model. This point can be chosen at arbitrary position on the model image. At this step, the perpendicular distance and angle from each pair model lines to a reference point was recording and saving (called reference table). To finish the training process, we find exist of the model in the scene image by PHT. For each pair of scene lines, we find the pair of model lines (in reference table) which similar with the pair of scene lines and increase the value in accumulator at respectively location. The accumulator is a two-dimension matrix, one axis presents for angle, the other axis presents for the perpendicular distanc. Finally, we choose the similar pair between model image and scene image. The chosen pair of scene lines obtained from best votes when we consider the similarity between each pair of scene image and model image.

### 2.3.2 Estimating process

The estimating process is duration that estimating the reference landmarks on the scene image. Firstly, we estimate the position of model's reference point on scene image. Secondly, we estimate the model's reference landmarks on the scene image from the position of model's reference point in the scene image. With a pair of scene lines agree with a pair of model lines was chosen at previous step, the model's reference point in the scene image can be detected by extending the perpendicular lines of the pair of scene lines at the appropriate position[1].

The estimation of model's landmarks in the scene image can be estimated by calculating the relatedness between the model's reference point and the model's landmarks. Besides, we also record the difference about rotation, orientation and scale between the model image and the scene image.

In figure 2.4, we apply the PHT to estimate the landmarks of the model to the scene. The image in figure 2.4a as the model. In the model, the small red circle and large red circles are the

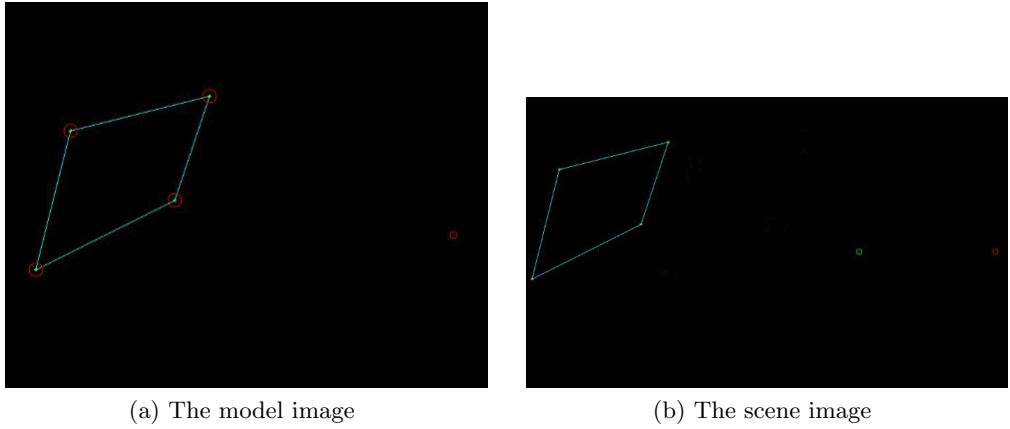


Figure 2.4: The landmarks estimated by probabilistic Hough transform

reference point and the landmarks in model, respectively. The image in figure 2.4b 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).

## 2.4 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. In the scope of this method, we use the cross-correlation to refine the estimated landmarks from PHT stage.

Cross-correlation is a method of estimating the similarity between the two signals (searching a short signal in a long signal). By computing the sum of products between two signals when a signal is sliding on another signal. The position is considered similarity if the value at this position is maximal. 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 2.4):

$$R_{ccorr}(x, y) = \sum_{x',y'} [T(x',y').I(x + x', y + y')] \quad (2.2)$$

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:

$$Z(x, y) = \sqrt{\sum_{x',y'} T(x'.y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2} \quad (2.3)$$

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

$$R_{ccorr\_norm}(x, y) = \frac{R_{ccorr}(x, y)}{Z(x, y)} = \frac{\sum_{x',y'} [T(x'.y').I(x + x', y + y')]}{\sqrt{\sum_{x',y'} T(x'.y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}} \quad (2.4)$$

# Chapter 3

## Conception

Previous chapter has discussed about the method which we used to estimate the landmarks on biological image. This chapter describe different algorithms inside this method.

### 3.1 Features extraction

#### 3.1.1 Image preprocessing

As discuss in previous, we use the thresholding technique to pre-process the image. As we know about thresholding technique, with a threshold value “ $t$ ”, we can decrease the noise and obtain the interested features. The output of this technique is a binary image, which can be used to extract the features. The threshold value are 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 beginning 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 calculate the mean of them. The threshold value “ $t$ ” obtained by the mean of the two mean values in two parts of histogram (as algorithm 1).

To implement the threshold technique, we apply the threshold method with *BINARY* mode (keep the pixel has value greater than threshold value) to pre-process image.

**Algorithm 1:** Algorithm to get the threshold value and pre-process image

**Data:** inputImage: the input image  
**Result:** outputImage: the image after processing

- 1 Convert the input image into gray scale image;
- 2 Calculate the histogram on gray scale image and store the result in *histogram* variable ;
- 3 Compute the *mean* value and *median* value of histogram;
- 4  $limit \leftarrow (mean > median ? median : mean);$
- 5  $limitSub \leftarrow ((limit >= 120) ? (limit - 25) : (limit - 5));$
- 6 Declare some variables: *int imax*  $\leftarrow -1$ , *max*  $\leftarrow -1$ ;
- 7 **for**  $i \leftarrow 0$  to *limitSub* **do**
- 8     **if** *histogram*[*i*]  $> max$  **then**
- 9         *max*  $\leftarrow histogram[i];$
- 10         *imax*  $\leftarrow i;$
- 11     **end**
- 12 **end**
- 13 Declare some variables: *int imin*  $\leftarrow -1$ , *min*  $\leftarrow max$ ;
- 14 **for**  $k \leftarrow imax$  to *limit* **do**
- 15     **if** *histogram*[*k*]  $< min$  **then**
- 16         *min*  $\leftarrow histogram[k];$
- 17         *imin*  $\leftarrow k;$
- 18     **end**
- 19 **end**
- 20 Declare some variables: *int max2*  $\leftarrow -1$ , *imax2*  $\leftarrow -1$ ;
- 21 **for**  $j \leftarrow limit$  to *end\_of\_histogram* **do**
- 22     **if** *histogram*[*j*]  $> max2$  **then**
- 23         *max2*  $\leftarrow histogram[j];$
- 24         *imax2*  $\leftarrow j;$
- 25     **end**
- 26 **end**
- 27  $middle1 \leftarrow (imax1 + imin)/2 ;$
- 28  $middle2 \leftarrow (imax2 + imin)/2 ;$
- 29  $middle \leftarrow (middle1 + middle2)/2 ;$
- 30 Apply the threshold with threshold value is *middle*;

The parameter provides to **threshold** method are:

- *Input image*: the input image (in grayscale mode)
- *Output image*: the output image
- *Threshold value*: the value **t** used to threshold image
- *Maximum value*: the maximum value to use with **BINARY** operations
- *Threshold operation*: the operation used to threshold image (we use **BINARY** mode)

### 3.1.2 Edges extraction

The edges extraction are finished follows two steps:

- Edge extraction by applying Canny algorithm
- Edge retrieve by applying function to get the contours.

The threshold value used in Canny algorithm is the value used in the previous step (pre-process image), and the ratio between lower threshold and upper threshold is 1 : 3 (follows the article [9] but have to be modified).

The value of parameters in Canny algorithm are:

- Source image: the input image after pre-processing (in grayscale mode),
- Destination image: the output image,
- Lower threshold value: the lower threshold value ( $t$ ),
- Upper threshold value: the upper threshold value ( $3t$ ),
- Kernel size: size of kernel, aperture for the Sobel operator (five pixels).

To retrieve the contours after applying the Canny algorithm, we apply the method to analysis the structure of topology to get the edges. This method will return to a list of the edges, and each edge was presented by a list of the points. The parameters used in find contours method are as follows:

- Source: the binary input image (output of Canny algorithm),
- 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 fragmentation

The steps of the method used to fragment the edges 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  $\alpha$ , then the edge is split at this point. The value chosen for  $\alpha$  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.

And the algorithm is presented as follows (algorithm 2):

<b>Algorithm 2:</b> Algorithm to segment an edge into approximate lines
---

**Data:**  $listPoints$ : list of points which presented the edge

**Result:** Queue of “step” points on the edge

```

1 Declare the first endpoint:  $p0 \leftarrow listPoints[0]$ ;
2 Declare the second endpoint:  $pend \leftarrow listPoints[size - 1]$ , size is the size of
 $listPoints$ ;
3 Set up a straight line between the two endpoints  $p0, pend$  (line  $d$ );
4 Initialization the max value:  $maxDistance \leftarrow 0$ ;
5 Declare a “split point”:  $imax \leftarrow 0$  ;
6 Declare a variable:  $distance \leftarrow 0$ ;
7 for  $point p$  in  $listPoints$  do
8    $distance \leftarrow$  from  $p$  to line  $d$ ;
9   if  $distance > max\_distance$  then
10    |  $maxDistance \leftarrow distance$ ;
11    |  $imax \leftarrow$  position of  $p$ ;
12  end
13 end
14 if  $maxDistance > 3$  then
15  | split the list of points at  $imax$  and put into 2 parts ( $part1, part2$ );
16  | Pre-process on  $part1$ ;
17  | Pre-process on  $part2$ ;
18 end
19 if  $p0$  does not exist in result queue then
20  | push  $p0$  into queue;
21  | // queue is a variable of class
22 end
23 if  $pend$  does not exist in result queue then
24  | push  $pend$  into queue;
25  | // queue is a variable of class
26 end
```

## 3.2 Features description (Pairwise Geometric Histogram)

The PGH is represented as two-dimension matrix. An axis presents for angle information (rows) and another presents for perpendicular distance information (columns). Based on the accuracy of requirements, the range of angle and distance axis can be changed. In default, the range of angle axis and distance axis are  $(0 - \pi)$  and 500, respectively.

### 3.2.1 Local PGH constructor

The proceed to construct the local PGH was described in below:

- Choose the **reference line** (other lines called **object lines**),
- For each object line, compute the angle between it and reference line (assigned **angle**); and the perpendicular distance from two endpoints of object line to reference line (assigned **dmin** and **dmax**),
- Recording the perpendicular distance and the angle relation between reference line and the object lines into the matrix (PGH histogram). The recording was finished by marking the cells at row **angle** and from column **dmin** to column **dmax**.

### 3.2.2 Global PGH constructor

The duration to construct the global PGH also similar to local PGH. Especially, the information of all local PGH are presented in the same matrix.

### 3.2.3 PGH matching

Follows the analysis, the Bhattacharya metric used to compute the similarity between two models. The models were presented by the global PGH which should be have the same size (width and height). The algorithm to compute the Bhattacharya metric between two models is described as below:

**Algorithm 3:** Algorithm to compute the measure metric between 2 PGH using Bhattacharya

**Data:**

- model: the global PGH of model
- scene: the global PGH of scene

**Result:** Measure distance between model and scene

```

1 width ← the width of model (or scene) PGH;
2 height ← the height of model (or scene) PGH ;
3 modelEntries ← the total entries of model PGH;
4 sceneEntries ← the total entries of scene PGH;
5 double distance, distance1, distance2;
6 for size_t i from 0 to width do
7   for size_t j from 0 to height do
8     distance1 = sqrt( model[i][j]/modelEntries );
9     distance2 = sqrt( scene[i][j]/sceneEntries );
10    distance += distance1 * distance2;
11  end
12 end

```

Besides the Bhattacharya metric, we may choose another metric to find the similarity the histograms, such as: **Chi-squared** metric and **Intersection** metric. The forms are presented as below:

**Chi-squared metric:**

$$d_{Chi-squared}(H_i H_j) = \frac{\sum_{\theta}^{\pi} \sum_{d}^{d_{max}} \left( \frac{(H_i(\theta, d) - H_j(\theta, d))^2}{(H_i(\theta, d) + H_j(\theta, d))} \right)}{2} \quad (3.1)$$

**Intersection metric**

$$d_{Intersection}(H_i H_j) = \sum_{\theta}^{\pi} \sum_{d}^{d_{max}} min(H_i(\theta, d), H_j(\theta, d)) \quad (3.2)$$

The significance of parameters in formula (3.1) and (3.2) are similar to formula (2.1). 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 Global pose estimation (Probabilistic Hough Transform)

The core of PHT uses vote procedure to find the model's position in the scene image. The process to estimate the model's landmarks on scene image 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. And save the information into a table,
- 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 model's reference point in the scene image.

#### 3.3.1 Training process

The training process starts with creating the model's **reference table**. Each entry in reference table includes pair of model lines and geometric information (angle and perpendicular distance) from each line to reference point. To reduce the time complexity processing during training process, we consider the “closest pair lines”. The algorithm considers a pair of closest lines and constructs the reference table as follows (algorithm 4 and 5):

**Algorithm 4:** Algorithm to consider the closest lines

**Data:** line1 (the first line), line2 (the second line)  
**Result:** Two line closet or not (bool)

```
1 distance1  $\leftarrow$  distance from the first endpoint of line1 to line2;  
2 distance2  $\leftarrow$  distance from the second endpoint of line1 to line2;  
3 if line1.length()  $> 60$  and line2.length()  $> 60$   
4 and line1 not parallel with line2  
5 and (distance1 <= 5 or ditance2 <= 5 ) then  
6 | return true;  
7 end  
8 return false;
```

**Algorithm 5:** Algorithm to construct the reference table

**Data:** lines (a list of lines), refPoint (the reference point)  
**Result:** The reference table

```
1 Declare the reference table refTable ;  
2 for line i in lines.size() do  
3 | for line j in lines.size() do  
4 | | if i  $\neq$  j and line(i) closet with line(j) then  
5 | | | Compute the angle and perpendicular distance from line(i) to refPoint;  
6 | | | Compute the angle and perpendicular distance from line(j) to refPoint;  
7 | | | Create an entry to store pair of lines and its information ;  
8 | | | Add the entry into reference table ;  
9 | | end  
10 | end  
11 end  
12 return refTable ;
```

The training process is finished by finding a pair of scene lines agree with a pair of model lines. An accumulator was created 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.

The below algorithms describe the steps to finish the training process (algorithm 6, 7, 8):

**Algorithm 6:** 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 (bool value)

```

1 angle1  $\leftarrow$  angle between line1 and line2;
2 angle2  $\leftarrow$  angle between sline1 and sline2;
3 mdistance  $\leftarrow$  sum of perpendicular distance from two endpoints of line1 to line2;
4 sdistance  $\leftarrow$  sum of perpendicular distance from two endpoints of sline1 to sline2;
5 if abs(angle1 - angle2) < 1
6 and abs(line1.length() / sline1.length() - line2.length() / sline2.length()) < 1
7 and abs(mdistance - sdistance) < 2 then
8   | return true;
9 end
10 return false ;

```

**Algorithm 7:** 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

```

1 Declare the return entries in reference table entries ;
2 for entry et in refTable do
3   | if agree between lines in entry and slines then
4     |   | put the entry et into list of entries entries;
5   | end
6 end
7 return entries ;

```

**Algorithm 8:** Algorithm to find similar pairs between model image and scene image

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

**Result:** The pair of scene lines which has the best vote and the corresponding entry in reference table

```

1 Create an accumulator, acc;
2 Declare the reference table refTable ;
3 for line i in lines.size() do
4   | for line j in lines.size() do
5     |   | if i != j and line(i) closet with line(j) then
6       |     |     Find the agreement of pair scene lines in mode;
7       |     |     Increase the value in acc with correspondence position;
8       |     |     Marked the maximum value, pair of scene lines and entry in reference table;
9     |   | end
10   | end
11 end
12 Indicate the pair of scene lines has the best vote;
13 Indicate the pair of model lines correspondence with pair of scene lines;

```

### 3.3.2 Estimation process

At the end of training process, we obtained similar pairs between model image and scene image. In this step, the model reference point estimated by extending the perpendicular lines of the pair of scene lines at the appropriate position. The algorithm to find the model reference point in scene image as follows (algorithm 9):

**Algorithm 9:** Algorithm to find position of model reference point in scene image

**Data:** pair of scene lines, pair of model lines (entry in reference table)

**Result:** The position of model reference point in scene image

- 1 Find the match lines between pair of scene lines and pair of model lines;
- 2 Find two lines which parallel with pair of scene lines that the distance between them following the distance in entry;
- 3 Find the intersection between two parallel lines;

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 Landmarks refinement

Refining landmarks is finished by applying the cross-correlation. In this case, the template is a region around each landmark in model image and the image is also a region around the Hough landmark detection (the landmarks estimated in previous stage) in scene image. Hence, to save the processing time, before applying the cross-correlation, the scene image is rotated to match with model. The difference angle between the model image and the scene image obtained from Hough estimate in previous stage.

For each landmark in the model 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*. Doing the same with each estimated landmarks on scene image but using a larger size to create the *small image*. Then applying the cross-correlation between *template* and *small 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 10):

**Algorithm 10:** 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

```
1 Get the matrix of image (image presented by matrix):
  MatmatImg = image.getMatrix();
2 // Indicate the top left-corner of bounding box:
3 int lx = (landmark.x - tsize/2) < 0 ? 0 : (landmark.x - tsize/2);
4 int yx = (landmark.y - tsize/2) < 0 ? 0 : (landmark.y - tsize/2);
5 // Keep the distance from the landmark to bounding box
6 distance.x = landmark.x - lx;
7 distance.y = landmark.y - ly;
8 // Indicate the low right-corner of bounding box
9 int lx2 = (landmark.x + tsize/2) > matImg.cols ? matImg.cols :
  (landmark.x + tsize/2);
10 int yx2 = (landmark.y + tsize/2) < matImg.rows ? matImg.rows :
  (landmark.y + tsize/2);
11 // Create the bounding box around landmark
12 Mat box(matImg, Rect(lx, ly, lx2 - lx, ly2 - ly));
13 return the box;
```

The below algorithm (algorithm 11) describe a method to refine the estimated landmarks on a scene image by using cross-correlation. Remember that, before applying the cross-correlation, the scene image is rotated to match with the model; and 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.

Cross correlation method allows us compare the template overlapping the image. 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. 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.

**Algorithm 11:** 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

```
1 Get the reference landmarks from file and store in list refLandmarks;  
2 Create a variable to store the new landmarks: sceneLandmarks;  
3 Estimate the reference landmarks (refLandmarks) in scene image using probabilistic  
Hough transform and save into a variable: esLandmarks;  
4 // Get the matrix of scene image  
5 sceneMatrix = sceneImage.getMatrix();  
6 Rotate the scene matrix with appropriate angle;  
7 for variable i in esLandmarks.size() do  
8     // Get the reference landmark  
9     Point refPoint = refLandmarks.at(i);  
10    // Create a bounding box of reference landmark refPoint  
11    Mat template = createTemplate(refImage, refPoint, size);  
12    // Get the estimate landmark  
13    Point esPoint = esLandmarks.at(i);  
14    // Create a bounding box of estimate landmark esPoint  
15    Mat sceneImg = createTemplate(sceneImage, esPoint, size);  
16    Create the matrix to store the value when do the cross-correlation: result ;  
17    // Apply the matching and store the result into matrix result  
18    cv :: matchTemplate(sceneMatrix, template, result, CV_TM_CCORR_NORMED);  
19    // Get the maximum value and position in result matrix  
20    double maxValue, minValue;  
21    Point maxLoc, minLoc;  
22    cv :: minMaxLoc(result, &minValue, &maxValue, &minLoc, &maxLoc, Mat());  
23    Compute the position of landmark from maximum position;  
24    Push the landmark into the list sceneLandmarks;  
25 end  
26 Return the list of landmarks;
```

# Chapter 4

## Realisation and Result

This chapter describes the model, the architecture and the modules of MAELab (Morphometry automatically extraction LaBRI) software. The first part is description about model as well as the modules of software. Then, we give the software architecture, class diagram and explain about it.

### 4.1 General model of software

The software includes four main modules corresponding to four processes to estimate the landmarks: segmentation module, pairwise geometric histogram (PGH) module, probabilistic Hough transform (PHT) module and landmark detection module. Besides, the environment module provides the common classes. The classes in environment module can be used in all modules of software (figure 4.1).

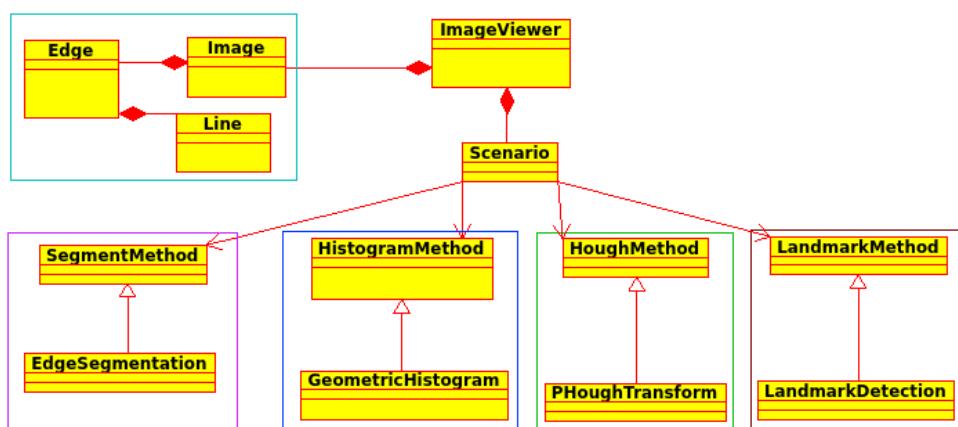


Figure 4.1: The modules of program with main classes

## **Environment module**

Environment module contains the classes used to describe the information presented for object in image: `Image`, `Edge`, `Line`. The classes in this module can be used for all modules in MAELab.

### **4.1.1 Segmentation module**

The segmentation module implements the pre-process image and extracts the edges from image. The main classes in this module includes `SegmentMethod` class and `EdgeSegmentation` class. `SegmentMethod` class is designed as an abstract class of segmentation, this is the connecting port between this module and other modules. Class `EdgeSegmentation` extends from `SegmentMethod`, it implements the method to pre-process and extract the features from image.

### **4.1.2 PGH module**

The PGH module provides the ways to implement the constructor of PGH and compute the measure metric between PGHs. In PGH module, class `HistogramMethod` is a abstract class, it provides the method to construct the PGH of an image. Its methods was implemented in `GeometricHistogram` class (`GeometricHistogram` extends from `HistogramMethod`). Basically, the method to construct the PGH come from other classes. Class `GeometricHistogram` uses these methods to finish the implementation from abstract class. In addition, `GeometricHistogram` also provides different methods in PGH such as: compute the measure metric by Bhattacharya metric, Chi-squared metric or Intersection metric; change the accuracy of PGH.

### **4.1.3 PHT module**

PHT module finishes the works in estimation stage. The main methods of PHT module stay in `PHoughTransform` class. It is a class extended from `HoughTransform` class, an abstract class. Besides the method extends from abstract class, `PHoughTransform` also implements the methods when we apply the probabilistic Hough transform as described in previous chapters.

### **4.1.4 Landmark detection module**

The landmark detection module used to implement the methods to refine the estimated landmarks from estimation stage. It includes an abstract class, `LandmarkMethod`, and an implement class which extends from abstract class, `LandmarkDetection` class. `LandmarkDetection` class

implements the methods to refine the estimated landmarks and compare the accuracy between the estimated landmarks and manual landmarks.

## 4.2 Software architecture and the packages

The MAELab software mainly includes two packages: **MAgIS** package and **Morphometry** package. The **MAgIS** package contains the methods to segment images and construct the interface of program. This package was finished by NGUYEN Hoang Thao. The **Morphometry** package contains the implementations to estimate landmarks automatically. Besides, we also use the method provides by OpenCV library(OpenCV library) and Qt framework (Qt Framework package) (see in figure 4.2).

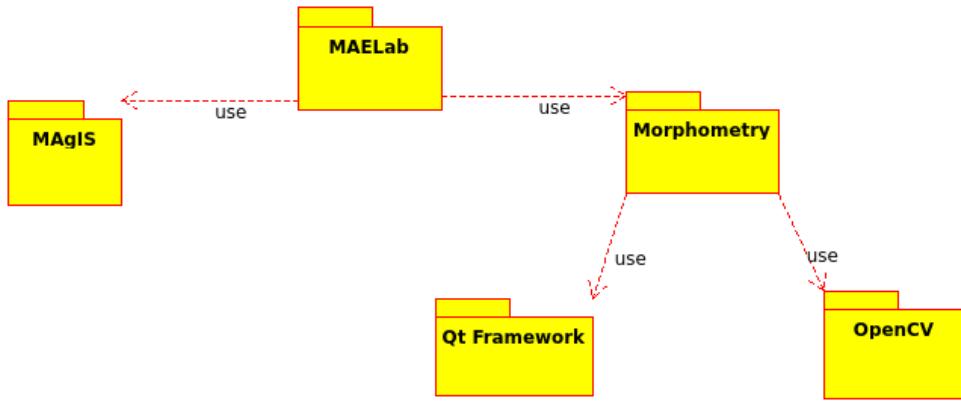


Figure 4.2: The packages of program

The figure 4.3 shows class diagram of this software. The **main** methods located in the **ImageViewer** class, where contains all functions of the software and connects with the **MAgIS** package. The connecting between the software and the modules made via **Scenario** class. To represent the information of image, we use the classes: **Line**, **Edge**, **Image**. For each main modules in program, we have the abstract classes, implementation classes and the support classes. They are described in the below subsections.

### Environment classes

The *environment classes* 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 describes 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

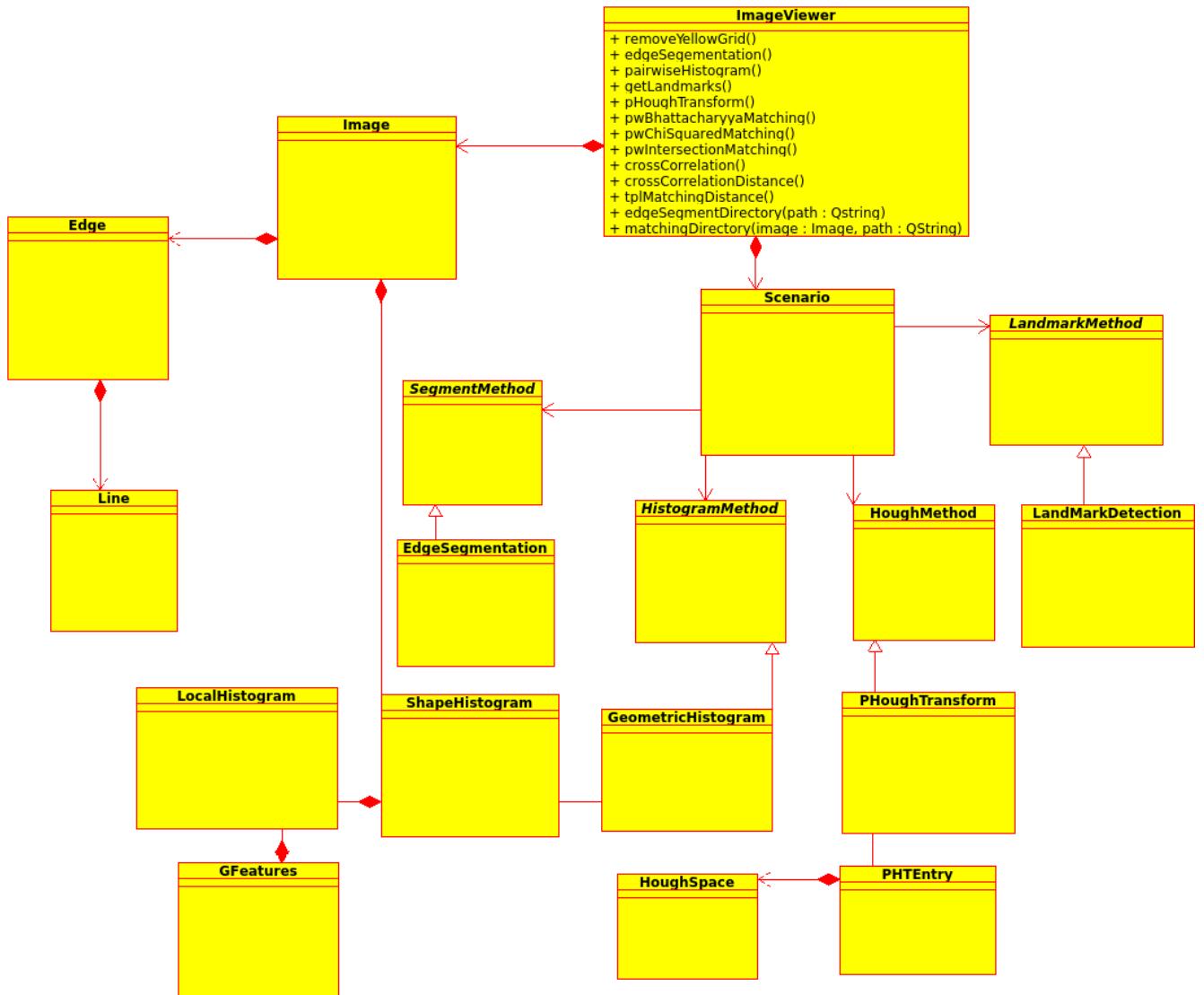


Figure 4.3: The class diagram of program

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, get the PGH of image, read its landmarks from a file, etc.

## The abstract classes

The abstract classes contains the abstract methods correspondence with the main modules of program. They provide the abstract methods which was implemented in the inherit classes. As introduce in previous section, the abstract classes include: `HistogramMethod` class, `SegmentMethod` class, `HoughMethod` class, `LandmarkMethod` class.

### 4.2.1 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.

### 4.2.2 Pairwise Geometric Histogram

This section describes the classes used to construct the PGH and compare the measurement of them.

`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.

#### 4.2.3 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.

#### 4.2.4 Refine the landmarks

`LandmarkDetection` class provides the methods to refine the estimated landmarks by probabilistic Hough transform. It also implements the cross-correlation technique to match the model and the image and provides the methods to compare the estimated result with the original result (by comparing distance between the landmarks).

## 4.3 Result

As presentation about our result, in this section we introduce about MAELab software, the parameters used in execution and the discussion about the result.

### 4.3.1 New interface

The results of this internship are integrated into IPM<sup>1</sup> software, but the software interface was reorganized (figure 4.4). Besides the functions of previous version, the clients can be use new functions to segment image or automatic extraction landmarks from image.

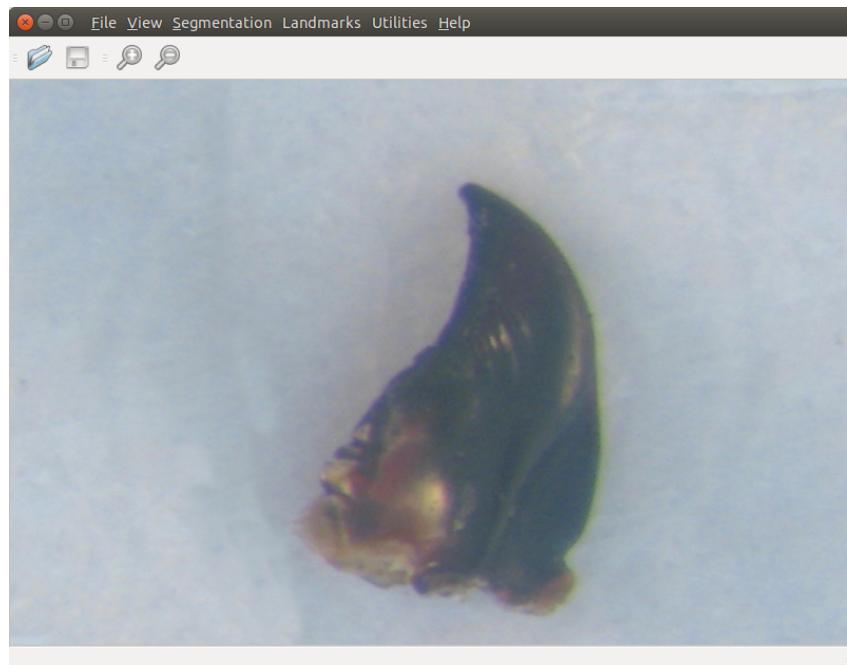


Figure 4.4: The graphic user interface of MAELab software

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 automatically estimation landmarks and analysis the result from estimated landmarks. The process can be finished on an image or a list of images.

Besides, the other functions developed in MAELab software are:

- Image segmentation by analysing histogram (in **Segmentation** menu)
- Removing the yellow grid on image (in **Utilities** menu)
- Loading the manual landmarks (in **Utilities** menu)

---

<sup>1</sup>Image Processing for Morphometrics

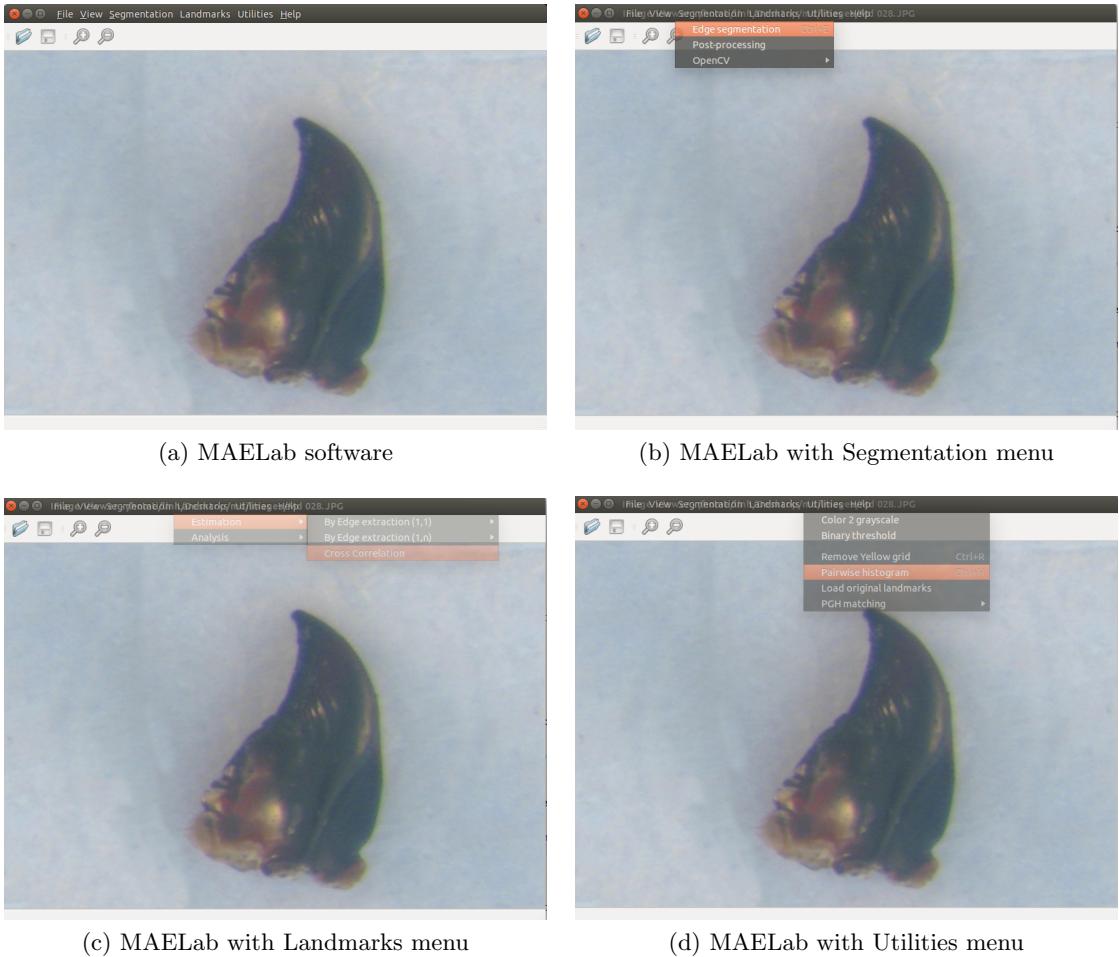


Figure 4.5: The MAELab software with its menu

- 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)

### 4.3.2 Experimentation

The experimentation is deployed on two machines with difference equipment:

- Machine 1: Intel(R) Core(TM) 2 Duo CPU T8100 2.1GHz, 2 GB of RAM
- Machine 2: 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
- Mg 007.JPG
- Mg 248.JPG
- Md 146.JPG
- Mg 040.JPG
- Mg 292.JPG
- Md 238.JPG
- Mg 066.JPG
- Mg 003.JPG
- Mg 159.JPG

Because the estimation method has several steps, we decide focus on some steps of method, such as segmentation and estimation. For each step, we compute the execution time on one image and a list of images. Then we compare the runtime between two examination systems (see in table 4.1 and table 4.2).

Machine 1:

No of images	Segmentation(second)	Estimation(second)
1	0.844	31.4245
290	571.576	13000.9131

Table 4.1: The runtime of program on machine 1

Machine 2: Based on the runtime tables, the most of runtime stays in stage of estimation.

No of images	Segmentation(second)	Estimation(second)
1	0.27782	10.4392
286	171.589	4665.79

Table 4.2: The runtime of program on machine 2

This is the time to execute two steps: estimated landmarks (by PHT) and refine landmarks (by template matching). Hence, it also depend on the configuration of system.

### 4.3.3 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  $1 * \text{threshold}$  value and the upper value is  $3 * \text{threshold}$  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.

#### 4.3.4 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.

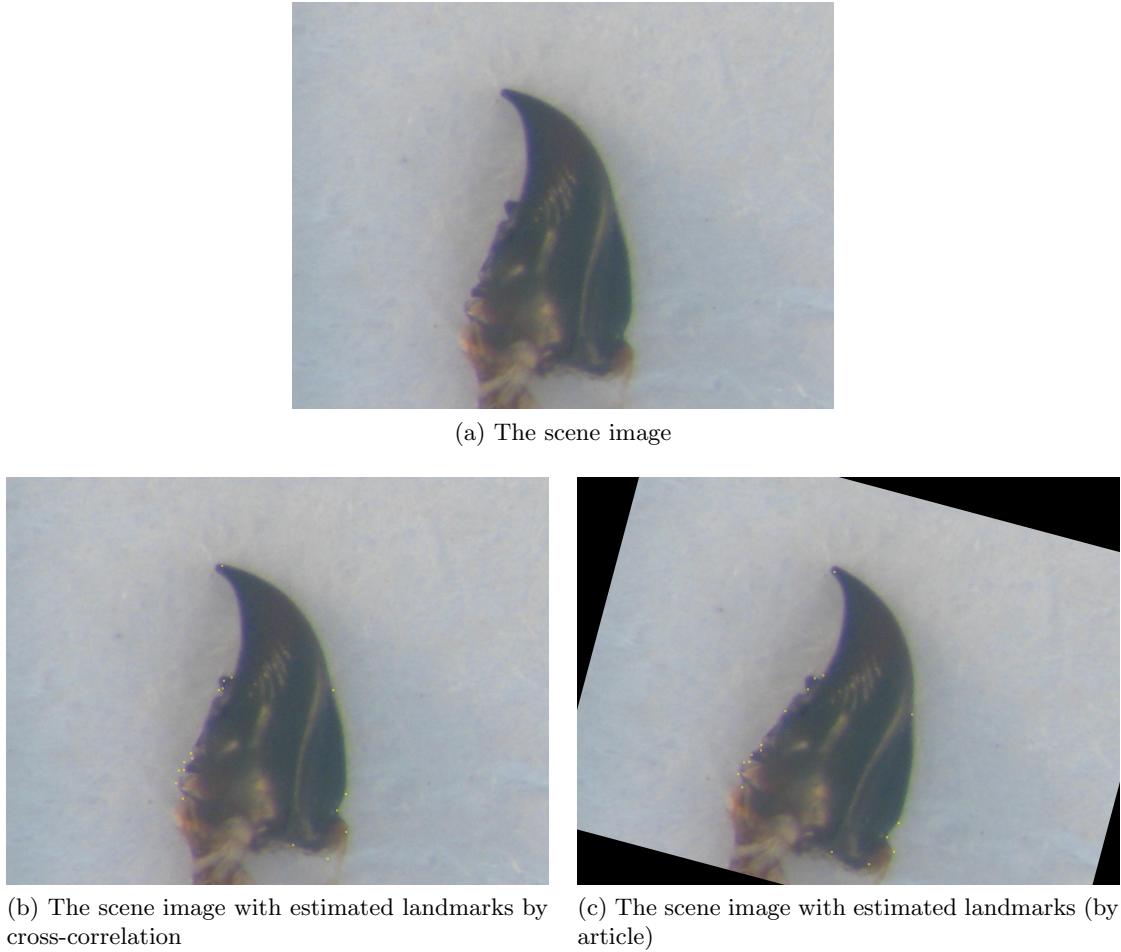


Figure 4.6: Automatic identification of the landmarks

To have the basic evaluation about this method, we compare its result with the result from “**original**” of cross-correlation method. Obvious, the location of landmarks obtained from method (follows article) more closer than the landmarks obtained from cross-correlation (see in figure 4.6).

Besides, 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.

#### **4.3.5 Limits and future works**

The method includes several steps to get the last result. At each step, we can use different ways to do. The methods proposed in this report just a part of them. Moreover, the program has used the parameters as the conditions, changing the values of parameters can effect to the accuracy of program. Currently, the work finishes on two sets of data: *right mandible* and *left mandible*. In the future, we can evaluate and improve this method. Then applying this method on other dataset of biological images.

# Conclusion

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*. But we can try to implement by using other methods. This work is needed to compare and choose a best method to estimated the landmarks automatically.

# 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] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.
- [3] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.
- [4] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.
- [5] 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.
- [6] Jos Luis Espinosa Aranda Jesus Salido Tercero Ismael Serrano Gracia Noelia Vlez Enano Gloria Bueno Garca, Oscar Deniz Suarez. *Learning Image Processing with OpenCV*. Packt Publishing, 2015.
- [7] Paul VC Hough. Method and means for recognizing complex patterns. Technical report, 1962.
- [8] David G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395, 1987.
- [9] Sasirekha Palaniswamy, Neil A Thacker, and Christian Peter Klingenberg. Automatic identification of landmarks in digital images. *IET Computer Vision*, 4(4):247–260, 2010.
- [10] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

- [11] 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.

# Annex

## 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

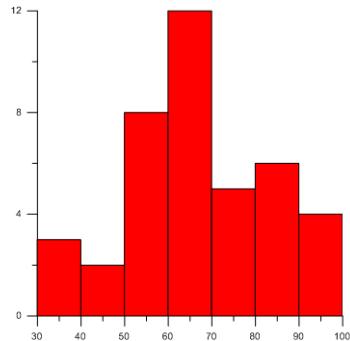


Figure 7: An example about histogram

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.

## 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 <sup>2</sup>:

- Apply the Gaussian filter to smooth the image (remove the noise),

---

<sup>2</sup>[https://en.wikipedia.org/wiki/Canny\\_edge\\_detector](https://en.wikipedia.org/wiki/Canny_edge_detector)

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

## Color processing<sup>[6]</sup>

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 affected 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 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.