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 3, 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

Morphometry is a concept refers to qualitative analysis of form, it includes the size and the shape of an object. Mophormetry analyses are commonly performed on organism. An objective of morphometry is to statistic hypotheses based on the effect of shape. In image processing, morphometry 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 morphometry forms: traditional morphometry, landmark-based geometric morphometry, outline-based morphometry.

In this internship, we consider landmark-base geometric morphometry. 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.

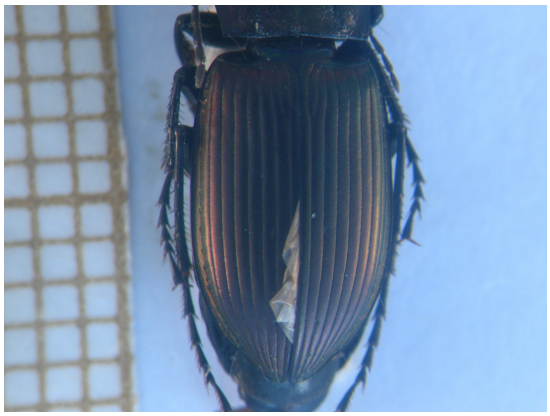
1.3 The Internship

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.

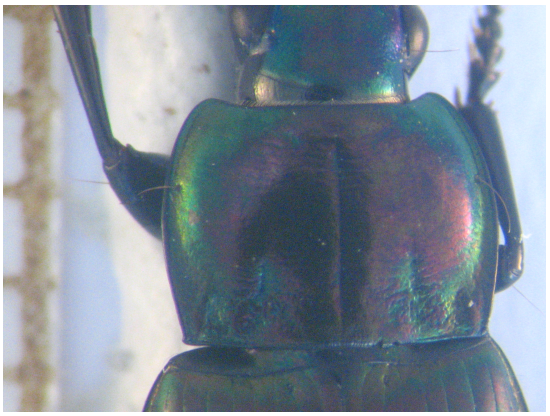
As a part of collaboration between LaBRI and INRA Rennes, this project working on the agriculture field. The goals of project is tracking, collecting and classifying the insects based on the morphometry of them, specific beetle. The works are finished in several ways on the different parts of beetle such as head, body, right mandible, left mandible and pronotum.

As mentioned in previous part, in this internship, we consider landmark-base geometric morphometry to estimate the landmarks on beetle. **Morphometric landmarks are points that can be defined in all specimens and located precisely**[7]. 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.

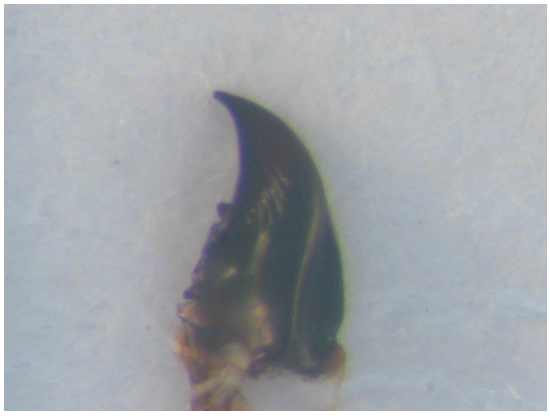
As part of goals, the objectives of this internship is implementing a method to automatic extract landmarks from biological images and witness on two sets of data: *right mandible* and *left mandible*. This method based on the article: **"Automatic identification of landmarks in digital images"**, by Palaniswamy[7]. Besides, we compare the estimated landmarks with the manual landmarks, which has been determined.



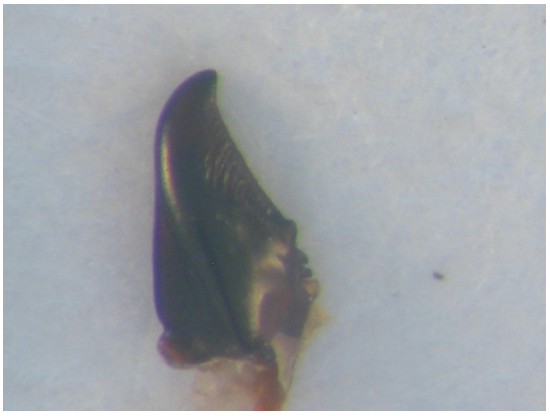
(a) Body part of beetle



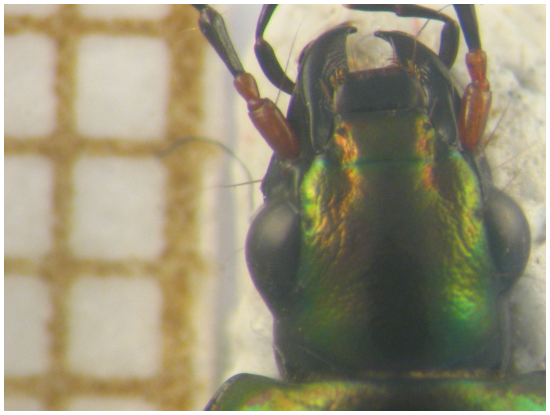
(b) Pronotum part of beetle



(c) Right mandible of beetle



(d) Left mandible of beetle



(e) Head of beetle

Figure 1.1: The parts of beetle

Chapter 2

Analysis

In this chapter, we describe a method to automatic extraction landmarks from biological images. 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.

2.1 Segmentation

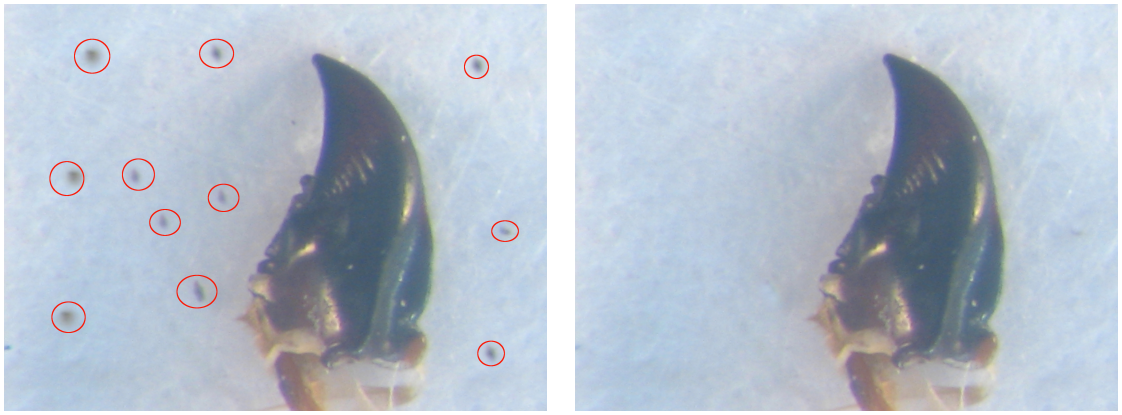
Segmentation is a process to extract interested features (lines) from the digital image. The expected result in this process is the list of the approximate lines which are used to construct the pairwise geometric histogram.

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 as well as enhance the quality of image. In second stage, we extract the features based on the edge segmentation. After that, by applying the appropriate technique to obtain the step edges and broken the edges into approximate lines.

2.1.1 Pre-process image

Pre-processing is a process that reduces the noises of image and enhance the quality of image's features. To do that, the simplest method in segmentation was used, **threshold**. When we apply **threshold** method, the *threshold value* is important in this technique. The threshold

value might indicated follows two ways: manually or automatically. In the concept of this report, we will introduce an method to indicate the threshold value by analysing the image histogram. This method will be focused in next chapter.



(a) Image with noise (b) Image without noises

Figure 2.1: Example about noises in image

2.1.2 Feature extraction

After reducing the noise from input image in pre-processing stage. In this stage, by applying suitable technique, we can extract the interested in features (edges) from the image. The Canny[3] algorithm is used to detect the step edges, which incorporates non-maximal suppression and hysteresis thresholding. In Canny algorithm, the important parameters are the two threshold values (*lower threshold* and *upper threshold*) and the aperture size of the Sobel operator, it decides the pixels kept. Normally, the value of lower threshold and upper threshold follows a certain rate, and some inputs that we need to know when applying the Canny algorithm as follows:

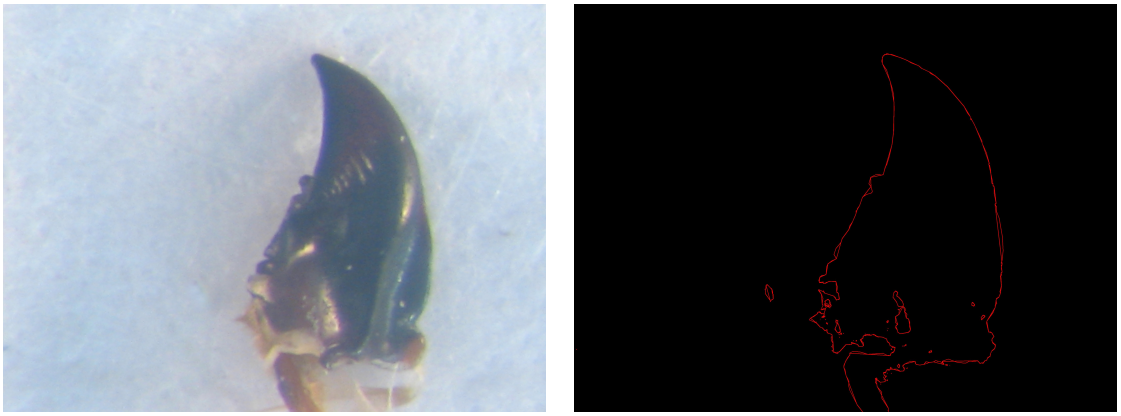
- *Source*: the input image (in grayscale mode)
- *Destination*: the output image,
- *Lower threshold*: the first (lower) threshold value,
- *Upper threshold*: the second (upper) threshold value,
- *Kernel size*: size of kernel, aperture for the Sobel operator.

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 which was proposed by Suzuki[8].

2.1.3 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 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[9]. The last result of this step is a set of approximate lines with the edges of image. These will used to describe the shape in the compact invariant. An example about edge segmentation is presented in figure 2.2.



(a) Original image (b) Image with line segmentation

Figure 2.2: An example about image segmentation

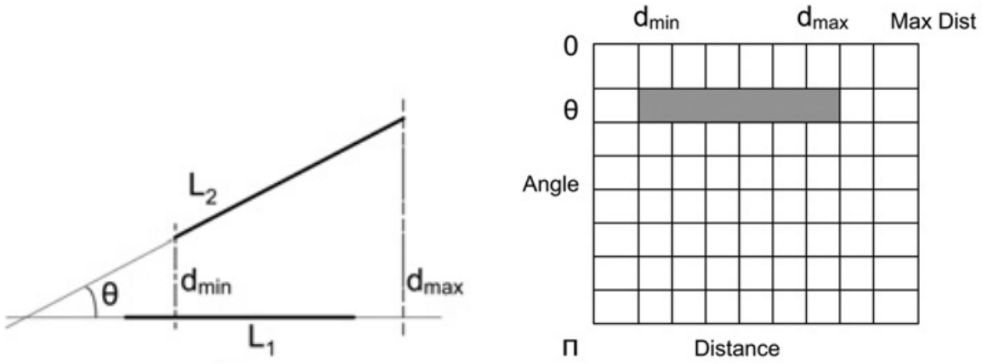
2.2 Pairwise Geometric Histogram

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

As example in figure 2.3, the figure 2.3a represents the geometric relationship between two



(a) The geometric relationship between two lines

(b) The pairwise geometric histogram

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

lines and 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 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 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. 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 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.

2.2.3 Histogram matching

"The histogram matching enables robust classification of shape features by finding similarity between the scene and reference model"[7]. 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 image is represented by a global pairwise histograms and uses the Bhattacharya[9] metric to determine the similar distance between the two models [7]. In general, we have normalize the histograms before comparing. The form of Bhattacharrya metric used to compute the degree of 2 models is:

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

The significance of parameters in the formula 2.1, 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

2.3 Probabilistic Hough Transform

Hough transform is a technique used to extract the features in images. The main idea of this method based on the vote procedure to find the best similar of object in a certain class of objects. The Hough transform was originally developed to recognize the line[6] 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. And the method used to solve also based on the vote procedure. 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 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, then finding the similar pairs between model image and scene image. Secondly, estimating process starts as predicting the pose of scene different from the model, then we estimate the location of the landmarks.

2.3.1 Training process

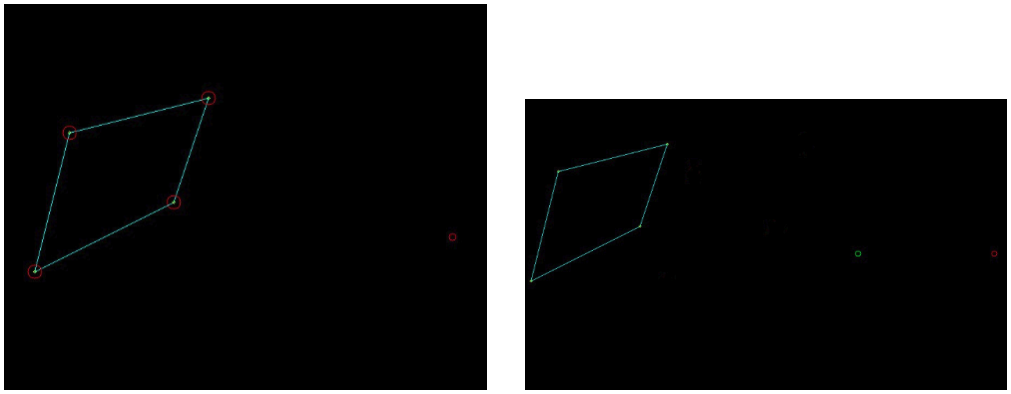
Training process is process recording the relationship of model to a reference point in model. At this step, the perpendicular distance and angle from each pair model lines to a reference point was recording and saving (called reference table). The reference point can be chosen at arbitrary position on the model image. The expected result in this process is similar pairs between model and scene. At this step, PHT used to do that. The chosen pair of scene lines obtained from best votes when we consider the similarity between each pair of scene and model.

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 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. With a pair of scene lines agree with a pair of model lines was chosen at previous step, the model reference point in the scene image can be detected by extending the perpendicular lines of the pair of scene lines at the appropriate position.

The estimated landmarks of model 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.

In figure 4.1, 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



(a) The model image (b) The scene image

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. 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 2.4):

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

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:

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

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

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

Chapter 3

Conception

Previous chapter has discussed about the method which we used to estimate the landmarks on biological image. This chapter describes the conception of that method.

3.1 Segmentation

3.1.1 Pre-process image

As discuss in previous, we will 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 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 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 calculating the mean of it. The threshold value **"t"** obtained by the mean of the two mean values in two parts of histogram (as algorithm 1).
With the threshold value **"t"**, we apply the threshold method (in OpenCV[1]) to pre-process image in the CV_THRESH_BINARY mode (keep the pixel has value greater than threshold value).

[1]http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

Algorithm 1: Algorithm to get the threshold value and pre-process 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;
$limit \leftarrow (mean > median \ ? \ median : mean)$;
$limitSub \leftarrow ((limit >= 120) \ ? \ (limit - 25) : (limit - 5))$;
Declare some variables: $int \ imax \leftarrow -1, max \leftarrow -1$;
for $i \leftarrow 0 \ to \ limitSub$ **do**
    if $histogram[i] > max$ **then**
        $max = histogram[i]$;
        $imax = i$;
    end
end
Declare some variables: $int \ imin \leftarrow -1, min \leftarrow max$;
for $k \leftarrow imax \ to \ limit$ **do**
    if $histogram[k] < min$ **then**
        $min = histogram[k]$;
        $imin = k$;
    end
end
Declare some variables: $int \ max2 \leftarrow -1, imax2 \leftarrow -1$;
for $j \leftarrow limit \ to \ end\_of\_histogram$ **do**
    if $histogram[j] > max2$ **then**
        $max2 = histogram[j]$;
        $imax2 = j$;
    end
end
$middle1 \leftarrow (imax1 + imin)/2$;
$middle2 \leftarrow (imax2 + imin)/2$;
$middle \leftarrow (middle1 + middle2)/2$;
Apply the threshold with threshold value is *middle*;

3.1.2 Features 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 also the value used in the previous step, and the ratio between lower threshold and upper threshold is 1 : 3 (follows the article [7] but have to be modified). This operation can be used from OpenCV library. The parameters need to be

provided into Canny[2] operator 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 **findContours**[3] method in OpenCV. 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 **findContours** operation 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 segmentation

The method to segment the edges is the recursive algorithm[9] but it has some changes in the "stop condition" of the algorithm to simplify. The progresses of this algorithm are 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 ($\alpha = 3$).

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

- 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 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 ← listPoints[0];
2  Declare the second endpoint: pend ← 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 ← 0;
5  Declare a "split point": imax ← 0 ;
6  Declare a variable: distance ← 0;
7  for point p in listPoints do
8  |  distance ← from p to line d;
9  |  if distance > max_distance then
10 |  |  maxDistance ← distance;
11 |  |  imax ← 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 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 - π) and 500, respectively.

3.2.1 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 Histogram matching

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:

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

Intersection metric

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

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 Probabilistic Hough Transform

The core of PHT uses vote procedure to find the model's position in the image. This process includes two stages: firstly, we record the model information by calculating the angle and perpendicular distance of each pair of model lines to an arbitrary point (called **reference point**) in model. Secondly, we estimate the model's position in image by voting procedure. 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 (the best vote) is assumed as the reference point of the model in the scene image. From this reference point, we can find the position of model in image.

Following that, 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,
- 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.

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 3: Algorithm to consider the closest lines

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

```
1  distance1 ← distance from the first endpoint of line1 to line2;
2  distance2 ← 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 4: 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 != 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 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 (bool value)

```
1  angle1 ← angle between line1 and line2;
2  angle2 ← angle between sline1 and sline2;
3  mdistance ← sum of perpendicular distance from two endpoints of line1 to line2;
4  sdistance ← sum of perpendicular distance from two endpoints of sline1 to sline2;
5  if abs(ange1 - 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 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

```
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 7: 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 8: 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 Template matching

Refining landmarks is finished by apply the cross-correlation. 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 9: 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 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 $matchTemplate$[4] 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, $minMaxLoc$[5]. 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.

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

Algorithm 10: 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 a short description about model as well as the modules of software. Then, we give the software architecture and explain 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 can be used in all modules of software.

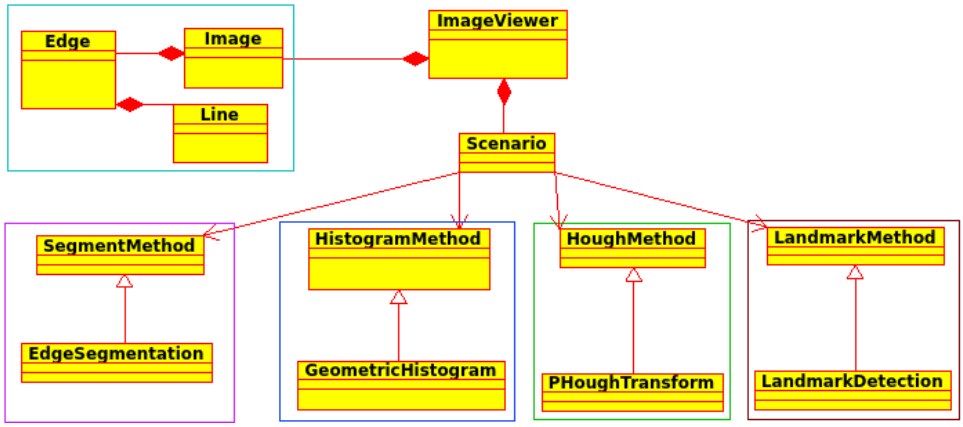


Figure 4.1: The modules of program with main classes

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 pairwise geometric histogram 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. 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. Besides the method extends from abstract class, `PHoughTransform` also implements the methods when we apply the probabilistic Hough transform as described above.

4.1.4 Landmark detection module

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

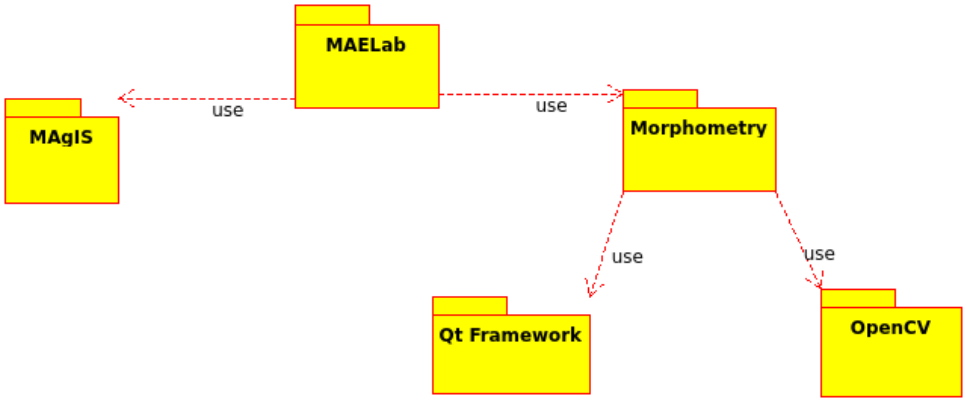


Figure 4.2: The packages of program

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 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

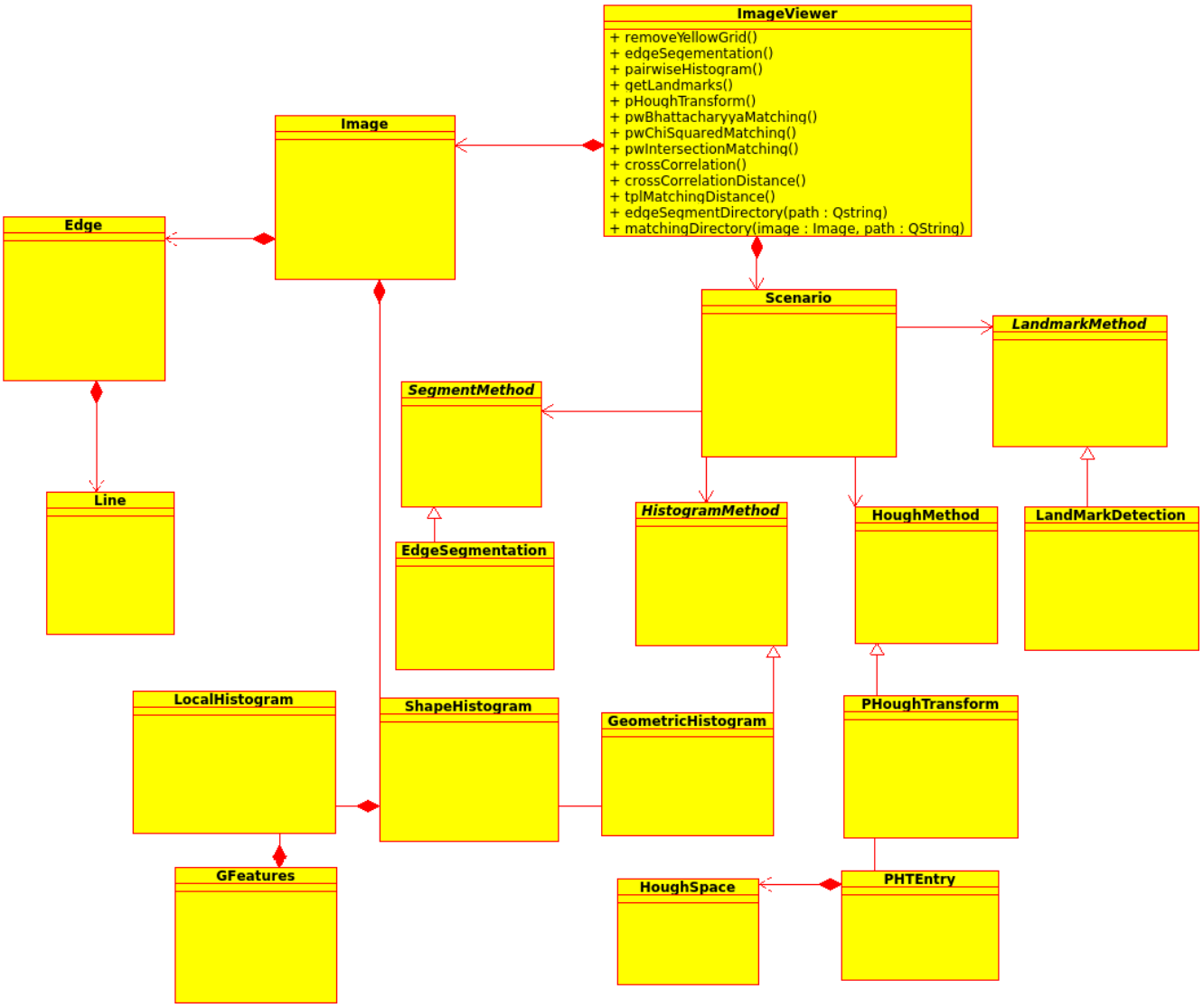


Figure 4.3: The class diagram of program

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 chapter 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[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.

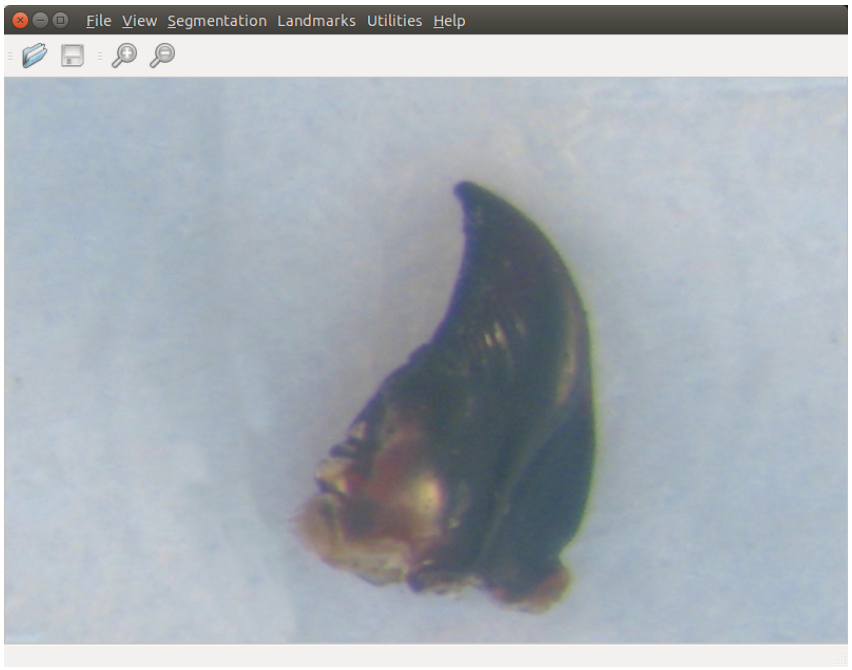


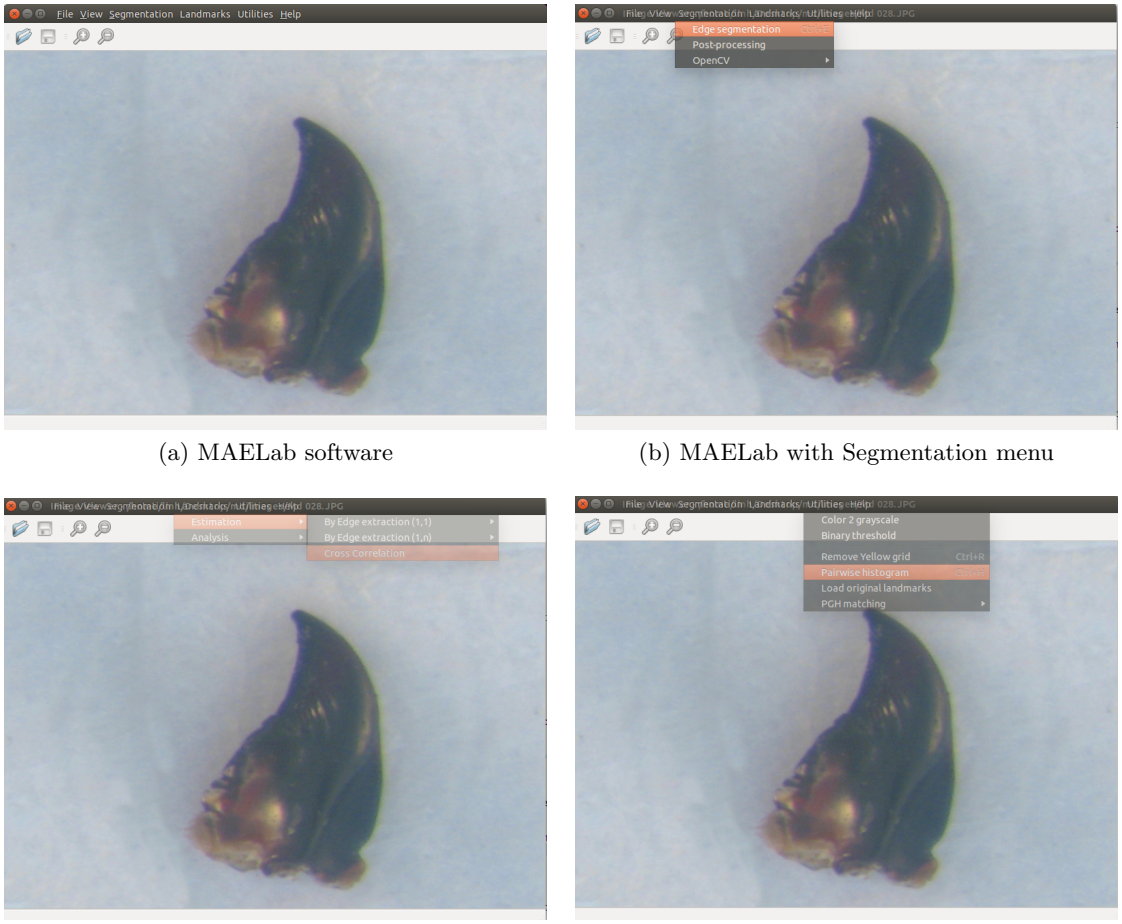
Figure 4.4: The graphic user interface of MAELab software

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, 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)

[1]Image Processing for Morphometrics



(a) MAELab software

(b) MAELab with Segmentation menu

(c) MAELab with Landmarks menu

(d) MAELab with Utilities menu

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
- 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

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
291	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
291	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 * *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**

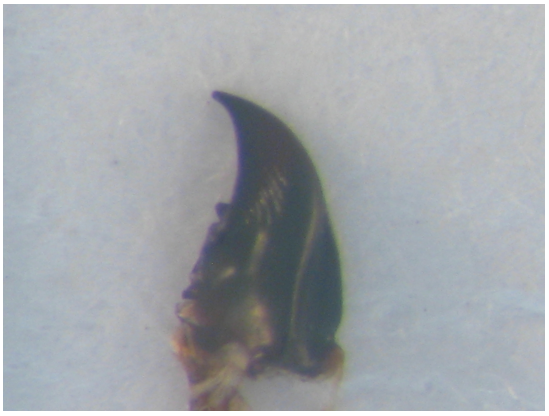
- The size of bounding box around the reference landmarks used for estimating landmarks by cross-correlation method or computes the estimated centroid is *400* pixels (used in method `crossCorrelation and crossCorrelationDistance`, class `ImageViewer`)

- 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.

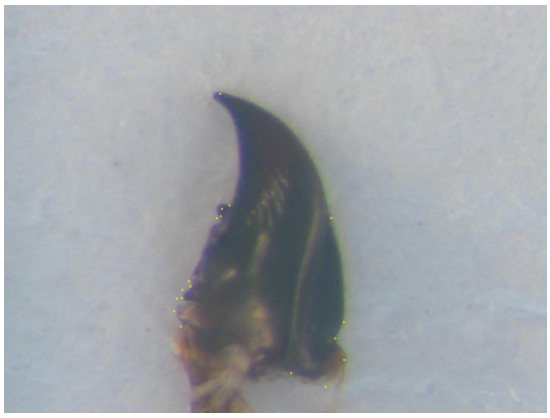
4.3.4 Results

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

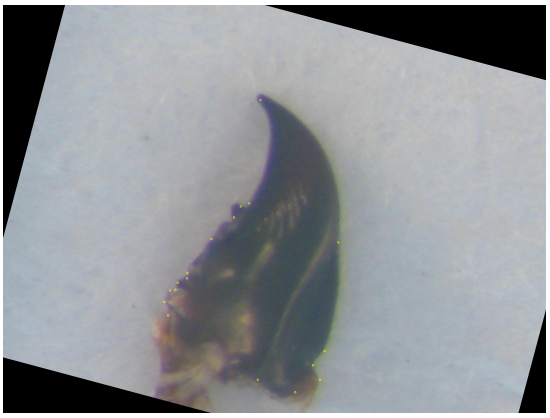
landmarks for each *left mandible* image.



(a) The scene image



(b) The scene image with estimated landmarks by cross-correlation



(c) The scene image with estimated landmarks (by article)

Figure 4.6: Automatic identification of the landmarks

To have the basic evaluation about this method, we compare its result with the result from cross-correlation. Obvious, the location of landmarks obtained from method (follows article) more closer than the landmarks obtained from cross-correlation.

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 use the parameters as the conditions, changing the values of parameters can effected 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] Paul VC Hough. Method and means for recognizing complex patterns. Technical report, 1962.

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

[8] 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.

[9] 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.