



UNIVERSITY OF BORDEAUX

INTERNSHIP REPORT

MASTER OF SOFTWARE ENGINEERING (2013 - 2015)

Design and programming of automatic classification methods applied to biological images

Student:
LE Van Linh

Supervisor:
Prof. Marie BEURTON-AIMAR

September 21, 2015

Acknowledgements

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

I would like to thank 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 for technique and gave me a professional working environment.

I would also like to thank all professors in University of Bordeaux and PUF-HCM, who imparted a lot of knowledge to learning and researching. I would also like to thank the dean of IT-DLU, who allowed me joined in this course. Finally, I would like to thank my family and colleague for their support and encouragement through my study.

Abstract

Image processing is a field that has many application in life. It can be from the usual application to the application in medicine or cosmology. To obtain the best result, all most of applications must follow two processes: Firstly, we should pre-process the image with some appropriate operations to enhance the interest in also reduce the noises. Secondly, we apply main operations to obtain the result.

The goal of project is built a program with full functions about processing base on the biological images. During my internship at LaBRI, my tasks are developing the algorithm to preprocessing image by removing the unexpected parts. Besides, we also program a method to automatic classification on biological images. The method based on the segmentation and classification.

Finally, I integrated my functions into the IMP tool, which was developed by NGUYEN Hoang Thao. Besides, we also debug the previous code and write the documentation for the next development.

Contents

1	Introduction	5
1.1	Pôle Universitaire Français	5
1.1.1	PUF-Ha Noi	5
1.1.2	PUF-HCM	5
1.2	Laboratoire Bordelais de Recherche en Informatique	6
1.3	The Internship	6
1.3.1	Objectives and my task	6
1.3.2	Organization of the document	7
2	Background	8
2.1	Overview about image processing	8
2.2	Smoothing filters	8
2.3	Histogram	8
2.4	Segmentation	9
2.5	Color processing	10
3	Preprocessing image	12
3.1	Problem	12
3.2	Analysis	12
3.2.1	Finding the limiting and the replacing point	13
3.2.2	Replacing the grid	14
3.3	Summary	16
4	Classification methods	18
4.1	Preprocessing image and feature extraction	18
4.1.1	Preprocess image	18
4.1.2	Feature extraction	19
4.1.3	Edge segmentation	20
4.2	Pairwise geometric histogram	21
4.2.1	Pairwise geometric histogram	21
4.2.2	Histogram matching	22
4.2.3	Statistical the matching result	23
5	Implementation	24
5.1	Software architecture	24
5.2	Image preprocessing	25
5.2.1	Line class	25
5.2.2	Edge class	26
5.2.3	Image class	26

5.3	Automatic classification	26
5.4	Result	26
6	Conclusion	27

List of Figures

2.1	An example about histogram	9
2.2	The images with color transformation from BGR to Gray	11
3.1	The input images with yellow grid	12
4.1	The geometric features and the PGH	22
5.1	The class diagram of program	24

Chapter 1

Introduction

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 Nationale du Vietnam - Ha Noi located in Ha Noi capital (PUF-Ha Noi) and Pôle Universitaire Français de l'Université Nationale 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-HCM¹ 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.

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.

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<http://pufhcm.edu.vn>

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

1.3 The Internship

The internship is considered a duration to apply the knowledge to the real environment. It shows the ability synthesis, evaluation and self-research of student. Besides, the student can be study the experience from the real working. My internship is done under the guidance of Prof. Marie BEURTON-AIMAR in a period of six months at LaBRI laboratory.

1.3.1 Objectives and my task

In any fields, constructing and developing a tool to support fully operations need a period of time. With the expect, creating a tool to support the operations about image processing, IMP was created. Begin in 2012, IMP was created by NGUYEN Hoang Thao. In the first version, IMP had basic operations about image processing such as segmentation, smoothly, morphology, transform,.... Besides, it also integrated some algorithm which was processing on image.

As a part of IMP, the general objectives of this internship is developing the operations in the IMP tool, as follows.

- Design and implementation the method to remove the grid on biological images
- Design and program the method to automatic classification on biological images
- Maintain some operations in IMP

²<http://www.labri.fr>

1.3.2 Organization of the document

The all report mainly have five chapters. In the chapter 1, this is the short introduction about my university, mainly information about the lab, where I do the internship and the objectives of my internship. In chapter 2, we talk about the necessary preliminaries in image processing field which we use to implement the methods. In the chapter 3, I propose the algorithm to preprocessing image, with the aim is decrease the noise in the input and increase the effective of the classification methods. In the chapter 4, I mention method to segmentation image and classification process. Finally, I present about the implementation of the preprocessing image algorithm and classification methods.

Chapter 2

Background

2.1 Overview about image processing

We have a lot of programs what used to edit the photos (e.g. photoshop, gimp, paint,...). By apply some technique, we can effectively some property to change the image such as: scaling image, blurring image, rotating image,.... We also know that, an image is a set of pixels. Each pixel have a value that present for the color at this location, and its location was indicated by coordinates in two-dimension. When combine the value of all pixels, we have the image as we can see in the real word. The changing on image really changing the value on each pixel in image. Behind the techniques in the programs are mathematical operations and the field using mathematical operation on an input image, called *image processing*. The output of image processing may be either an image or a set of characteristics related to the image. And most of image processing technique are performed on two-dimensional image.

2.2 Smoothing filters

Smoothing filters are used for blurring and noise reduction. This technique is used in preprocessing steps, such as remove some small object unexpected from input image, or bridging of small gaps in lines. Noise reduction can be done by blurring with a linear filter or order-statistics filter.

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

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

2.3 Histogram

Histogram is a representation about distribution of data on the regions (we called bin) in the data range. The bins are the number of sub-range when we divide the entire data range into several small interval (i.e. With the range from 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 numbers of data which have value belong to this bin. Normally, histogram represented by the columns chart with x-axis represented for the number of bins, and y-axis represented for the value of each bin.

Histogram can be used effectively for image enhancement, also useful in many image processing applications, such as image compression and segmentation.

Histogram equation: is a method allow adjust the contrast using the histogram of image. It

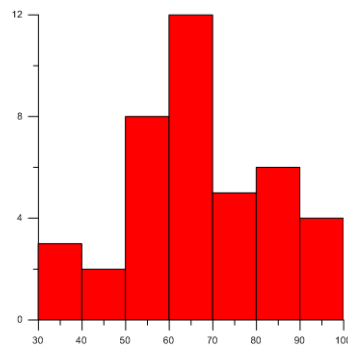


Figure 2.1: An example about histogram

mapping one distribution on a histogram to a wider distribution of intensity values. By this, the image can brighter.

Histogram matching: is a method adjustment of two image using the histogram. This method was finished by calculating the cumulative distribution functions of two histograms and find the histogram matching function. Finally, apply the matching function on each pixel of the image to get the result.

2.4 Segmentation

Segmentation subdivides an image into its regions. The size of regions is depend on the problem being solved. This mean, segmentation should stop when the regions of interest in application have been detected. In the real, the segmentation was applied into many fields such as machine vision, medical imaging, object detection, etc. The most 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. The second case, the image segmentation based on a predefined criteria. It means the image was segmented into regions that are similar according to a set of criteria. And, we have many the method to segment an image such as thresholding method, region growing, clustering method, histogram-based method, etc.

Thresholding is a simplest method of image segmentation. Thresholding use a particular threshold value “ t ”, we split the image into two parts: the first part includes pixels which have the value greater than t , and the second part contains the pixels vice versa. With this technique, thresholding can be used to create an binary image from a gray scale image. In fact, we have many type of threshold, 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 thresholding 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 image. The process of this algorithm can break into the steps follows ¹:

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

2.5 Color processing

The use of color in image processing do not just identify or extract an objects from scene, it also a factor for image analysis. Color processing can be effect on each component image individually or work 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 consist 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 integrate between 2 primary color in BGR. Specific, C (cyan) is consist from green and blue, M (magenta) is consist from red and blue and Y (yellow) is consist from red and green.
- HSV model: difference with BGR, HSV using the 3 components are hue, saturation and brightness to present image. Hue is a color attribute which describe a pure color (yellow, orange, red) and saturation give a degree to pick the pure color is diluted by white light. Brightness is a notation of intensity for color sensation.
- Grayscale model: The colors in grayscale just 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 color operations in image processing is transformation. This is a process to the conversion image between the color models by using a transform expression such as BGR to HSV, HSV to BGR, BGR to Grayscale.

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



(a) An image in BGR mode



(b) An image in Gray mode

Figure 2.2: The images with color transformation from BGR to Gray

Chapter 3

Preprocessing image

3.1 Problem

Preprocessing data is a stage that we remove the noises and unexpected objects on image. To obtain the good result when applying the automatic classification methods. In this chapter, we suggest a method preprocessing image.

With the input is a set of 293 images. Each image contains the parts of insect and an unexpected object, specifically yellow grid (figure 4.1). The method proposed to remove the yellow grid and keep the insect on the image.

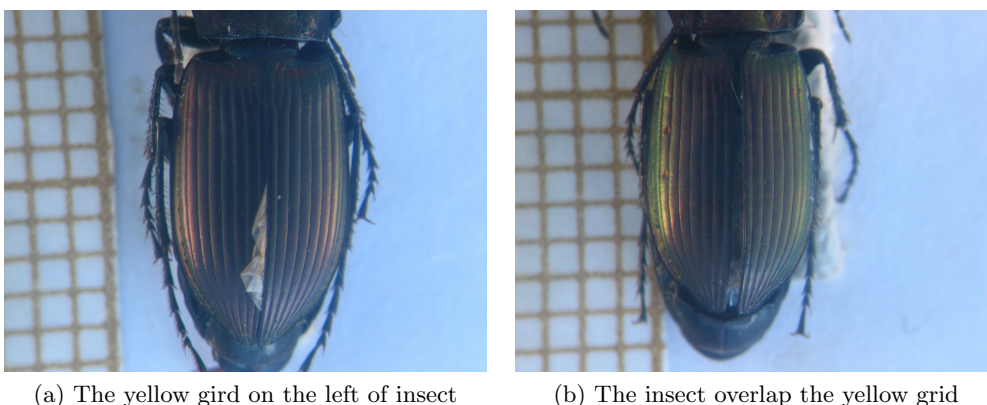


Figure 3.1: The input images with yellow grid

3.2 Analysis

Each input image contains the two objects: a part of insect (called insect) and the yellow grid (called grid). About the relative position, the grid always stayed in the left of insect, and insect can either overlap the grid. About the color, image is presented in BGR model with three main color groups: the background color, the yellow color of grid and the color of insect.

The method proposed to remove the grid based on the color processing. If we process the image in BGR model, the algorithm may be complex because the color at each pixel is combined of three values (blue, green, red). While HSV model has a specific channel to present the colors with clear range. We can apply this property for detecting and removing the grid. The proposed process to remove the grid as follows:

1. Find the “*limit*” point of grid: the points stay nearest outside grid.
2. Find the “*replace*” point: location that its value used to replace for grid.
3. Replace the grid by the value at “*replace*” point.

3.2.1 Finding the limiting and the replacing point

Browsing image to check and replace the pixels in grid need a long time. To reduce the time to do that, we should find the limit range of grid. The limit of grid is the points located out of grid and its closest. Instance of checking on all pixel, we just check the pixels stay on the left of limit points.

As we know, the width of grid usually less than a two-thirds of width of image. So, to reduce the time to finding the limit point, we also check from the begin of image to two-thirds of image. The result of this step is the limit points, these used for limiting the length when we check the pixels on yellow grid.

The algorithm to find the limit points are followed:

Data: *inputImage*: The input image (contains the insect and grid)

Result: The coordinate of limit point

```
1 Declare some variables: Mat hsvImage; vector < Mat > hsv_channel;  
2 Convert image from BGR to HSV:  
   cv : cvtColor(inputImage, hsvImage, COLOR_BGR2HSV) ;  
3 Split HSV image into several channel: cv :: split(hsvImage, hsv_channels);  
4 Set up initial limit_point and assign with the left-top corner: Point  
   limit_point = Point(0,0);  
5 Declare a variable yellow_count to count the number yellow points on each columns  
   when processing. An column become a limit line if the number of yellow points on this  
   column less than a constant value.;  
6 for j ← 10 to hsv_channel[0].cols do  
7   if H value at (5,j) > 100 || (H value at (5,j) > 70 && H value at (5,j) < 100  
8   && S value at (5,j) < 10 && V value at (5,j) > 175 ) then  
9     limit_point.x ← j;  
10    limit_point.y ← 0;  
11    yellow_count ← 0;  
12    for i ← 1 to hsv_channel[0].rows * 2/3 do  
13      if H value at (i,j) <= 38 then  
14        yellow_count ++;  
15        if yellow_count >= 8 then  
16          limit_point.x ← 0;  
17          limit_point.y ← 0;  
18          break;  
19        end  
20      end  
21    end  
22    if limit_point.x != 0 then  
23      break;  
24    end  
25  end  
26 end  
27 if limit_point.x == 0 then  
28   limit_point.x ← hsv_channel[0].columns/3 + 200;  
29   limit_point.y ← 0;  
30 end
```

Algorithm 1: Algorithm to find the limiting points

Now, we indicate which is the color used to replace the yellow points. Hence, we choose the points having the value nearest with the background color. The histogram is ideal for choosing the position to replace, but we also have some conditions to obtain a good value.

The algorithm to find the replacing points are followed:

Data: inputImage: the input image

Result: The coordinate of replacing point

```
1 Convert image to gray scale image;
2 Calculate the histogram on gray scale image and mean of histogram;
3 Split the HSV image into channels;
4 for  $i \leftarrow 0$  to grayImage.rows do
5     for  $j \leftarrow 0$  to grayImage.columns do
6         if value at  $(i, j) > \text{mean of histogram}$ 
7             &&  $H \text{ value } (i, j) > 90$ 
8             &&  $H \text{ value } (i, j) > 130$ 
9             &&  $S \text{ value at } (i, j) > 50$ 
10            &&  $V \text{ value at } (i, j) > 215$  then
11                return this position ;
12            end
13        end
14 end
```

Algorithm 2: Algorithm to find the replacing point

3.2.2 Replacing the grid

After having the limit points. By processing on all rows of image. At each row, we replace the pixels which have the color value stay in the range of yellow by another value. But the grid is not only created by the yellow point, it contains more the pixel have the value stay in the same range with background. But the brightness of these pixels is less than the background. So, we needs to replace it obtained the good image. In each row, this work repeated until meeting the limit points or a “special point” (called “break” point). It can be a point stayed on the insect or a point belong to background.

For each part of the insect, the color on insect or the background also have the difference value. So, we establish the difference values for each part. Based on the file name of image, we can classify it.

Data: filePath: the file path of image

Result: Which part of insect in image

```
1 QString temp ← filePath.toLowerCase();
2 if temp contains “ely” then
3   | return ELYTRE;
4 end
5 if temp contains “md” then
6   | return MDROITE;
7 end
8 if temp contains “mg” then
9   | return MGAUCHE;
10 end
11 if temp contains “prono” then
12   | return PRONOTUM;
13 end
14 if temp contains “tete” then
15   | return TETE;
16 end
17 return ELYTRE;
```

Algorithm 3: Algorithm to get the parts of insect

Data: inputImage: the input image; limit_point: the limit point; part: part of insect;
minBrightness: minimum of brightness; rpoint: replacing point

Result: The image after replace the yellow grid

```

1 for  $i \leftarrow 0$  to inputImage.rows do
2   for  $j \leftarrow 0$  to limit_point.x do
3     if part is ELYTRE then
4       if value at  $(i, j + 50)$  satisfy breaking condition then
5         break;
6       end
7     end
8     if part is MDROITE or MGAUCHE then
9       if value at  $(i, j + 50)$  satisfy breaking condition then
10        break;
11      end
12    end
13    if part is PRONOTUM then
14      if value at  $(i, j + 50)$  satisfy breaking condition then
15        break;
16      end
17    end
18    if part is TETE then
19      if value at  $(i, j + 50)$  satisfy breaking condition then
20        break;
21      end
22    end
23    if  $H$  value at  $(i, j + 50)$  in yellow range then
24      replace value at this point by the value at replacing point;
25    end
26    else if  $V$  at  $(i, j + 50) > minBrightness$  then
27      replace value at this point by the value at replacing point;
28    end
29    ;
30  end
31 end
32 Merging three channel of HSV;
33 Convert the image from HSV to BGR;

```

Algorithm 4: Algorithm to replace the yellow grid

3.3 Summary

In this chapter, we propose a method to remove the grid in the image. In short, the algorithm have steps followed:¹

1. Converting the input image to HSV model
2. Splitting the image (in HSV) to get the individual channel
3. Finding the limit points

¹The algorithm is combined from the algorithms in each step, which was described above.

4. Choosing the replace point (calculating the histogram and mean value)
5. Getting the type of input and establish the break conditions.
6. Finding and replacing the yellow points and the “miss brightness” point.
7. Merging the channels of HSV
8. Converting the HSV image to BGR image

Chapter 4

Classification methods

In previous chapter, we introduce a method to remove the unexpected object. In this chapter, we will propose a method to obtain the features what we are interested in and the method to detect the landmarks on the insect. This method was proposed by Palaniswamy^[1]. The processes can be discuss in follow steps:

1. Extracting the features:
2. Constructing and comparing the pairwise geometric histogram
3. Detecting the landmarks

4.1 Preprocessing image and feature extraction

To obtain the good result, before extracting the features in the image, we need to pre-process the image with a appropriate technique to reduce the noise as well as enhance the features that we care. Feature extraction is a process extracting interested features from digital image. The expect result in this result is list of approximate lines which use to construct the pairwise geometric histogram.

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

4.1.1 Preprocess image

In this application, we use the thresholding technique to pre-process the image. In 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 median of this histogram. With the histogram obtained, we split it into two parts: the first part begin from the bin 0 to the limit value (the limit value is smallest value between mean and median); the second part, starting from the limit value to the end of histogram. For each part, we find the maximum, minimum value and calculating the mean of it. The value “t” obtained by the mean of two mean values in two parts of histogram.

With the threshold value “t”, we apply the threshold technique to pre-process image in the

CV_THRESH_BINARY mode (keep the pixel has value greater than threshold value).

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  $limitSub \leftarrow ((limit \geq 120) ? (limit - 25) : (limit - 5))$ ;
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      $max = histogram[i]$ ;
10     $imax = i$ ;
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      $min = histogram[k]$ ;
17      $imin = k$ ;
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      $max2 = histogram[j]$ ;
24      $imax2 = j$ ;
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;
```

Algorithm 5: Algorithm to preprocess image

4.1.2 Feature extraction

After apply the threshold to pre-process image, we apply the Canny algorithm to detect the step edges, which incorporates non-maximal suppression and hysteresis thresholding. In Canny, the importance parameters are two threshold values and aperture size for the Sobel operator, it decides the pixels kept. The threshold value used in Canny algorithm also the value used in the previous step, and the ratio between lower threshold and upper threshold is 1.5 : 3 (follows the article [1]). In implementation, the Canny operation used from OpenCV library¹, and the parameters need to put into Canny are:

- source: the input image (in grayscale mode)
- destination: the output image

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

- `low_thresh`: the first (lower) threshold value
- `high_thresh`: the second (upper) threshold value
- `kernel_size`: size of kernel, aperture for the Sobel operator

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

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

4.1.3 Edge segmentation

The geometric relation can not constructed from the edges, it always construct 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 of representing an edge, we can represent a set of approximate lines of it. This way also useful when we want presentation the edges or describe the relation between it. With the set of step edges was obtained from find contours (the image structure). In this step, we will segment it to approximated lines. The method to segment the edges is a recursive algorithm^[2] but it have some change in the “stop condition” of algorithm to easy process, as follows:

- Establish a line “ l ” between two endpoints of 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 split at this point. The value chosen for α in the program is 3 ($\alpha = 3$).
- Reprocess both parts which was obtained from step 3.
- The algorithm continues until all edges fragments are represented.

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

The algorithm is presented as follows:

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

```

Algorithm 6: Algorithm to segment an edge

4.2 Pairwise geometric histogram

Pairwise geometric histogram(PGH) is used to encode the relative information between a line and a set of lines in an object. Therefore, an object can represented by a set of PGH. From the set of PGH, we can reconstructed the object or compare with another object. In this section, we will mention the constructing a PGH for an object based on the geometrical relationship and compute the similar distance between two objects.

4.2.1 Pairwise geometric histogram

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

The proceed to construct the PGH was described in below:

- Choose the reference line (called reference line, another called object line)

- Compute the angle between two lines
- Calculate the perpendicular distance from two endpoints of object line to the reference line (assigned d_{min} and d_{max}).
- Recording the perpendicular distance and angle relative between two lines.

Example ³:

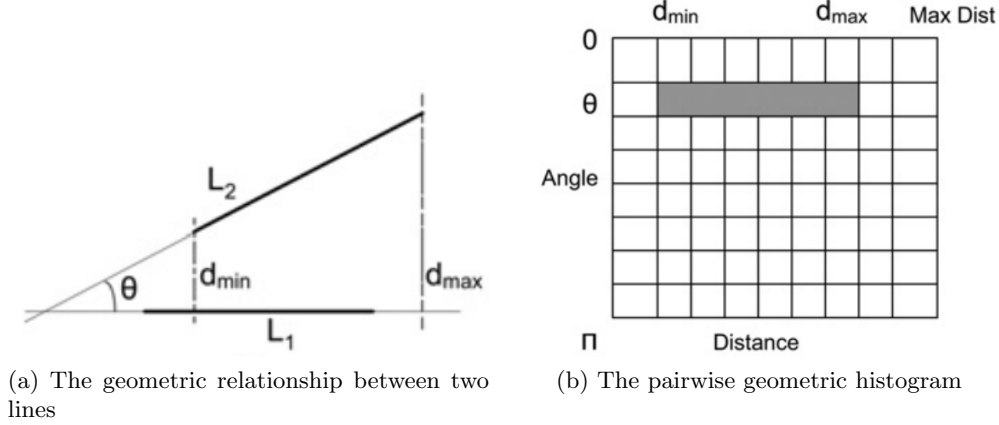


Figure 4.1: The geometric features and the PGH

The frequency of the geometric features is recorded as a two dimensional histogram with an angle axis ($0 - \pi$) and distance axis (range of perpendicular distance, d_{max} is the maximum distance on all distance of two arbitrary lines). The entries on PGH describe 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.

The full object representation is constructed by recording the PGH for each line within object. If the object is defined by n lines, the full shape representation will composed of n pairwise geometric histograms.

This method still 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 is invariant.

4.2.2 Histogram matching

“The histogram matching enables robust classification of shape features by finding similarity between the scene and reference model”^[1]. The similar between two models can obtain via the similar distance, which was computed by comparing their probability distribution on geometric histogram. In program, each image was represented by a comprises of many geometric histograms and using the Bhattacharyya metric to determine the similar distance between two models ^[1]. In general, we have normalize the histograms before comparing. The form of Bhattacharyya metric used to compute the degree of 2 model:

$$d_{Bhattacharyya}(H_i H_j) = \sum_{\theta}^{\pi} \sum_d^{d_{max}} \sqrt{H_i(\theta, d) H_j(\theta, d)} \quad (4.1)$$

³Images extract from the article [1]

The significance of parameters in the formula 4.1, 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

By the default, the range of angle axis from 0 to 180 degree (correspondence with 180 degree). Based on the accuracy of program, we can increase the range of angle axis. This design allow increase the range of angle axis to several time with default value. Example, the table below show the result when calculating Bhattacharyya distance between image *Md 028.JPG* and some images with difference accuracy:

Reference image	Scene image	180	2 * 180	4 * 180	6 * 180
Md 028.JPG	Md 001.JPG	0.977953	0.964167	0.93861	0.91471
Md 028.JPG	Md 005.JPG	0.96479	0.943657	0.906444	0.871756
Md 028.JPG	Md 010.JPG	0.976241	0.958061	0.925943	0.896445
Md 028.JPG	Md 027.JPG	0.980728	0.968233	0.945442	0.92485

Besides the Bhattacharyya metric, we can also choose another metric to matching the histograms, such as: **Chi-squared** metric and **Intersection** metric. The forms was presented as below:

Chi-squared metric:

$$d_{Chi-squared}(H_i H_j) = \frac{\sum_{\theta} \sum_d^{\pi} \left(\frac{(H_i(\theta, d) - H_j(\theta, d))^2}{(H_i(\theta, d) + H_j(\theta, d))} \right)}{2} \quad (4.2)$$

Intersection metric

$$d_{Intersection}(H_i H_j) = \sum_{\theta} \sum_d^{\pi} \min(H_i(\theta, d), H_j(\theta, d)) \quad (4.3)$$

The significance of parameters in equation (4.2) and (4.3) is similar with (4.1). 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, depend on the purpose of comparison will choose a suitable comparing method. In this program, we want to try on three method to have a general view result when matching the histograms.

4.2.3 Statistical the matching result

Based on the result of matching histogram. In this step, we will do a statistical to classify the result into other groups.

Chapter 5

Implementation

5.1 Software architecture

Continue the IMP tool, all the functions for this task was saved in the **impls_2015** package of program. Besides the method was created by myself, I also use some methods from the OpenCV (library for image processing) and Qt framework (framework for C++).

The class diagram¹ in 5.1 show mainly classes of my task. The *mainly* methods located in

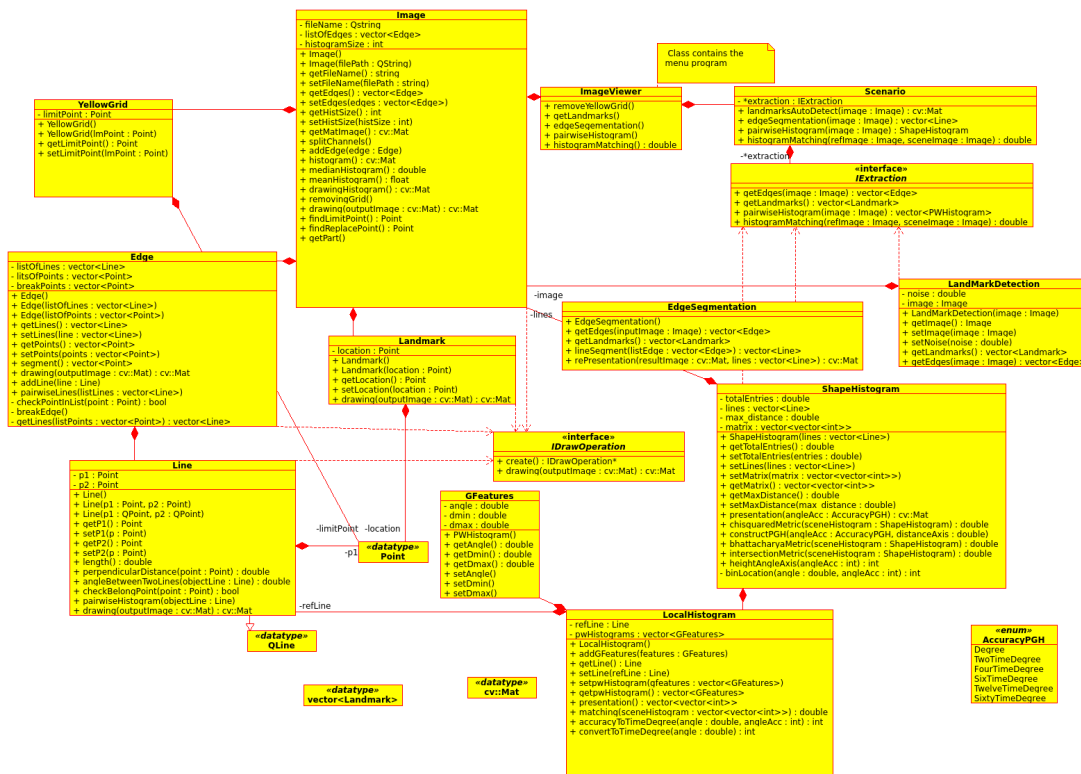


Figure 5.1: The class diagram of program

the *ImageViewer* class, where contains all functions of the software. To represent the information of image and preprocessing about clear the yellow grid, we use the classes such as *Line*, *Edge*, *Landmark*, *YellowGrid*, *Image* class. For the edge segmentation, construct the geomet-

¹See the full image in Appendix

ric histogram and landmarks detection, we have *GFeatures*, *LocalHistogram*, *ShapeHistogram*, *EdgeSegmentation*, *LandmarkDetection* class. All the main functions were inherited from the *IExtraction* interface and used in *main* class via the *Sceneario* class. The aim, properties and methods in each class was discussed in below sections.

5.2 Image preprocessing

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

5.2.1 Line class

Line class describe the information of a straight line and the methods can do with a line.

The attributes

- **p1**: the first endpoint of line.
- **p2**: the second endpoint of line.

The methods

- **Line()**: Constructor an empty line.
- **Line(p1: Point, p2: Point)**: Constructor a line with two endpoints p1, p2. With type of the endpoints is **Point** (in OpenCV)
- **getP1()**: Getter the first endpoint
- **setP1(p:Point)**: Setter the first endpoint
- **getP2()**: Getter the second endpoint
- **setP2(p:Point)**: Setter the second endpoint
- **length()**: Calculate the length of line
- **perpendicularDistance(point:Point)**: Compute the perpendicular distance from point “point” to the line
- **angleBetweenTwoLines(objectLine: Line)**: Compute the angle between two lines.
- **checkBelongPoint(point: Point)**: Check a point stay on the line or not. If the point stay on the line, the return value is true; otherwise, return false.
- **intersection(objectLine: Line)**: Finding the intersection point of two lines. If two lines have intersect, the method will return the coordinate of intersection point; otherwise, method will return a negative point.
- **pairwiseHistogram(objectLine: Line)**: Finding the geometric features between two lines. The return value is an **GFeatures** object, it includes the information of angle between two lines, the distance from two endpoints of *objectLine* to the reference line.
- **equationOfLine()**: Calculate the equation of line.
- **drawing(cv::Mat outputImage)**: Drawing the line on the output image.

5.2.2 Edge class

5.2.3 Image class

5.3 Automatic classification

about methods

5.4 Result

result...

Chapter 6

Conclusion

about conclusion

Bibliography

- [1] Sasirekha Palaniswamy, Neil A Thacker, and Christian Peter Klingenberg. Automatic identification of landmarks in digital images. *IET Computer Vision*, 4(4):247–260, 2010.
- [2] PA Riocreux, Neil A Thacker, and RB Yates. An analysis of pairwise geometric histograms for view-based object recognition. In *BMVC*, pages 1–10, 1994.