MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université de Paris

Master 1 Computer Science TER Project Report

Search for ordered series of patterns by combining descriptor of Generic Fourier and spatial structural representation

---

Mohamed Yakoub Azzouni – Bouzid Arezki

Année universitaire : 2019 – 2020

Supervisor : Laurent Wendling

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher Laurent Wendling who gave us the golden opportunity to do this wonderful project on the topic (Search for ordered series of patterns by combining descriptor of Generic Fourier and spatial structural representation), which also helped us in doing a lot of Research and we came to know about so many new things we are really thankful to him.

Secondly we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

ABSTRACT

The symbols are graphical shapes, in two dimensions, integrating into their composition higher level conceptual information.

Extraction and recognition of symbols, inside a document, are key steps towards understanding and interpretation of its content. When you consider a large database of technical documents in our case it's (maps of avionics type), it can be interesting to search on the fly for a particular symbol in a set of images and determining whether the corresponding patterns are structured together for example circles or segments following a straight line.

In this document we propose to study a hybrid approach which is composed of two steps firstable statistical by Generic Fourier descriptor, in the seconde place we site the structural methode which use composition rules following a direction for example, and that's to extract series of patterns.

TABLE OF CONTENTS

LISTE OF FIGURES

LISTE OF TABLES

GENERAL INTRODUCTION

1. General Introduction

Pattern recognition is used in several fields, mainly in medical imaging, text recognition for indexing purposes and automatic document archiving, in biometrics, in the post office for automatic mail sorting, in fingerprint detection, etc.

As the advancement of computers makes it possible to distribute operating systems with high computing power at reasonable prices, the discipline is making great strides in the industry. It brings together a variety of technologies ranging from the simple identification of isolated objects to the analysis of complex scenes. The problem that symbol recognition seeks to solve consists of associating tags with data that can take the form of original images. Now, several general methods have been developed in the field of pattern recognition to automatically extract information from sensitive data in order to characterize categories of patterns and automatically assign data to these categories. The implementation of these problem-specific general methods has led to the introduction of the concept of a recognition process, which raises the question of integrating pattern recognition methods into systems designed to recognize patterns. These domains focus identification problems on specific data, but in addition, they allow people to locate and identify in a broader image interpretation process, which involves the perception and level of knowledge of specific domains. In this project work we set ourselves the objective of modeling and designing a pattern recognition application that will be used to detect and search for ordered series of patterns. Our project is divided into 4 chapters, the first chapter is the segmentation phase where we explain how to extarct the connected components, the second one is the methodes of classifications, the third talks about the methology used in other terms the implementation of an apllication which has as an objective « searching for ordered series of patterns » finaly we’re gonna do the evaluation of our methode.

PRETREATMENT AND SEGMENTATION PHASE

2. Pretreatment and Segmentation phase

2.1 Introduction

In this chapter, we are going to talk about the transformations necessary for processing shapes. They show the different aberrations of the acquisition system against which it is necessary to guard before undertaking any action of analysis.

2.2 Preprocessing phase

Let's suppose we have a scanned image that contain symbols. can we improve this image with filtering ? any filtering technique that works with scanned image can be helpful to improve or deteriorate the visual aspect of the image, according to the original characteristics of the image.

2.2.1 Image binarization

Binarization is the step to do before treatment phase. It make distinguishing symboles easier, it consist to assigning to each pixel of the image a value 0 or 1 and to do this it applies the thresholding operation:

- 0 >>> represent the black or the symbols in other words,
- 1 >>> represent the white which is the background,

2.3 Segmentation tools

The analysis of the image calls for segmentation where we will try to associate with each component of the image a label which is the id of the component plus the width and height of the component, we also associate the centroide of the component and the point where it start. The image segmentation thus defined is a vast field where there are many approaches.

The main objective of the segmentation is to devide the image into several parts, segments having same characteristics or attributes, there are some examples where segmentation is used :

- Content-based image extraction,
- Medical Imaging,

2.3.1 Region Segmentation

Segmentation is a process which consists in dividing an image into connected regions.

The union of these regions must give back the initial image. Region segmentation is an important step for the extraction of qualitative information from the image.

Many practical problems in image analysis require prior analysis where the image must be divided into homogeneous regions separated by contours. In general, homogeneous regions are regions where the light intensity varies slowly with spatial coordinates. The contours separating these regions are narrow portions of the image where the intensity variations are large. There are a multitude of methods for detecting contours, hence another approach to image segmentation is to search for the contours of regions. The different edge detection methods used are Sobel, Prewitt, Robert, Canny. These methods have been proposed to detect transitions in images.

Among the regional segmentation techniques the conneced components segmentation. It provides a high level description, each region has a label giving qualitative informations (size, position, orientation). The image is therefore reduced to matrices where each matrix is a conected component which contains all the informations about the object or symbol. Topological or spatial information can also be stored, such as the fact that one region is above another. The interest of detecting regions is to be able to manipulate them afterwards to extract characteristics of shape, position, size...

2.4 Segmentation challenges

Applications where the information processed is symbols or forms, separating components, extracting symbols. Our task will be to find as simple and robust methods as possible to Isolate each symbol in a component.

2.5 Conclusion

Segmentation remains an active subject, despite the multitude of documents, because there are no general solutions, but also because there are many possible applications. The first is of course to be able to increase the performance of the indexing of multimedia content, in order to facilitate access to information. A good method of segmentation will therefore be the one that will lead to a good interpretation. It must have simplified the image without having reduced the content too much. Among other things, it must avoid irreversible choices.

FEATURES AND CLASSIFICATION

3. Features and Classification

3.1 Introduction

The purpose of this chapter is to describe the choice of characteristics to code the shapes or symbols as an indexing phase and then explain the techniques used for the classification phase. So depending on the nature of the extracted forms, their coding varies. Anyway, an invariant coding under certain transformations is sought, if possible robust to noise and these codes (characteristics) must be discriminating as much as possible.

3.2 Generic Fourier Descriptor

Before describing the Generic Fourier Descriptor (GFD) in detail, the main steps of the extraction of the generic Fourier descriptors are illustrated in the following figure:

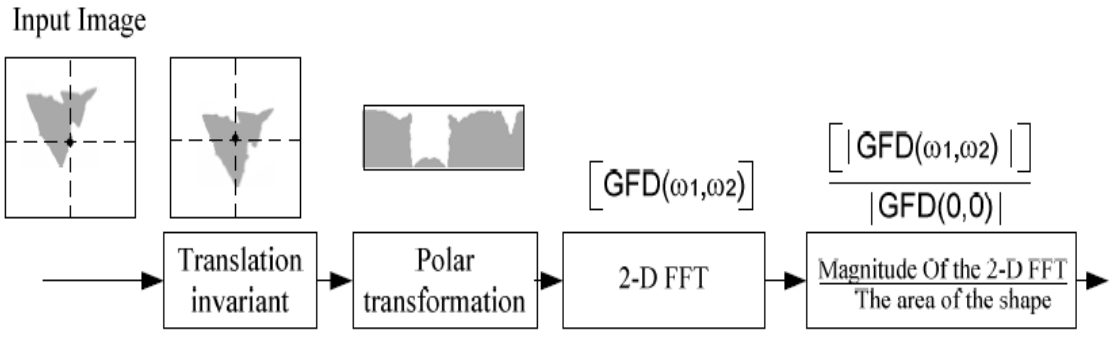


Figure 3.1 : generic fourier descriptor

3.2.1 One Dimensional Fourier Descriptor

A one-dimensional FD has been successfully applied to many applications for shape representation, including character recognition. It is invariant under certain transformations and robust to noise. Generally, One-Dimensional Fourier Descriptor (1-D FD) is obtained by Fourier Transform (FT) on a shape signature function derived from the shape boundary coordinates {(x (t), y (t)), t = 0, 1, ..., N-1}.

A typical form of signature function is the centroid distance function which is given by the distance of the boundary point from the center of the form (xc, yc).

$R(t) = ([x(t) - x_e]([x(t) - x_e]^2 + [y(t) - y_e]^2)^{1/2}$ **t = 0, 1, ….., N – 1**

Where :

$$X_e = \frac{1}{N}\sum_{t=0}^{N-1} x(t) \qquad y_e = \frac{1}{N}\sum_{t=0}^{N-1} y(t)$$

An example of a centroid distance function for an apple shape :

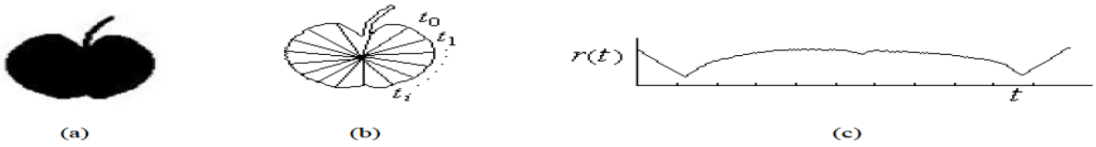


Figure 3.2 : (a) apple figure; (b) the contour of (a); (c) centroid distance function of (a) [D. S. Zhang ang G. Lu 2001]

A dimensional FT is then applied to r(t) to obtain the Fourier transform coefficients.

$$A_n = \frac{1}{n}\sum_{t=0}^{N-1} r(t)\exp(\frac{-j2\pi nt}{N}) \qquad n = 0, 1, \ldots\ldots\ldots, N - 1$$

The magnitudes of the coefficients an(n = 0, 1, ..., N-1) normalized by the magnitude of the first coefficient $a0$ are used as shape descriptors, called Fourier descriptors. The acquired FDs are invariant in transformation, rotation and scale.

It has been shown that shape representation using a Fourier descriptor (FD) prevents many other contour shapes [[D. S. Zhang ang G. Lu 2001], [Hannu Kauppinen, Tapio Seppanen and Matti Pietikainen 1995]]. However, all these methods require knowledge of shape boundary information which may not be available in general situations.For example, it is difficult to derive 1-D FD for the shape of Figure 3.2: (a) because the contour of the shape is not available.

In addition, 1-D FD cannot capture the inner content of the form which is important for form discrimination. For example, FD is not capable of distinguish the shape of Figure 3.3: (b) from that of Figure 3.3: (c). The disadvantages limit the application of FD 1D.

(a) (b) (c)

Figure 3.3 : (a) one shape without contour, (b) and (c) two shapes having the same contour but different inner content [D. S. Zhang ang G. Lu 2001].

3.2.2 Polar Fourier Transform

The Fourier transform has been widely used for image processing and analysis. The advantage of image analysis in the spectral domain over shape analysis in the spatial domain is that it is easy to overcome the noise problem that is common [Eric Persoon and King-sun Fu 1997].

In addition, the spectral characteristics of an image are generally more concise than those extracted from the space domain. A dimensional FT has been successfully applied to the contour shape to derive FD. The application of the one-dimensional shape to the shape presupposes the knowledge of information about the limits of the shape. There is no work reported on the FD region. In this section, we introduce generic DFs derived from PFT 2-D [Dengsheng Zhang 2002].

The continuous and discrete 2D Fourier transforms of an image of form f (x, y)

(0≤x <M, 0≤ y <N) are given by these equations respectively.

$$F(u,v) = \int_x \int_y f(x,y)\exp[-j2\pi(ux+vy)]\,dx\,dy \qquad (3.1)$$

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] \qquad (3.2)$$

The u and v values in second equation are the uth and vth spatial frequencies in the horizontal and vertical directions, respectively. The two-dimensional FT mode can be applied directly to any shaped image without assuming knowledge of the contour information. However, the direct application of 2-D FT directly to a shape image in Cartesian space to derive FDs is not practical because the functions captured by 2-D FT are not rotationally invariant. The

rotational invariance of a shape is important because similar shapes can be oriented differently. For example, the two models in Figure 3.4: (a) and (b) are similar models (shapes), however, their Fourier spectrum distributions (Figure 3.4: (c) and (d)) in terms of frequencies are different [P. J. van Otterloo. 1991].

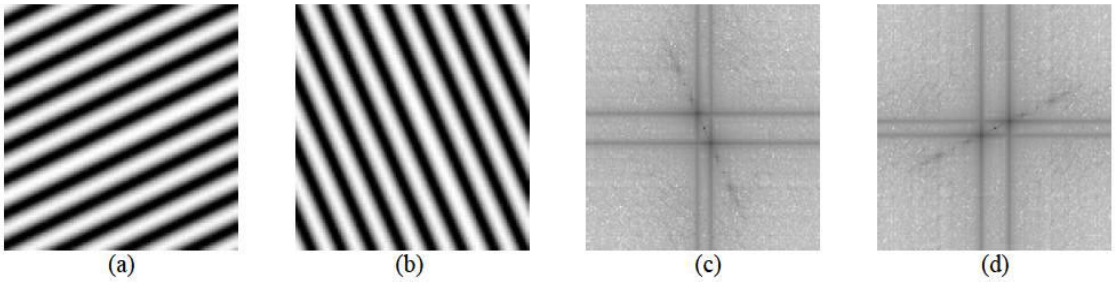


Figure 3.4 (a) a pattern; (b) pattern (a) rotated by 90 degree; (c) Fourier spectra of (a); (d) Fourier spectra of (b).

Therefore, we consider the shape image in polar space and the application of the Polar Fourier Transform (PFT) on the shape image. The PFT produces rotation-invariant data, which is particularly well suited for the precise extraction of orientation characteristics.

To derive PFT, both the data f(x, y) and the spectra F(u, v) are put into polar space, that is, let

$$X = r.\cos\theta,\ y = r.\sin\theta$$

$$U = \rho.cos\psi\ ,\ v = \rho.sin\psi \qquad (3.3)$$

(r, θ) is the polar coordinates in image plane and (ρ,ψ) is the polar coordinates in frequency plane.The differentials of x and y are:

$$dx = \cos\theta\, dr - r\sin\theta\, d\theta$$

$$dy = \sin\theta\, dr - r\sin\theta\, d\theta \qquad (3.4)$$

The Jacobian of (3.4) is r. By substituting (3.3) and () into (3.4) into (3.1) we have the polar Fourier transform (PFT1) :

$$PF_1(\rho,\psi) = \int_r \int_\theta rf(r,\theta)\exp[-j2\pi r\rho\sin(\theta+\psi)]\,dr\,d\theta \qquad (3.5)$$

The discrete PFT1 is then obtained as

$$PF_1(\rho_l,\psi_m)=\sum_p\sum_i f(r_p,\theta_i)\cdot r_p\cdot\exp[-j2\pi r_p\rho_l\sin(\theta_i+\psi_m)] \quad (3.6)$$

where rp = p/R, θi = i(2π/T) (0≤ i<T); ρl = l (0≤ l<R) and ψm = mθi. R and T are the resolution of radial frequency and angular frequency respectively. The acquired polar Fourier coefficients PF1(ρl, ψm) are used to derive normalized FD for shape representation.

PFT1 is the direct result from the polar transform of (3.1). However, due to the presence of ψm within sine function sin(θi +ψm), the physical meaning of ψm is not the mth angular frequency. The features captured by the PFT1 lose physical meaning in circular direction. To overcome the problem, a modified polar FT (PFT2) is derived by treating the polar image in polar space as a normal two-dimensional rectangular image in Cartesian space. Figure 3.5 demonstrates the rectangular polar images. For example, Figure 3.5(a) is the original shape image in polar space, Figure 3.5(b) is the rectangular polar image plotted into Cartesian space.

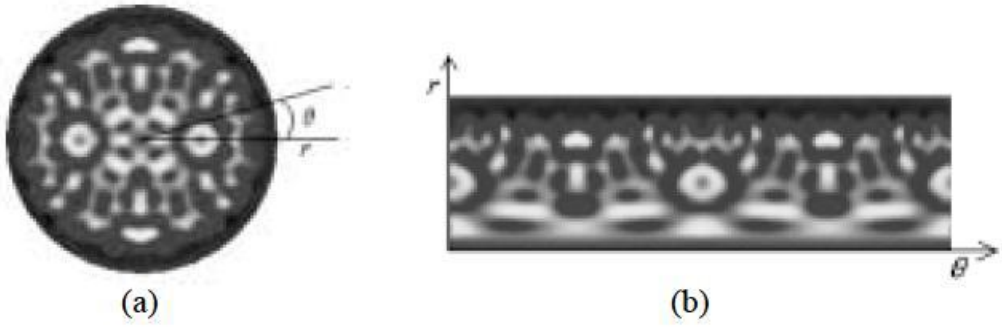


Figure 3.5 (a) original shape image in polar space ; (b) polar image of (a) plotted into Cartesian space.

The polar image in Figure 3.5(b) is the normal rectangular image. Therefore, if 2-D FT is applied on this rectangular image, the polar FT has the similar form to the normal 2-D discrete FT of (3.2) in Cartesian space. Consequently, the modified polar FT is defined as

$$PF_2(\rho,\phi)=\sum_r\sum_i f(r,\theta_i)\exp[j2\pi(\frac{r}{R}\rho+\frac{2\pi i}{T}\phi)] \quad (3.7)$$

where 0≤r<R and θi = i(2π/T) (0≤ i<T); 0≤ρ<R, 0≤φ<T. R and T are the radial frequency resolution and angular frequency resolution respectively. PFT2 has a simpler form than PFT1. There is no need to the physical meaning of ρ and φ in (3.7) is similar to u and v in (3.1) and (3.2). The ρ and φ are simply the radial frequency and the angular frequency respectively.

The determination of the number of ρ and φ is physically achievable, because significant shape features are usually captured by a few lower frequencies. Figure 3.6(a)(b) shows the polar images of the two patterns in Figure 3.4(a)(b) and their polar Fourier spectra are shown in Figure 3.6(c) and (d). It can be observed from Figure 3.6 that rotation of pattern in Cartesian space results in circular shift in polar space. The circular shift does not change the spectra distribution on polar space. If phase is ignored, the spectra is shift or translation invariant. This is demonstrated in Figure 3.6(c) and (d). The polar Fourier spectra is more concentrated around the origin of the polar space. This is particularly well-suited for shape representation, because for efficient shape representation, the number of spectra features selected to describe the shape should not be large. Since f(x, y) is a real function, the spectra is circularly symmetric, only one quarter of the spectra features are needed to describe the shape. The acquired polar Fourier coefficients F(ρ, φ) are used to derive normalized FD for shape representation.

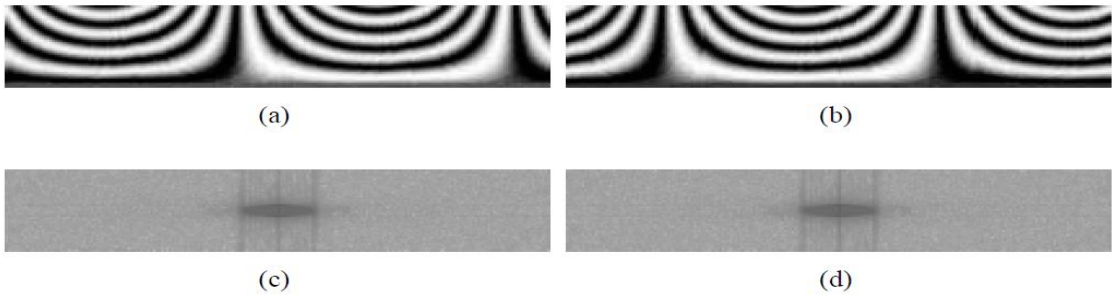


Figure 3.6 : (a)(b) polar images of the two patterns in Figure 3.4(a) and (b); (c) Fourier spectra of (a); (d) Fourier spectra of (b).

3.3 Derivation of Geniric FD

In this section, the derivation of FD using the above described PFTs is given in details. The

two polar FTs: PFT1 and PFT2 are both implemented in the experiments to derive FD for the purpose of determining which one is the more desirable for shape retrieval.

Given a shape image I = {f(x, y); 0≤x<M, 0≤y<N}. To apply PFT, the shape image is

converted from Cartesian space to polar space Ip = {f(r, θ); 0≤r<R, 0≤θ<2π}, R is the maximum radius of the shape. The origin of the polar space is set to be the centroid of the shape, so that the shape is translation invariant. The centroid (xc, yc) is given by the PFTs are applied on Ip. The acquired coefficients of the two PFTs are translation invariant due to the use of centroid as polar space origin. Rotation invariance is achieved by ignoring the phase information in the coefficients and only retaining the magnitudes of the coefficients. To achieve scale invariance, the first magnitude value is normalized by the area of the circle in which the polar image resides or the mass of the shape , and all the other magnitude values

are normalized by the magnitude of the first coefficient. The translation, rotation and scale normalized PFT coefficients are used as the shape descriptors. To summarize, the shape descriptor derived from PFT1 and PFT2 are FD1 and FD2 respectively, they are shown as following where m is the maximum number of the radial frequencies selected and n is the maximum number of angular frequencies selected. m and n can be adjusted to achieve hierarchical coarse to fine representation requirement. Normally, the first coefficient, or the DC component is used as the normalization factor and is discarded after normalization. However, this component is used as an additional feature, because it reflects the average energy (scale) of the shape which is useful for shape description. For efficient shape description, only a small number of the acquired FD features are selected for shape representation. The selected FD features form a feature vector which is used for indexing the shape. For two shapes represented by their FD features, the similarity between the two shapes is measured by the city block distance between the two feature vectors of the shapes. Therefore, the online matching is efficient and simple [Dengsheng Zhang 2002].

3.4 Implementaion of GFD

The implementation of GFD can be summarized into 4 major steps, translation normalization, polar Fourier transform, rotation normalization and scale normalization. The algorithm for deriving GFD using PFT2 is given in the following. The algorithm for deriving GFD using PFT1 is similar, with the difference in the basis calculation of polar Fourier transform step.

Algorithm of computing GFD:

1. Input shape image data f(x, y);
2. Get centroid of the shape (xc, yc);
3. Set the centroid as the origion; /* translation normalization */
4. Get the maximum radius of the shape image (maxRad);
5. Polar Fourier transform

```
For radial frequency (rad) from zero to maximum radial frequency (m)
  For angular frequency (ang) from zero to maximum angular frequency (n)
    For x from zero to width of the shape image
      For y from zero to height of the shape image
    {
        radius = square root[(x-xc) 2 + (y-yc) 2 ];
```

```
theta = arctan2[(y-xc)/(x-yc)]; /* theta falls within [–π, +π] */
if (theta<0) theta += 2π; /* convert theta to [0, 2π] */
FR[rad][ang] += f(x,y)×cos[2π×rad×(radius/maxRad) + ang×theta]; /*real part*/
FI[rad][ang] −= f(x,y) ×sin[2π×rad×(radius/maxRad) + ang×theta]; /*imaginary*/
}
```

6. Calculate GFD

```
For rad from zero to m
 For ang from zero to n
{
/* rotation and scale normalization */
  If (rad=0 & ang=0)
    DC= square root[(FR2[0][0] + FR2[0][0])];
    GFD[0] = DC/(π×maxRad2);
  Else
   GFD[rad×n+ang] = square root[(FR2[rad][ang] + FI2[rad][ang])]/DC;
}
```

7. Output feature vector GFD [Dengsheng Zhang 2002].

3.5 Classification Methode

After calculating the output feature vector GFD of the symbols, we assosiate for each symbol an image therefore after the segmentation phase the output feature vector GFD is calculated for each connected component in order to classify them in the right images by using the manhattan distance with a threshold where the manhattan distance (The taxicab distance), is the distance (d1) between two vectors P, Q in an n-dimensional real vector space with fixed Cartesian coordinate system, the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. More formally [1],

d1(p,q) = ||p - q|| = $\sum_{i=1}^{n}$ ||Pi – Qi||

where p,q are vectors

P = (P1, ... Pn) and Q = (Q1, ... Qn)

For example, in the plane, the taxicab distance between (p1,p2) and (q1,q2) is

|p1 - q1| + |p2 - q2|.

After calculating the manhattan distance we take the minimum distance, then we compare the width and height of the connected componant with the width and the height of the symbol.

If it’s lower then the treshold of classification we classify the component as the symbol else we reject it.

3.6 Relational histogram

The relational histogram presents a new compact shape representation for retrieving line-patterns from images. The basic idea is to exploit both geometric attributes and structural information to construct a shape histogram. it's realized by computing the N-nearest neighbor graph in our work we took the 4-nearest neighbors graph for the lines-segments for each pattern. The edges of the neighborhood graphs are used to acces contributions to a two-dimensional pairwise geometric histogram. Shapes are indexed by searching for the line-pattern that maximizes the cross correlation of the normalized histogram bin-contents [Benoit Huet, Edwin R. Hancock 1999].

3.7 Conclusion

There are many methods of feature extraction. The generic Fourier descriptor is very promising for use in feature extraction because it has the following properties :

- Invariants by shape translation,
- Invariants by change of scale,
- Invariants by rotation,
- Invariants by change of origin.

To compare shapes, we compare their descriptors by using the manhattan distance to finally take the minimum distance and use a classification threshold to classify the component, which often performs well.To search for lines of symbols (series of pattern) the relational histogram gives good results to the given problem.

APPLICATION SETUP FOR SEARCHING FOR ORDERED SERIES OF PATTERNS

4. Application setup for searching for ordered series of pattterns

4.1 Introduction

This last chapter is devoted to the design of the application that will allow to classify the symbol and searching for ordered series of patterns using all the methods mentioned in the previous chapters. Our program is written with Visual studio software. It uses the OpenCV libraries to be able to perform shape processing and to use the mentioned methodes.

4.2 Work environment

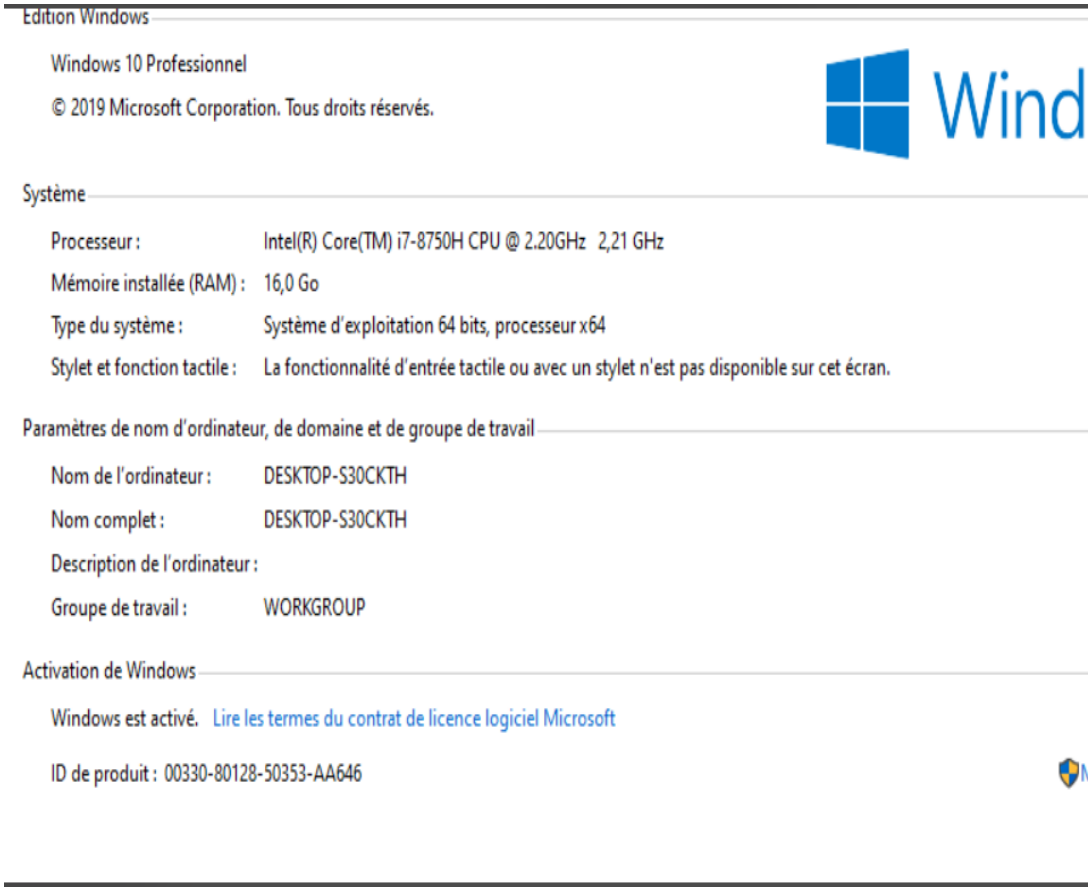


Figure 4.1 : work environment

4.3 Technologies used

4.3.1 C++

It is a compiled programming language, much used in the field of image processing where performance is a must [2].

4.3.2 OpenCv

OpenCV is an open source graphics library, it was designed and developed initially by the Intel group, this library is specialized in the field of image processing, and is distributed under the BSD license [3].

4.4 The schematics of the application

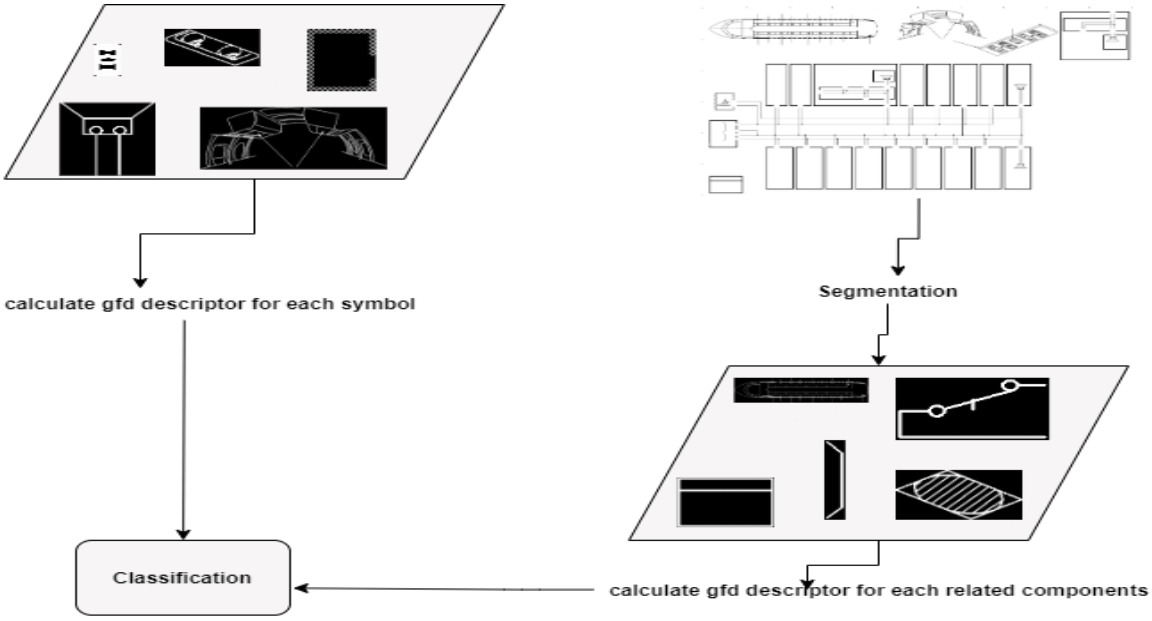


Figure 4.2 : extraction and classification of connected components

As we explained earlier in this paper work, the first step of our methode is the pre-processing phase which have the objective of improving the image and it's done with the binarisation then comes the segmentation phase in which we extract the CCS (connected components) furthermore we have the classification phase where we calculate the feature vector for each symbol of the database as it's shown in the figure 4.2 and the connected components with the help of GFD where the values of n and m are 9 and 4 respectively because it gave us the best results.The database contain the symbols realted for each image, each symbol can have several images with diffrent sizes. After calculating the feature vectors we compare them using the manhattan distance, in where we take the smallest distance found.

Afterwarsd we compare the the width and height of the connected component with the width and height of the symbol if it's lower then the classification treshold we classify the component as the symbol else we reject it.

After classifying the conected componants we create a matrix (S) in which each line is a symbol and each column is a connected componant classified as the symbol.

$$\mathrm{S}_{n.} = \begin{pmatrix} C_{11} & C_{12} & & \\ C_{21} & C_{22} & C_{23} & \\ C_{31} & C_{32} & \cdots & C_{3.} \\ \vdots & \ddots & \vdots & \vdots \\ C_{n1} & \cdots & \cdots & C_{n.} \end{pmatrix}$$

The matrix (S) is needed to draw the images related for each symbol, for exemple if we have 10 symbols in the database we will get as a result 10 images, each image contain the

CCS calssified as the symbol related to the image.

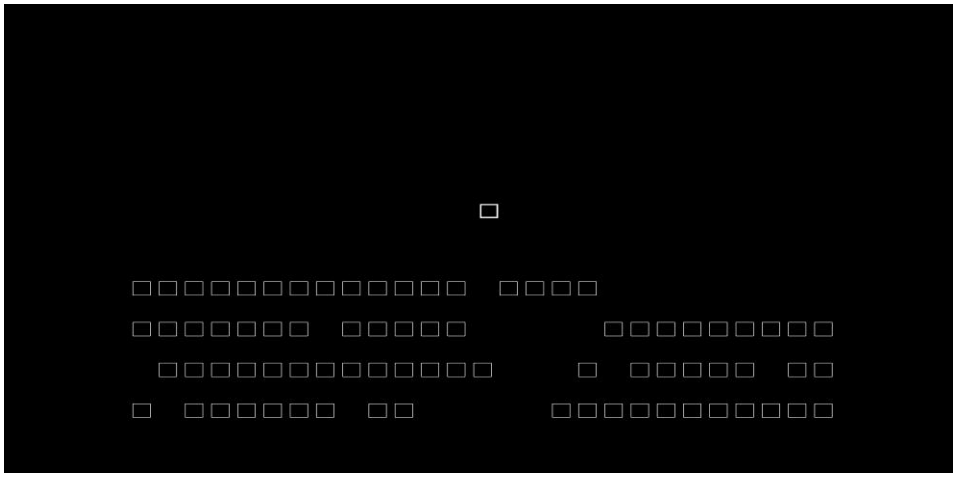


Figure 4.3 : image contain the connected components of squares

After that we create a matrix (D) where the value of the lines and colums represent the distance between the connected componnats for the same symbol, in a matter of fact the matrix (D) is created from the lines of the matrix (S) which contains more then two CCS.

$$\mathrm{D}_{nn} = \begin{pmatrix} dist_{11} & \cdots & dist_{1n} \\ \vdots & \ddots & \vdots \\ dist_{n1} & \cdots & dist_{nn} \end{pmatrix}$$

The distance is calculated as follows, for each two component we do the following

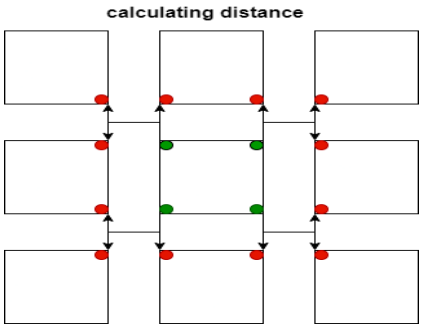


Figure 4.4 : methode of calculating the distance

The center square is the first component and the others are the positions where the second component could be located, we calculate the distance using the euclidean distance between the closest points of the two component as shown in the figure 4.4, as for the values we insert in the matrix (D) is the minimal distances calculated.

After all the previous steps comes the the detection of the series of patterns in which we use the matrix (D), for each CC (Connected Component) we create a tree with four leaves each one represent a CC, we chose the leaves by the distance calculed earlier in other terms the values of the matrix (D) related to the CC where we take the four smalest distances.The branches represent the distance between the CCs.

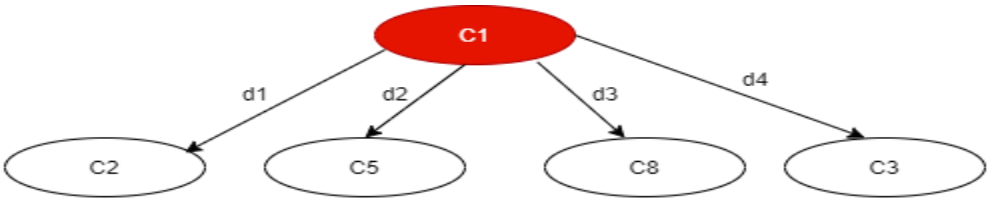


Figure 4.5 : the tree of one connected component

The next step is classifying the CCS as a line or not, it's done with the help of the relational histogram and SR (similarity report), first we calculate the relational histogram between three CCs where the distance is minimal, for exemple C2 and C5 knowing that d1<d2<d3<d4 in a way that we take two per two (C1, C2) (C5, C1), then SR is calculated. If it's bigger than 0.5 we classify it as a line else we try with the two other CCs if it isn't bigger then 0.5 too, we stop the treatment for this CC (father node). The similarity report is calculated as follows :

$$SR = \sum_{i=1}^{n} \min(HistoAB[i], HistoCA[i]) \,/\, \sum_{i=1}^{n} \max(HistoAB[i], HistoCA[i])$$

In our exemple (the figure 4.5) A = C1, B = C2, C= C5.

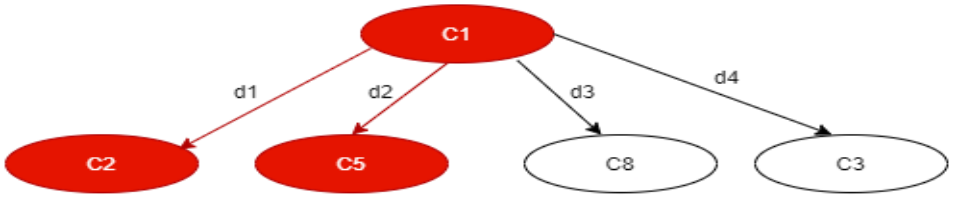


Figure 4.6 : the choice of the two connected components to detect the line

If the three CCS are considered as a line of patterns we do the same treatment for C2 and C5 which means C2 and C5 become the father node and it's done with the propagation. Furthermore we consider the three CCs classified as a line as one CC where we create an image which contain them, then for the treatment we calculate the relational histogram between the neightbors of C2, C5 and the image created, and we do this treatment in a loop until we can't find a line. If a CC is already treated we don't do the treatment another time.

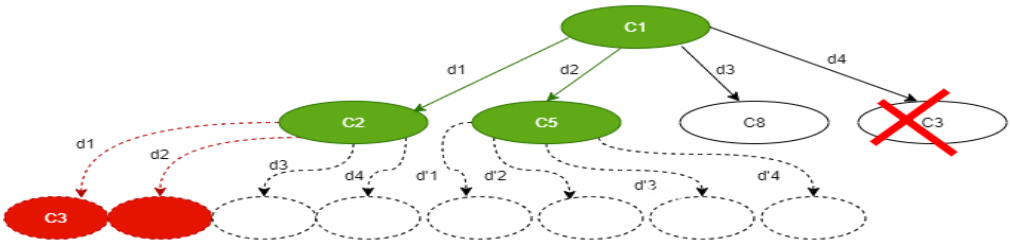


Figure 4.7 : deleting the components already treated

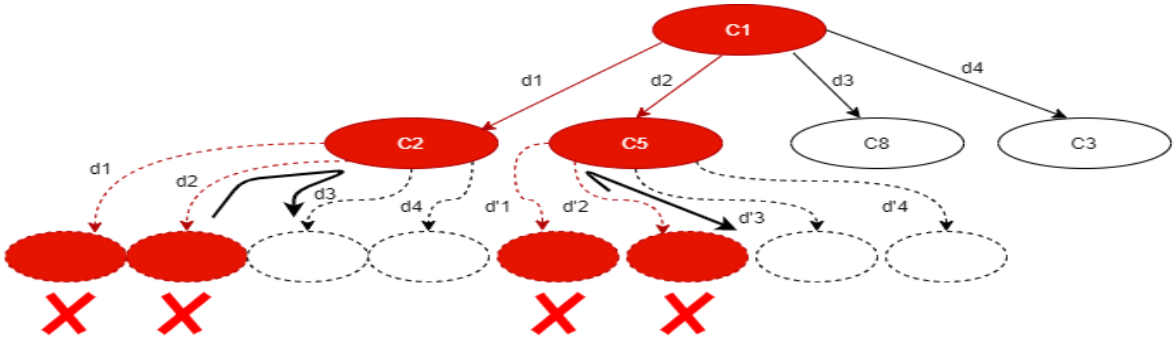


Figure 4.8 : end of treatment of the components colored with red

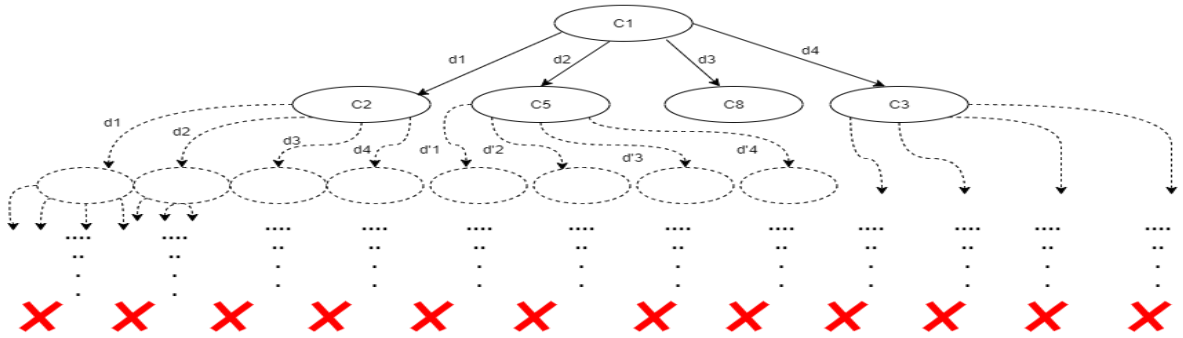


Figure 4.9 : end of treatment

4.5 Conclusion

We have succeeded in developing a pattern recognition algorithm using static images. This system makes it possible to process the images to make the classification of the symbols and the detections of series of patterns. This work has allowed us to better understand the theme of pattern recognition which involves image processing.

EVALUATION

5. Evaluation

5.1 Introduction

In this section we discuss the results of the execution of our code. Where we are going to site the confusing table and the accuracy related to it. Finaly we are going to conclude our work.

5.2 Evaluation

5.2.1 Exemple 1

-Ground Truth :

CCS	***symbols***	***background***	***diode***	***Arrow***	***rectangle***	***Symbol(1)***	***Symbol(2)***	***Symbol(3)***	***Symbol(4)***
99	34	1	4	1	13	15	1	2	2
Symbol (5)	***Symbol (6)***	***Symbol (7)***	***Symbol (8)***	***Symbol (9)***	***Symbol (10)***	***Symbol (11)***	***Symbol (12)***	***Symbol (13)***	***Symbol (14)***
2	2	2	1	1	1	1	1	1	1
Symbol (15)	***Symbol (16)***	***Symbol (17)***	***Symbol (18)***	***Symbol (19)***	***Symbol (20)***	***Symbol (21)***	***Symbol (22)***	***Symbol (23)***	***Symbol (24)***
1	1	1	1	1	1	1	1	1	1
Symbol (25)	***Symbol (26)***	***Symbol (27)***	***Symbol (28)***	***Symbol (29)***	***Symbol (30)***	***Symbol (31)***			
1	1	1	1	9	11	13			

Table 5.1 : ground truth (exemple 1)

-Binarisation threshold

threshold	0	20	50	80	100	120	150	255
Number of CCs detected	1855	1029	101	**99**	**99**	**99**	98	2

Table 5.2 : percentage of detection of the connected components (exemple 1)

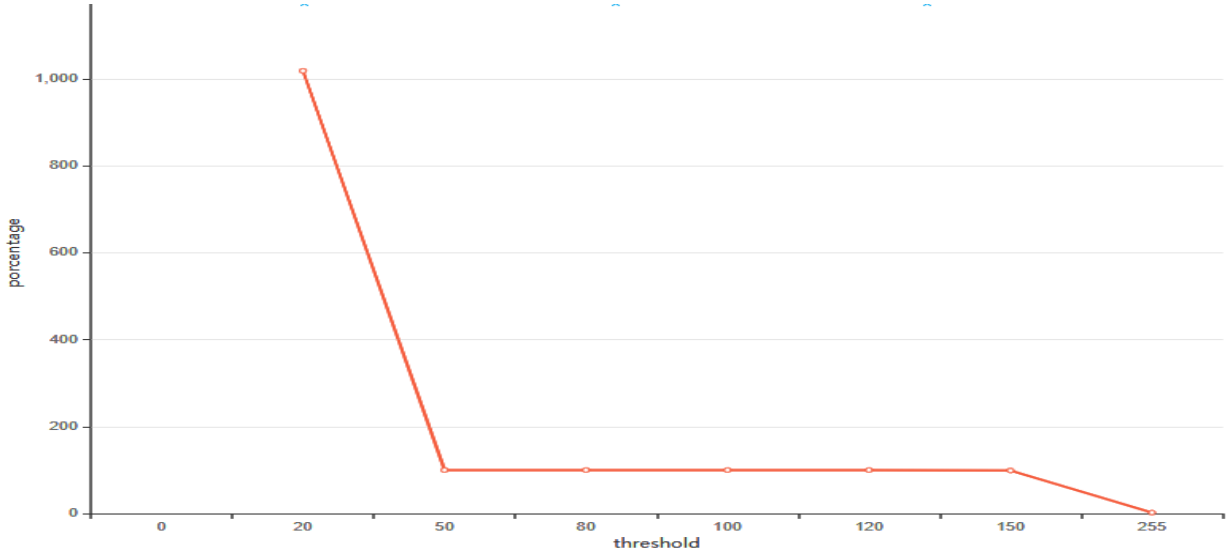


Figure 5.1 : graph of the percentage of detection of the connected components (exemple 1)

In this exemple we choosed the value 100 for the threshold of binarisation because it gave us the best results as shown in the table 5.2, and for the value of the classification threshold we choosed 20px (px = pixel) in width and height for the same reason.

-Results of classification

We notice that our methode gives good results for the classification of the symbols, where we obtained the precision of 100% for all the symbols except the symbol 17, 2 and 28, and the accuracy of 98%. However there is some mistakes where the symbol 2 was classified as the symbol 28 plus the symbol 17 was rejected where it shouldn't as shown in the confusion table (table 5.4)

This kind of mistakes are common due to the similary of the shape and the size of the symbols.

Note : in all the evaluation we didn't consider the background CC because it's not a symbol.

CCS	*symbols*	*diode*	*Arrow*	*rectangle*	*Symbol (1)*	*Symbol (2)*	*Symbol (3)*	*Symbol (4)*	*Symbol (5)*
98	34	4	1	13	15	0	2	2	2

Symbol (6)	*Symbol (7)*	*Symbol (8)*	*Symbol (9)*	*Symbol (10)*	*Symbol (11)*	*Symbol (12)*	*Symbol (13)*	*Symbol (14)*	*Symbol (15)*
2	2	1	1	1	1	1	1	1	1

Symbol (16)	*Symbol (17)*	*Symbol (18)*	*Symbol (19)*	*Symbol (20)*	*Symbol (21)*	*Symbol (22)*	*Symbol (23)*	*Symbol (24)*	*Symbol (25)*
1	0	1	1	1	1	1	1	1	1

Symbol (26)	*Symbol (27)*	*Symbol (28)*	*Symbol (29)*	*Symbol (30)*	*Symbol (31)*	*Class Reject*
1	1	2	9	11	13	1

Table 5.3 : results of classification (exemple 1)

nage 1 :

	D	A	R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Reject class
D	4																																		
A		1																																	
R			13																																
1				15																															
2					0																										1				
3						2																													
4							2																												
5								2																											
6									2																										
7										2																									
8											1																								
9												1																							
10													1																						
11														1																					
12															1																				
13																1																			
14																	1																		
15																		1																	
16																			1																
17																				0															1
18																					1														
19																						1													
20																							1												
21																								1											
22																									1										
23																										1									
24																											1								
25																												1							
26																													1						
27																														1					
28																															1				
29																																9			
30																																	11		
31																																		13	

Table 5.4 : Confusion table (exemple 1)

-line detection results

	diode	*rectangle*	Symbol (1)	Symbol (29)	Symbol (30)	Symbol (31)
True positive	0	1	2	1	1	2
False positive	1	1	0	0	0	1

Table 5.5 : line detection result (exemple1)

In the line detection results we had some errors due to the bad configuration of the parameters in the relational histogram function in a way that the bigger the interval is the more the results are precise but the computing time is bigger too. Here are some exemples of line detections.

Note : the green symbol is the father node.

Figure 5.2 : bad line detection (exemple 1)

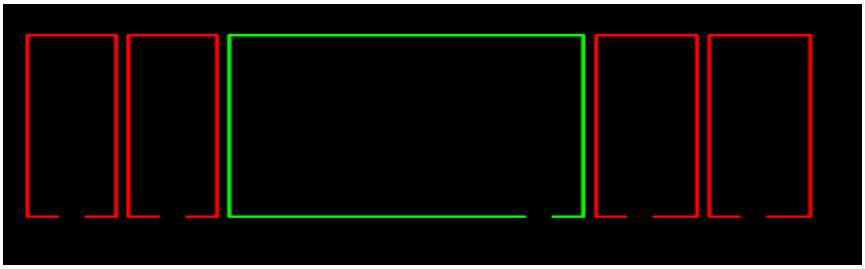


Figure 5.3 : good line detection (exemple 1)

5.2.2 Exemple 2

-Ground truth

CCS	symbols	background	*Horizontal rectangle*	*Vertical rectangle*	*Horizontal diode*	*Vertical diode*	*Circle*	*Symbol(1)*	*Symbol(2)*
711	29	1	218	333	3	21	31	15	4
Symbol (3)	*Symbol (4)*	*Symbol (5)*	*Symbol (6)*	*Symbol (7)*	*Symbol (8)*	*Symbol (9)*	*Symbole (10)*	*Symbol (11)*	*Symbol (12)*
2	18	2	10	3	2	1	1	1	2
Symbol (13)	*Symbol (14)*	*Symbol (15)*	*Symbol (16)*	*Symbol (17)*	*Symbol (18)*	*Symbol (19)*	*Symbol (20)*	*Symbol (21)*	*Symbol (22)*
2	2	1	1	1	1	1	1	1	1
Symbol (23)	*Symbol (24)*								
1	1								

Table 5.6 : Ground truth (exemple 2)

-Binarisation threshold

threshold	0	20	50	80	100	120	150	255
Number of CCs detected	4658	4148	706	**711**	**711**	696	693	2

Table 5.7 : percentage of detection of the connected components (exemple 2)

In this exemple we choosed the value 100 for the threshold of binarisation because it gave us the best results as shown in the table 5.6, and for the value of the classification threshold we choosed the value 11px in width and height for the same reason. We also rejected all the CCS where the width and the height are lower then 8x8 because it's considered as noises.

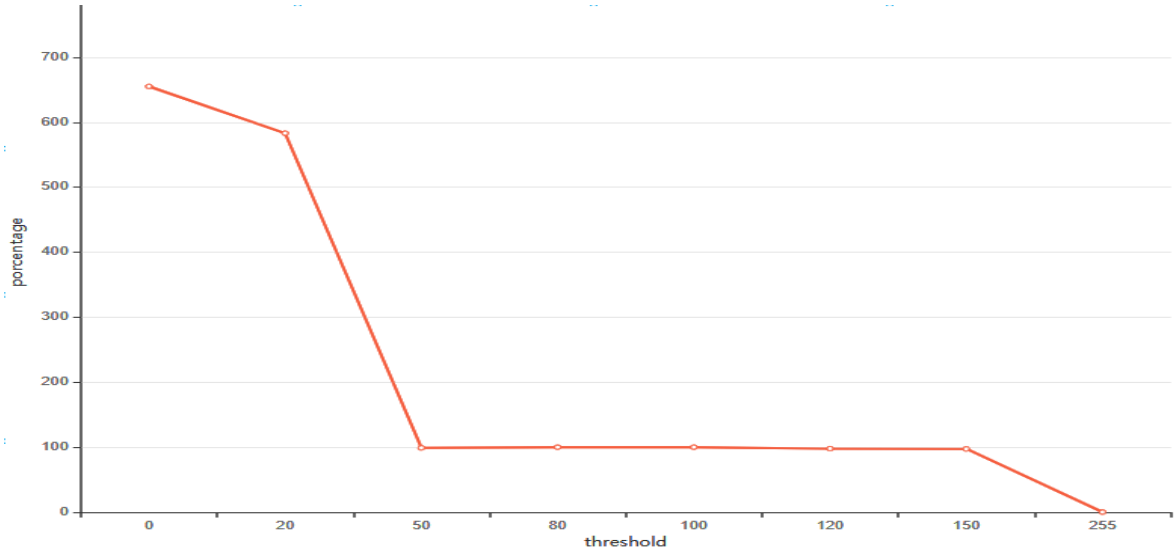


Figure 5.4 : graph of the percentage of detection of the connected components (exemple 2)

-Results of classification

In this exemple we obtained the precision of 100% for all thesymbols except for the vertical rectangle, vertical diode and the symbol 2 where the value of the precison is 99%, 76% and 50% respectively. For the accuracy we obtained the value 95% which is good, the details are in the confusion table (table 5.9)

CCS	***symbols***	**Horizontal rectangle**	**Vertical rectangle**	**Horizontal diode**	**Vertical diode**	***Circle***	***Symbol(1)***	***Symbol(2)***	***Symbol(3)***
711	29	218	331	3	16	31	15	2	2
Symbol (4)	***Symbol (5)***	***Symbol (6)***	***Symbol (7)***	***Symbol (8)***	***Symbol (9)***	***Symbole (10)***	***Symbol (11)***	***Symbol (12)***	***Symbol (13)***
19	2	10	3	2	1	1	1	2	2
Symbol (14)	***Symbol (15)***	***Symbol (16)***	***Symbol (17)***	***Symbol (18)***	***Symbol (19)***	***Symbol (20)***	***Symbol (21)***	***Symbol (22)***	***Symbol (23)***
2	1	1	1	1	1	1	1	1	1
Symbol (24)	***Reject class***								
1	37								

Table 5.8 : results of classification (exemple 2)

	RH	RV	DH	DV	C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	Reject class
RH	218																													
RV		331																												2
DH			3																											
DV				16																										5
C					31																									
1						15																								
2							2																							2
3								2																						
4									18																					
5										2																				
6											10																			
7												3																		
8													2																	
9														1																
10															1															
11																1														
12																	2													
13																		2												
14																			2											
15									1											1										
16																					1									
17																						1								
18																							1							
19																								1						
20																									1					
21																										1				
22																											1			
23																												1		
24																													1	
Reject class																														+28

Table 5.9 : Confusion table (exemple 2)

-line detection results

	circle	Horizontal diode	Vertical diode	Horizontal rectangle	Vertical rectangle	Symbol (1)	Symbol (4)	Symbol (6)	Symbol (7)
True positive	1	1	1	5	2	2	3	0	1
False positive	3	0	0	4	3	0	0	2	0

Table 5.10 : line detection result (exemple 2)

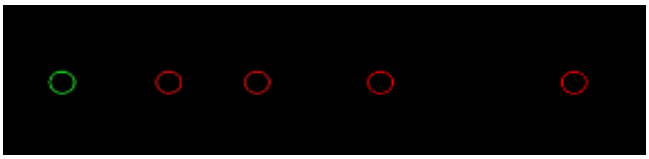


Figure 5.5 : good line detection (exemple 2)

Figure 5.6 : bad line detection (exemple 2)

GENERAL CONCLUSION AND PERSPECTIVE

6. General conclusion and perspective

Symbol detection in images is a discipline, that still in the research and development stage, used in many Applications. It allows the computer to understand the image.

In this project, we tried to develop a system for symbols detection in an image for an application that has as an objective to search for ordered series of patterns.

This project is really interesting because it allows us to deepen and apply our knowledge in the field of image processing. We faced some difficulties such as : symbols positions in the image, create and adapts a threshold to determine the best distance to take in the classification of symbols, the configuration of the parameters of the relational histogram function etc...

It took us a long time to find out what allowed us to complete this project, especially in terms of identifying the lines of symbols in the image, the subject is certainly far too vast but inevitable. We are facing difficulties because of the tuning according to thresholds or configuration and the computing time.

The OpenCV library is an important element in the realization and development of our project and this is mainly due to the wealth of different functions that compose it especially in the processing of images.

Each of the studied algorithms is more efficient than the others in specific cases, due to their simplicity or precision for exemple the (connectedComponents) function. They have nevertheless a common point, it is necessary to make compromises, it is necessary to recognize the object without committing errors.

We have analyzed some very important points in shape recognition, we have put a method for image segmentation on a black background, objects in white, it is a binarization step. The pattern recognition methods proposed by GFD are morphological study methods, but they require a lot of computing time in the case of complex geometric shapes but in our case it gave the best results hence the use of the GFD, regional segmentation method allowed us to obtain adequate related components. Furthermore the line (series of pattern) detection methode proposed by the relational histogram are efficient however the propagation algorithm that we used take a lot of time in term of computing but it gives acceptable results therefore we decided to work with it.

This experience has been encouraging to draw our attention to this vast domain because of its importance, with the possibility of improving its algorithms, which have gained valuable experience and knowledge. We can therefore offer some perspective improve the system :

- They to make the threholds configuration automatic,
- reduce spatial and temporal complexity,
- involve diffrent spatial relationsof the relational histogram

GLOSSARY

7. Glossary

FD	Fourier Descriptor
GFD	Geniric Fourier Descriptor
FT	Fouier Transform
CCS	Connected components
CC	Connected component
Px	pixel
SR	similarity report

REFERENCES

8. References

[Benoit Huet, Edwin R. Hancock 1999] ‘’Line Pattern Retrieval Using Relational Histograms’’ http://www.eurecom.fr/~huet/papers/pami99.pdf

[Dengsheng Zhang 2002] "Shape-based image retrieval using generic Fourier descriptor" https://www.sciencedirect.com/science/article/abs/pii/S092359650200084X?via%3Dihub

[D. S. Zhang and G. Lu 2001] "Content-Based Shape Retrieval Using Different Shape Descriptors: A Comparative Study". Dans Proc. Conférence internationale IEEE sur le multimédia et les expositions (ICME2001), Tokyo, Japon, pp.317-320.

[Eric Persoon and King-sun Fu. 1977] "Shape Discrimination Using Fourier Descriptors". IEEE Trans. On Systems, Man and Cybernetics, Vol.SMC-7(3):170-179.

[Hannu Kauppinen, Tapio Seppanen and Matti Pietikainen. 1995] "An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification". IEEE Trans. PAMI-17(2):201-207.

[4] [J.P. Cocquerez et S. Philipp 1995], Analyse d’images : filtrage et segmentation Ed.

Masson.

[61][Palvides .T. 1996] "Algorithms for Graphic and image processing", Rockville, MD: Computer science press.

[P. J. van Otterloo. 1991] "A contour-Oriented Approach to Shape Analysis". Prentice Hall International (UK) Ltd, pp90-108.

[1] https://fr.wikipedia.org/wiki/Distance_de_Manhattan

[2] https://fr.wikipedia.org/wiki/C%2B%2B

[3] https://fr.wikipedia.org/wki/OpenCV

ANNEXES

9. Annexes

9.1 Evalutaion

9.1.1 Exemple 3

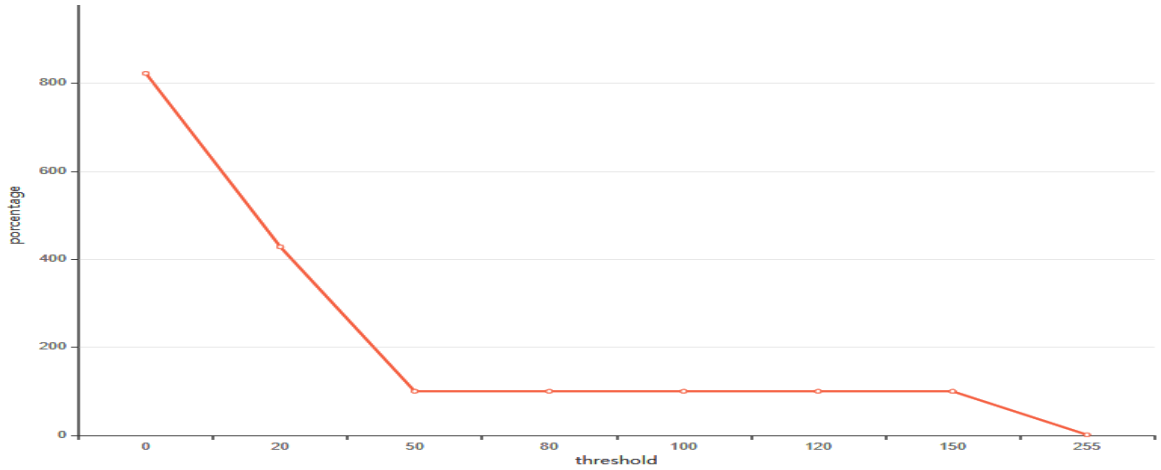
-ground Truth

CCS	symbols	background	Plane	Circle	Semi circle	Double arrow	Arrow	Rectangle 3D	Horizontal rectangle	Vertical rectangle
168	15	1	1	39	2	1	1	1	52	22
Empty rectangle	*Symbol (1)*	*Symbol (2)*	*Symbol (3)*	*Symbol (4)*	*Symbol (5)*	*Symbol (6)*				
36	4	2	2	2	1	1				

-Binarisation threshold

Binarisation threshold = 100 ; classification threhold = 17px.

threshold	0	20	50	80	100	120	150	255
number of CCs detected	489	255	168	**168**	**168**	**168**	**168**	2



-Results of classification

CCS	*symbols*	Plane	Circle	Semi circle	Double arrow	*Arrow*	*Rectangle 3D*	Horizontal rectangle	*Vertical rectangle*	*Empty rectangle*
168	15	1	32	1	1	1	1	52	22	36
Symbol (1)	***Symbol (2)***	***Symbol (3)***	***Symbol (4)***	***Symbol (5)***	***Symbol (6)***	***Reject class***				
4	2	1	1	1	1	10				

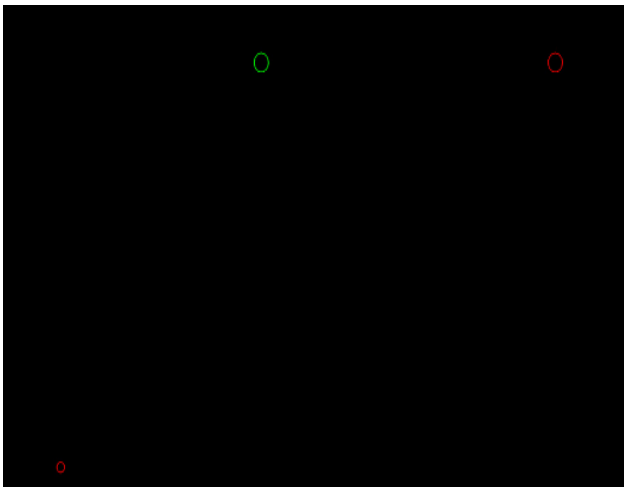
	P	C	S C	D A	A	Rec 3d	R h	R V	R E	1	2	3	4	5	6	Reject Class
P	1															
C		32														7
S C			1													1
D A				1												
A					1											
Rec 3d						1										
R h							52									
Rv								22								
RE									36							
1										4						
2											2					
3												1				1
4													1			1
5														1		
6															1	

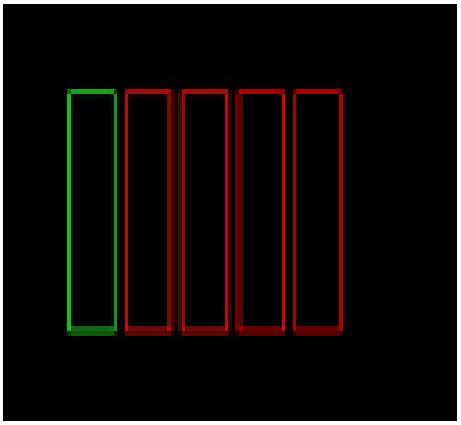
Precision = 100% except for the circle, semi circle, symbol 3 and symbol 4 where the precision was 82%, 50%, 50% and 50% respectively.

accuracy = 94%.

-line detection results

	circle	Horizontal rectangle	Vertical rectangle	Empty rectangle	Symbol (1)
True positive	0	1	1	3	0
False positive	6	2	2	0	1





9.1.2 Exemple 4

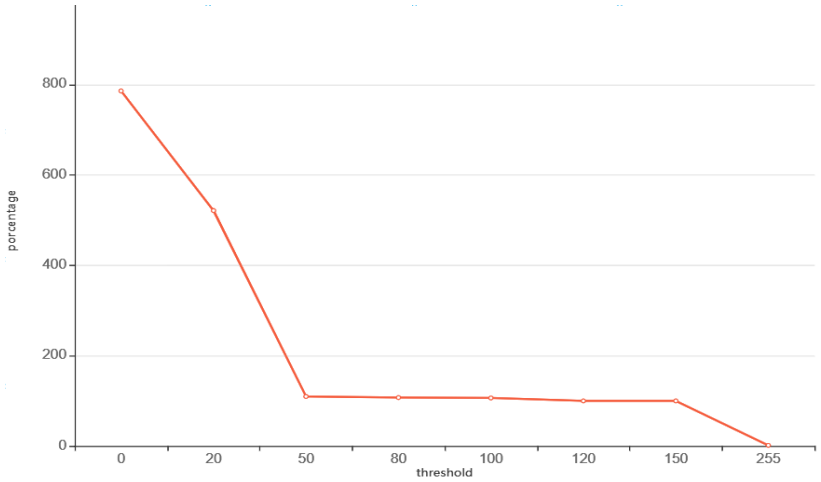
-ground Truth

CCS	symbols	background	Horizontal rectangle	Vertical rectangle	Square	Circle	Symbol(1)	Symbol(2)	Symbol(3)
122	16	1	7	10	80	6	2	2	1
Symbol (4)	*Symbol (5)*	*Symbol (6)*	*Symbol (7)*	*Symbol (8)*	*Symbol (9)*	*Symbol (10)*	*Symbole (11)*	*Symbol (12)*	
2	1	1	1	1	1	1	1	4	

-Binarisation threshold

Binarisation threhold = 150; classification threshold = .10px.

threshold	0	20	50	80	100	120	150	255
number of CCs detected	959	636	134	131	130	**122**	**122**	2



-Results of classification

CCS	symbols	Horizontal rectangle	Vertical rectangle	Square	Circle	Symbol(1)	Symbol(2)	Symbol(3)	Symbol(4)
122	16	5	10	80	6	2	2	1	2
Symbol (5)	**Symbol (6)**	**Symbol (7)**	**Symbol (8)**	**Symbol (9)**	**Symbol (10)**	**Symbole (11)**	**Symbol (12)**	**Reject class**	
1	1	1	1	1	1	1	3	3	

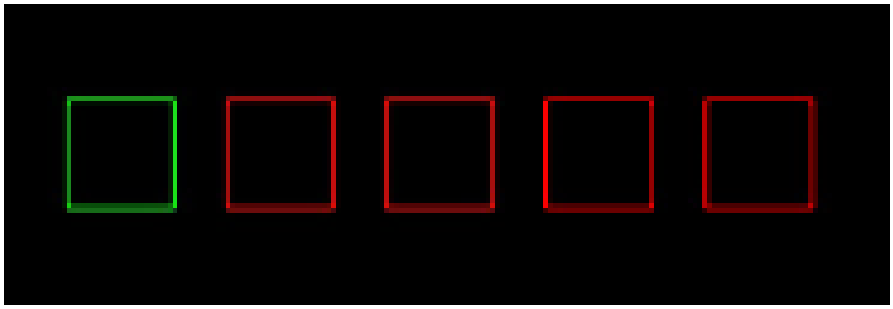
	RH	RV	CA	CE	1	2	3	4	5	6	7	8	9	10	11	12	Reject class
RH	5																2
RV		10															
CA			80														
CE				6													
1					2												
2						2											
3							1										
4								2									
5									1								
6										1							
7											1						
8												1					
9													1				
10														1			
11															1		
12																3	1

Precision = 100% except for the horizontal rectangle and the symbol 12 where the precision is 70% and 75% respectively.

accuracy = 98%.

-line detection results

	square	**circle**	**Horizontal rectangle**	**Vertical rectangle**	**Symbol (12)**
true positive	3	1	1	1	0
False positive	1	0	0	0	1



9.1.3 Exemple 5

-ground Truth

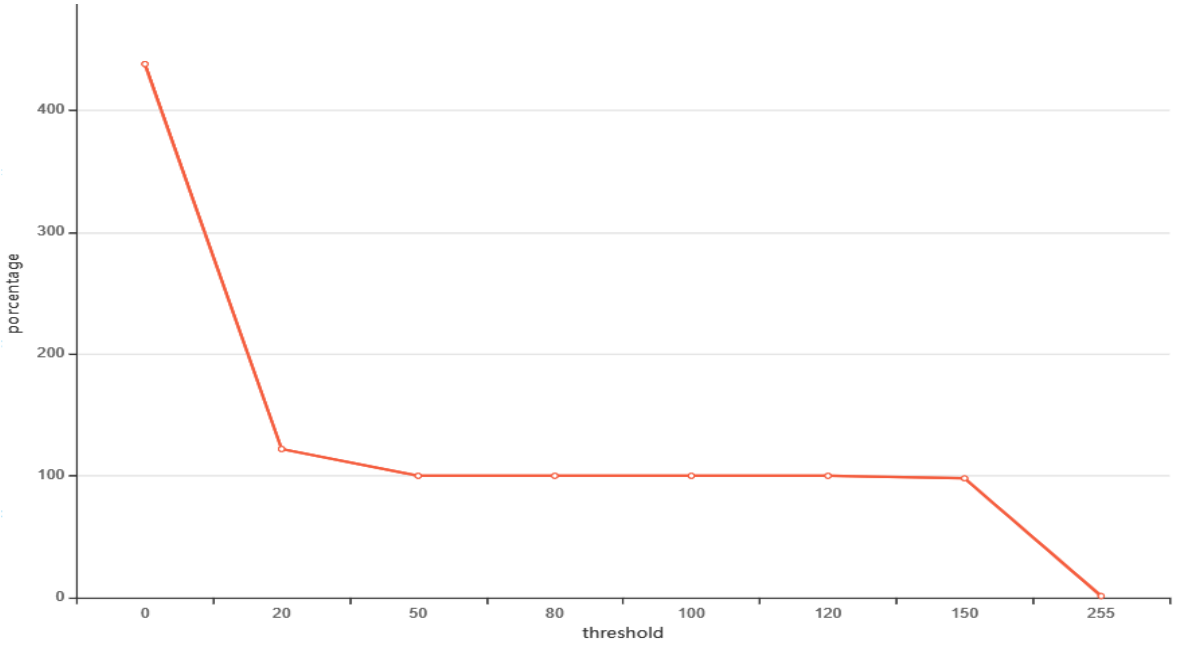
CCS	*symbols*	*background*	Horizontal rectangle	Vertical rectangle	*Symbol(1)*	*Symbol(2)*	*Symbol(3)*	*Symbol(4)*	*Symbol(5)*
149	13	1	59	71	4	4	2	1	1

Symbol (6)	*Symbol (7)*	*Symbol (8)*	*Symbol (9)*	*Symbol (10)*	*Symbol (11)*
1	1	1	1	1	1

-Binarisation threshold

Binarisation threhold = 120; classification threhold = 20px.

threshold	0	20	50	80	100	120	150	255
number of CCs detected	653	182	**149**	**149**	**149**	**149**	146	2



-Results of classification

CCS	symbols	Horizontal rectangle	Vertical rectangle	Symbol(1)	Symbol(2)	Symbol(3)	Symbol(4)	Symbol(5)	Symbol (6)
149	13	60	70	4	4	2	1	1	1
Symbol (7)	**Symbol (8)**	**Symbol (9)**	**Symbol (10)**	**Symbol (11)**	**Reject class**				
1	1	1	1	1	1				

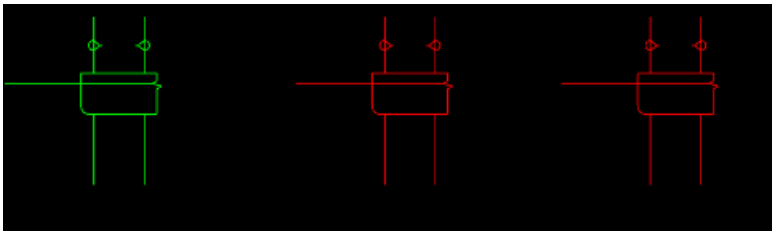
	RH	RV	1	2	3	4	5	6	7	8	9	10	11	Reject class
RH	59													
RV	1	70												
1			4											
2				4										
3					2									
4						1								
5							1							
6								1						
7									1					
8										1				
9											1			
10												1		
11													1	

Precision = 100%.

accuracy = 99%.

-line detection results

	Horizontal rectangle	Vertical rectangle	Symbol (1)	Symbol (2)
True positive	2	6	1	1
False positive	5	5	0	0



Files (.txt) generated by the algorithm. It contain the classification details (the distances) and the detailed procedure for detecting the lines for the first exemple.

```
search 4 nearest neighbors of Symbole 1_1
histogram calculation Symbole 1_1&Symbole 1_2
save to Symbole 1_1&Symbole 1_2.txt
histogram calculationSymbole 1_3&Symbole 1_1
save to Symbole 1_3&Symbole 1_1.txt
similaritey ratio = 0.990175>0.500000
save to Symbole 1[1_2_3].jpg
transform to Symbole 1[1_2_3].pgm
search 4 nearest neighbors of Symbole 1_2
search 4 nearest neighbors of Symbole 1_3
propagate the nearest neighbors of A or B
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_4
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_4.txt
save to Symbole 1_5&Symbole 1_Symbole 1[1_2_3].txt
0.228878<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_1
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_1.txt
save to Symbole 1_3&Symbole 1_Symbole 1[1_2_3].txt
0.230476<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_4
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_4.txt
save to Symbole 1_5&Symbole 1_Symbole 1[1_2_3].txt
0.228878<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_1
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_1.txt
save to Symbole 1_3&Symbole 1_Symbole 1[1_2_3].txt
0.230476<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_4
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_4.txt
save to Symbole 1_5&Symbole 1_Symbole 1[1_2_3].txt
0.228878<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_1
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_1.txt
save to Symbole 1_3&Symbole 1_Symbole 1[1_2_3].txt
0.230476<0.500000 => Stop
histogram calculation Symbole 1_Symbole 1[1_2_3]&Symbole 1_4
save to Symbole 1_Symbole 1[1_2_3]&Symbole 1_4.txt
save to Symbole 1_5&Symbole 1_Symbole 1[1_2_3].txt
0.228878<0.500000 => Stop
look for other lines
look for other lines
look for other lines
search 4 nearest neighbors of Symbole 1_4
no neighbors or already taken => Stop
look for other lines
search 4 nearest neighbors of Symbole 1_5
no neighbors or already taken => Stop
look for other lines
search 4 nearest neighbors of Symbole 1_6
```

```
number classified components as diode :4
number classified components as fleche :1
number classified components as Rectangle Empty trou bottom :9
number classified components as Rectangle Empty trou top :11
number classified components as Rectangle H :13
number classified components as Rectangle V :13
number classified components as Symbole 1 :15
number classified components as Symbole 10 :1
number classified components as Symbole 11 :1
number classified components as Symbole 12 :1
number classified components as Symbole 13 :1
number classified components as Symbole 14 :1
number classified components as Symbole 15 :1
number classified components as Symbole 16 :1
number classified components as Symbole 17 :0
number classified components as Symbole 18 :1
number classified components as Symbole 19 :1
number classified components as Symbole 2 :0
number classified components as Symbole 20 :1
number classified components as Symbole 21 :1
number classified components as Symbole 22 :1
number classified components as Symbole 23 :1
number classified components as Symbole 24 :1
number classified components as Symbole 25 :1
number classified components as Symbole 26 :1
number classified components as Symbole 27 :1
number classified components as Symbole 28 :2
number classified components as Symbole 3 :2
number classified components as Symbole 4 :2
number classified components as Symbole 5 :2
number classified components as Symbole 6 :2
number classified components as Symbole 7 :2
number classified components as Symbole 8 :1
number classified components as Symbole 9 :1
nombre composants rejeter :1
```

```
nombre CCs detected (background including): 99
Component_1 =>diode 5665.12 | fleche 5740.11 | Rectangle Empty trou bottom 5609.36 | Rectangle Empty trou top 5654.45 | Rectangle H 5654.89 | Rectangle V 8.1826e-06 | Symbole 1 4984.71 | Symbole 10 5739.76 | Symbole 11 5741.09 | Symbole 12 5705.5 | Symbole 13 5702.5 |
Symbole 14 5698.27 | Symbole 15 5691.91 | Symbole 16 5586.5 | Symbole 17 5735.16 | Symbole 18 5740.89 | Symbole 19 5661.96 | Symbole 2 5702.86 | Symbole 20 5689.48 | Symbole 21 5736.76 | Symbole 22 5738.96 | Symbole 23 5719.16 | Symbole 24 5736.75 | Symbole 25 5703.39 |
Symbole 26 5709.6 | Symbole 27 5695.81 | Symbole 28 5706.26 | Symbole 3 5520.22 | Symbole 4 5728.59 | Symbole 5 5730.13 | Symbole 6 5523.3 | Symbole 7 5420.13 | Symbole 8 5732.34 | Symbole 9 5736.28 |  => 8.1826e-06 => Rectangle V
Component_2 =>diode 5665.12 | fleche 5740.11 | Rectangle Empty trou bottom 5609.36 | Rectangle Empty trou top 5654.45 | Rectangle H 5654.89 | Rectangle V 8.1826e-06 | Symbole 1 4984.71 | Symbole 10 5739.76 | Symbole 11 5741.09 | Symbole 12 5705.5 | Symbole 13 5702.5 |
Symbole 14 5698.27 | Symbole 15 5691.91 | Symbole 16 5586.5 | Symbole 17 5735.16 | Symbole 18 5740.89 | Symbole 19 5661.96 | Symbole 2 5702.86 | Symbole 20 5689.48 | Symbole 21 5736.76 | Symbole 22 5738.96 | Symbole 23 5719.16 | Symbole 24 5736.75 | Symbole 25 5703.39 |
Symbole 26 5709.6 | Symbole 27 5695.81 | Symbole 28 5706.26 | Symbole 3 5520.22 | Symbole 4 5728.59 | Symbole 5 5730.13 | Symbole 6 5523.3 | Symbole 7 5420.13 | Symbole 8 5732.34 | Symbole 9 5736.28 |  => 8.1826e-06 => Rectangle V
Component_3 =>diode 5665.12 | fleche 5740.11 | Rectangle Empty trou bottom 5609.36 | Rectangle Empty trou top 5654.45 | Rectangle H 5654.89 | Rectangle V 8.1826e-06 | Symbole 1 4984.71 | Symbole 10 5739.76 | Symbole 11 5741.09 | Symbole 12 5705.5 | Symbole 13 5702.5 |
Symbole 14 5698.27 | Symbole 15 5691.91 | Symbole 16 5586.5 | Symbole 17 5735.16 | Symbole 18 5740.89 | Symbole 19 5661.96 | Symbole 2 5702.86 | Symbole 20 5689.48 | Symbole 21 5736.76 | Symbole 22 5738.96 | Symbole 23 5719.16 | Symbole 24 5736.75 | Symbole 25 5703.39 |
Symbole 26 5709.6 | Symbole 27 5695.81 | Symbole 28 5706.26 | Symbole 3 5520.22 | Symbole 4 5728.59 | Symbole 5 5730.13 | Symbole 6 5523.3 | Symbole 7 5420.13 | Symbole 8 5732.34 | Symbole 9 5736.28 |  => 8.1826e-06 => Rectangle V
Component_4 =>diode 5665.12 | fleche 5740.11 | Rectangle Empty trou bottom 5609.36 | Rectangle Empty trou top 5654.45 | Rectangle H 5654.89 | Rectangle V 8.1826e-06 | Symbole 1 4984.71 | Symbole 10 5739.76 | Symbole 11 5741.09 | Symbole 12 5705.5 | Symbole 13 5702.5 |
Symbole 14 5698.27 | Symbole 15 5691.91 | Symbole 16 5586.5 | Symbole 17 5735.16 | Symbole 18 5740.89 | Symbole 19 5661.96 | Symbole 2 5702.86 | Symbole 20 5689.48 | Symbole 21 5736.76 | Symbole 22 5738.96 | Symbole 23 5719.16 | Symbole 24 5736.75 | Symbole 25 5703.39 |
Symbole 26 5709.6 | Symbole 27 5695.81 | Symbole 28 5706.26 | Symbole 3 5520.22 | Symbole 4 5728.59 | Symbole 5 5730.13 | Symbole 6 5523.3 | Symbole 7 5420.13 | Symbole 8 5732.34 | Symbole 9 5736.28 |  => 8.1826e-06 => Rectangle V
Component_5 =>diode 5665.12 | fleche 5740.11 | Rectangle Empty trou bottom 5609.36 | Rectangle Empty trou top 5654.45 | Rectangle H 5654.89 | Rectangle V 8.1826e-06 | Symbole 1 4984.71 | Symbole 10 5739.76 | Symbole 11 5741.09 | Symbole 12 5705.5 | Symbole 13 5702.5 |
Symbole 14 5698.27 | Symbole 15 5691.91 | Symbole 16 5586.5 | Symbole 17 5735.16 | Symbole 18 5740.89 | Symbole 19 5661.96 | Symbole 2 5702.86 | Symbole 20 5689.48 | Symbole 21 5736.76 | Symbole 22 5738.96 | Symbole 23 5719.16 | Symbole 24 5736.75 | Symbole 25 5703.39 |
Symbole 26 5709.6 | Symbole 27 5695.81 | Symbole 28 5706.26 | Symbole 3 5520.22 | Symbole 4 5728.59 | Symbole 5 5730.13 | Symbole 6 5523.3 | Symbole 7 5420.13 | Symbole 8 5732.34 | Symbole 9 5736.28 |  => 8.1826e-06 => Rectangle V
```

For more informations :

Github link : https://github.com/arezkibouzid/TER