Team Number:	apmcm2104264
Problem Chosen:	A

2021 APMCM summary sheet

In today's era, people's accuracy of workpieces and parts has reached a new height, which also leads to a rise in the accuracy requirements of measuring devices. The measurement application is also slowly transitioning from manual caliper measurement application to digital image measurement application. Workpiece size is one of the most important data contents of workpiece. Different from the traditional direct measurement by manual caliper, digital image measurement needs to take pictures of the workpiece first, then extract the contour of the workpiece from the image, and cooperate with the point matrix of the calibration image to obtain the dimension data of the workpiece.

For this problem, we first need to preprocess the captured image so that the next work can be carried out better. We use two-dimensional convolution to reduce the impact of noise on contour recognition, and use graying algorithm and binarization method to better adapt the image to the next contour extraction. Next, the coordinates of contour points are obtained by using the contour extraction algorithm of binary image. Next, according to the preset calibration plate, we can convert the actual size of the workpiece.

Based on the methods mentioned above, we establish the model of image measurement, test it with some pictures, and record the test results as a table.

Keywords: Digital Image Processing 2-d convolution Contour Detection

Contents

I. Introduction

In order to indicate the origin of problems, the following background is worth mentioning.

1.1 Problem Background

In the midst of rapidly changing technology, computers have gradually begun to replace many of the jobs around us, and our human resources have been liberated to a great extent. There are many mechanical tasks where computers are much more effective than people, such as the assembly of devices, short-distance transportation of materials, and the production of parts. Due to the massive popularity of computers, the demand for engineering parts has increased dramatically, and how to make computers shine on top of engineering parts has become a very hot topic.

Can a computer obtain basic information such as the dimensions of a part from a single picture? If the measurement of a device could be achieved by computer alone with industrial accuracy, we would no longer need a lot of manpower to measure a device manually using things like vernier calipers and spiral micrometers, and they could have more time to do something else. For this reason, our group discussed and tried to create programs for computerized measurement based on image recognition to find information about edge contours and lengths from a picture.

1.2 Our work

Here, we have processed the images using the python language based opencv image processing library. We first preprocessed the obtained images in terms of noise reduction, blurring, and enhancement in order to obtain better results for image contour extraction. We built a basic framework model for image preprocessing and contour extraction, tested the images, and finally found the parameters (including many parameters in the noise reduction, blurring, enhancement, and contour extraction algorithms) with relatively good results.The overall framework of the paper is shown in Figure 1

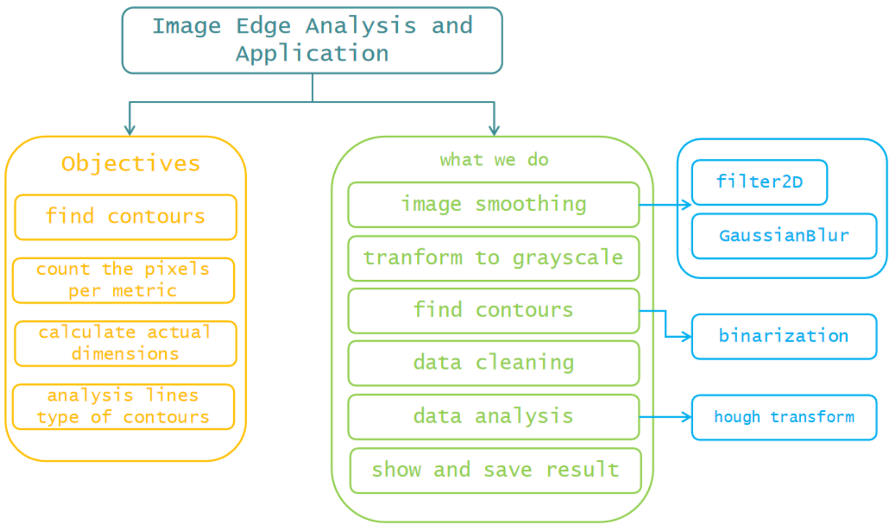


Figure 1 : The overall framework of the paper

II. The Description of the Problem

2.1 How do we approximate the whole course of ?

- For the problem of contour analysis of images, the main problem we need to solve is how to know which points are contours and thus determine the position of the whole contour. For this we need to use the method of contour extraction for binary images, which is the findContours function in opencv, a method based on the one described in the article *Computer Vision, Graphics, and Image Processing*.
- Before running the contours algorithm, we need to do some pre-processing of the image, and we used two-dimensional convolution for noise reduction and enhancement, which proved to be within our expectation. The noise-reduced image is enhanced as appropriate to avoid loss of detail due to noise reduction and can emphasize the features of the image.
- After the contouring algorithm is run, the obtained data needs to be cleaned and screened out of useless data, then the data is analyzed and processed, such as the number of contours, length, location of points, types of contour lines (straight line, circle, ellipse), and the final valid data is displayed and saved.

2.2 How do we define the optimal configuration?

The optimal configuration needs to ensure that the contour data we get can best match the real situation. In the process, we need to continuously adjust and test various parameters, then get various results, analyze the results, take the part that approximates with the local optimum, and use that part as the parameters of this final image for the whole image analysis process (noise reduction, enhancement, extraction of contours).

2.3 The local optimization and the overall optimization

- In the continuous attempts, we record the results and set an expectation, and finally find the parameter that is closest to our predefined expectation, and use this parameter as the local optimum.
- Finally, we try to find multiple local optima, and select the global optimal solution that is closest to our defined expectation, and use the parameters corresponding to this global optimal solution as the parameters for the final processing of this image. After our tests, most of the images can be processed well by a set of parameters.

III. Analysis and Modelling

3.1 image enhancement and smoothing base on 2-d convolution

Two-dimensional convolution is the mathematical method of applying convolution to a two-bit matrix, changing from a one-dimensional vector to a two-dimensional matrix. The main use of two-dimensional convolution is for feature extraction of two-dimensional information, such as image processing.

The mathematical definition of two-dimensional convolution is as follows

$$f(x,y) * g(x,y) = \int_{\tau 1=-\infty}^{\infty} f(\tau 1, \tau 2) \cdot g(x-\tau 1, y-\tau 2) d\tau 1 d\tau 2$$

Two-dimensional convolution is often encountered in image processing, and most of the image processing uses a discrete form of two-dimensional convolution: the

$$f[x,y] * g[x,y] = \sum_{n1=-\infty}^{\infty} \sum_{n2=-\infty}^{\infty} f[n1,n2] \cdot g[x-n1, y-n2]$$

The two-dimensional convolution in the actual computation process is represented by computing and translating the convolution kernel matrix on the matrix being convolved, multiplying the elements corresponding to each position in it, and then adding the results of each operation as an element of the new matrix **Examples of two-dimensional

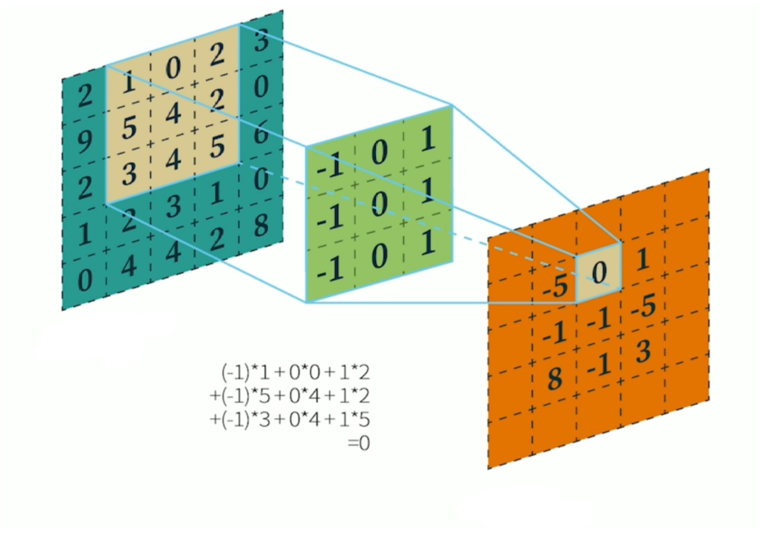


Figure 2 Two-dimensional convolution

convolution**

$$M = \begin{bmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad N = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

where the matrix M is the convolution matrix, N is the convolution kernel, and a matrix R can be obtained after the convolution operation

$$R = \begin{bmatrix} 12 & 12 & 17 \\ 10 & 17 & 19 \\ 9 & 6 & 14 \end{bmatrix}$$

It can be seen that the first element of the first row of R is derived as follows

$$12 = 3 * 0 + 3 * 1 + 2 * 2 + 0 * 2 + 0 * 2 + 1 * 0 + 3 * 0 + 1 * 1 + 2 * 2$$

So, the exact process of two-dimensional convolution is shown below, where the leftmost matrix is passed through the middle convolution kernel to obtain the convolved matrix shown on the rightmost side

The image as an input matrix can be processed by a two-dimensional convolution, using a defined filter kernel (also called an operator, convolution kernel) to perform a convolution on the image matrix, and the convolution matrix is the filtered image.

We mainly achieve two effects with the 2D convolution model: smoothing and image enhancement

Smoothing: The difference between the two effects lies in the different definitions for the convolution kernel, which can achieve different convolution effects. We use the mean filter kernel, whose specific form is shown below

$$K = \frac{1}{25}\begin{bmatrix} 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

Using this filtering kernel K, we convolve each 5x5 range in the image and then find the mean value, which can do a smoothing of the image. If it is a three-channel color image, we will convolve each channel separately in the processing, and if it is a single-channel black-and-white image, we only need to do the two-dimensional convolution introduced before.

enhancement: image enhancement of the convolution kernel K we use the Laplace operator

Laplace operator is an image neighborhood enhancement algorithm derived by second-order differentiation based on the calculation of pixel grayscale difference in the image neighborhood. Its basic idea is that when the grayscale of the center pixel in the neighborhood is lower than the average grayscale of other pixels in the neighborhood, the grayscale of this center pixel should be further reduced; when it is higher than that, the grayscale of the center pixel should be further increased, so as to achieve image sharpening. In the implementation of the algorithm, the gradient of the central pixel in the neighborhood is determined by finding the gradient in four or eight directions, and summing the gradient to determine the relationship between the grayscale of the central pixel and the grayscale of other pixels in the neighborhood, and adjusting the pixel grayscale with the result of the gradient operation. A continuous binary function f(x,y), whose Laplace operation is defined as de the original source link and this statement.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

the Laplace operator can be simplified as

$$g(i,j) = 4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(j,j-1)$$

It can also be expressed in the form of a convolution.

$$g(i,j) = \sum_{r=-k}^{k} \sum_{s=-l}^{l} f(i-r, j-s)H(r,s), i,j = 0,1,2,\cdots,N-1$$

When K=1, I=1 when H(r,s) takes the following equation, four directional template

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The result of the convolution operation of the template can be found to be 0 when the grayness of the pixels in the neighborhood is the same, positive when the grayness of the center pixel is higher than the average grayness of the other pixels in the neighborhood, and negative when the grayness of the center pixel is lower than the average grayness of the other pixels in the neighborhood. The result of the convolution operation is processed with an appropriate fading factor and added to the original center pixel to achieve sharpening of the image

3.2 image graying model

Image grayscale refers to the original color three-channel image by grayscale processing to make it into a single-channel grayscale image, that is, black and white images.[3]

- **Convert RGB to YUV and then use Y as grayscale**
  opencv provides a method, which is the one we use in the processing.
  First, the RGB values are mapped according to the relationship function between RGB and YUV color space (as follows) to find the content of Y

$$gray = Y = 0.3R + 0.59G + 0.11B$$

Where R, G and B correspond to the three components in RGB color space, respectively, and Y represents the value of Y in YUV color space. In the YUV color space, Y represents the luminance, U and V are represented in the chromatogram as horizontal and vertical coordinates, corresponding to a color, which is the expression of the YUV color space. the value chromatogram of UV is shown in the following figure

In opencv's cvtColor method, use the Y approximation equal to the grayscale value, thus achieving the grayscale transformation effect

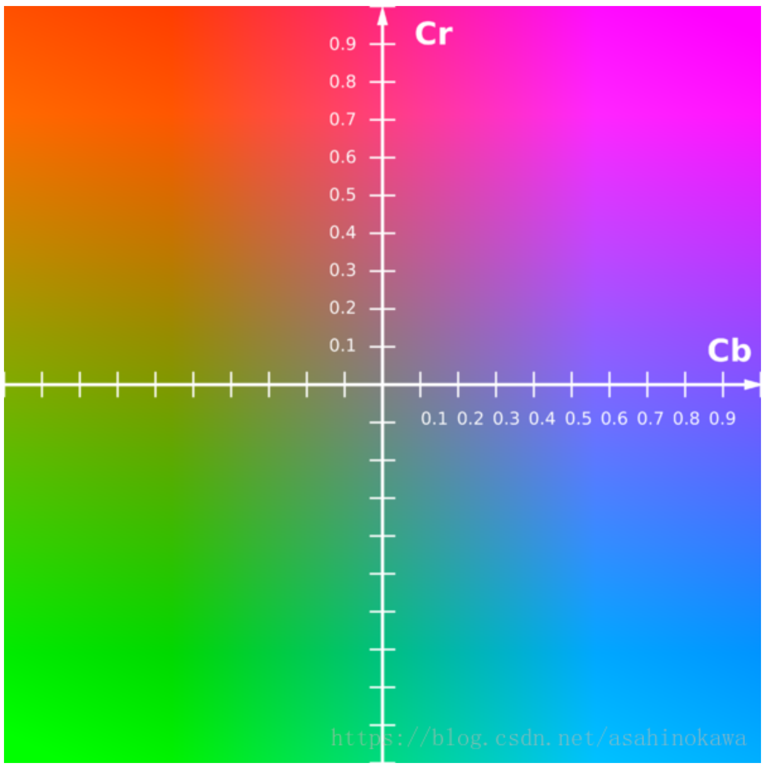


Figure 3 YUV color space chromatography

- **Maximum value graying**

  Maximum grayscale means that the largest of the three components of the RGB color space is used as the output grayscale value, as expressed by the following equation

$$gray = max([B, G, R])$$

- **Average graying**

$$gray = \frac{(B + G + R)}{3}$$

- **Gamma-corrected grayscale**

  In the RGB color space, the three RGB components are not simply linearly related to the physical light power, but a power function relationship, where the exponent of the power function is called the Gamma value, which is generally 2.2, and the conversion process is called Gamma correction.

  Different people have different perceptions of light power, for example, a power of 50% gray, the actual brightness perceived by the human eye is

$$\sqrt[2.2]{0.5} * 100\% = 72.97\%$$

So that for the grayscale values in RGB, Gamma correction is needed in order to take into account a smaller storage range and a more balanced ratio of light to dark. So the three RGB components cannot simply be added directly, but must be mapped to physical optical power using a power function with an exponent of 2.2 before they can be calculated, so the formula for grayscale becomes the following form

$$Gray = \sqrt[2.2]{\frac{R^{2.2} + (1.5G)^{2.2} + (0.6B)^{2.2}}{1 + 1.5^{2.2} + 0.6^{2.2}}}$$

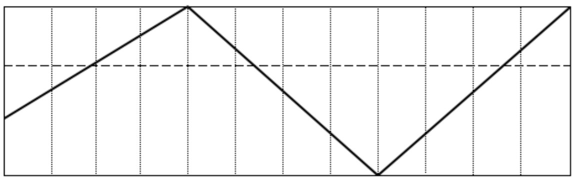
As you can see from the formula, the RGB color values here are first converted to physical optical power, and this process is called Gamma correction, which is Gamma corrected grayscale.

3.3 Image binaryzation

Binarization of an image means converting a grayscale map into a binary map according to a set criterion, which has only two colors, black (0) and white (255)

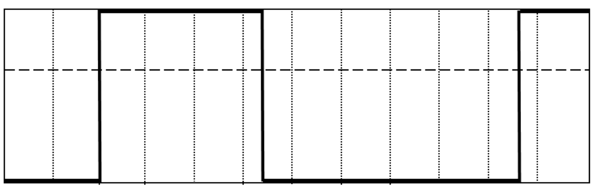
Binarization of an image can be approximated as a classifier. The process of binarization is to traverse the entire grayscale map, mapping each pixel of the grayscale map to a discrete interval [0,1]. If the grayscale exceeds a set threshold threshold, it becomes 255, and vice versa, it becomes 0

- The threshold method in opencv supports several modes of binarization, the following modes are introduced



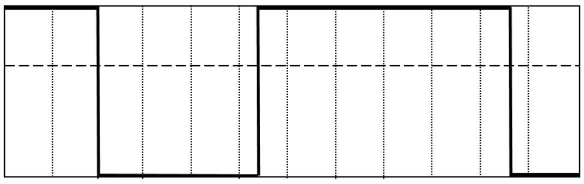
- **THRESH_BINARY**

  The part larger than the threshold is set to 255 and the part smaller is set to 0



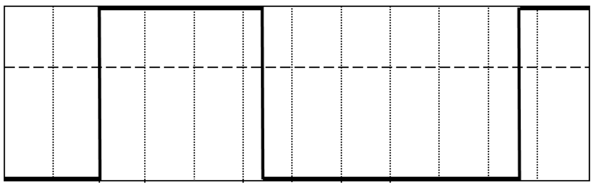
- **THRESH_BINARY_INV**

  Larger than the prefabricated part is set to 0, smaller than the part is set to 255



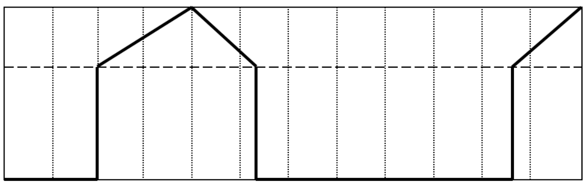
- **THRESH_TRUNC**

  Larger than the prefabricated part is set to the set threshold, smaller than the part unchanged



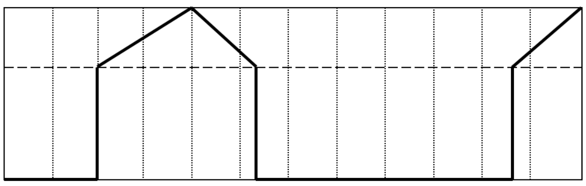
- **THRESH_TOZERO**

  The part greater than the threshold is unchanged: the part less than is set to 0



- **THRESH_TOZERO_INV**

  The part larger than the threshold is set to 0, and the part smaller is unchanged



3.4 Contour detection algorithm

The contour extraction algorithm in opencv is an implementation of the contour extraction algorithm described in the paper computer vision, graphics, and image processing[1]. After learning this paper, our group briefly summarized the algorithm and applied it to our program.

3.4.1 *Terms,Definitions and Symbols*

frame: The edge of an image that forms the frame of this image. if an image is 640*480, the frame of this image is the 0th row of pixels, the 479th row of pixels, the 0th column of pixels, and the 639th column of pixels.

0-pixel (0 pixels), 1-pixel (1 pixel): Pixels with grayscale values of 0 and 1 (0 and 255 in binarization) are called 0pixel and 1-pixel, respectively.

(i, j): Denotes the element of the i-th row and j-th column of the image.

f_{ij}: Denotes the grayscale value of pixel point (i, j).

0-component, 1-component: A connected field consisting of 0-pixel or 1-pixel. If a 0-component contains a **frame**, the connected field is called a **background**, otherwise it is called a **hole**.

4 (8) connected scenes: 1 pixel is 4 (8) connected, 0 pixel is 8 (4) connected.

border point: In a 4(8) connected scene, if a 1 pixel (i, j) has 0 pixels (p, q) present in its 4(8) connected domain, the 1 pixel (i, j) is said to be a **border point**.

surroundness among connected components: In a binary image, if there are 2 among connected components named S1 and S2 and any pixel point in S1 has a pixel point of S2 on the path from any direction (4 directions) to the frame, we called S2 **surround** S1. If S2 surrounds S1 and there is a **border point** between S2 and S1, then S2 **directly surrounds** S1.

outer border and hole border: Assume the existing 1-connected domain S1,0-connected domain S2. if S2 **directly surrounds** S1, the boundary between S2 and S1 is called **the outer boundary**; if S1 directly surrounds S2, the edge between S2 and S1 is called **the hole boundary**.

parent border: Suppose existing 1 connects domain S1 and S3,0 connects domain S2, S2 directly surrounds S1, S3 directly surrounds S2, the boundary between S1 and S2 is B1, the boundary between S2 and S3 is B2, then B2 is the parent boundary of B1. If S2 is **background**, then the parent boundary of B1 is **frame**.

surroundness among borders: Fix two boundaries B_0 and B_n, if there exists a sequence of boundaries $\boldsymbol{B_0, B_1, ..., B_n}$, where $\boldsymbol{B_k}$ is the parent boundary of $\boldsymbol{B_{k-1}}$, then we say that $\boldsymbol{B_n}$ **surrounds** $\boldsymbol{B_0}$.

By using the above definition, the boundaries and connected domains in a picture can be composed of the following topological relations(Figure 4)

RasterScan: Means from left to right, from top to bottom, first after scanning a line, and then move to the start of the next line to continue line by line scanning.

starting point: The boundary start points are divided into **outer boundary start points** (Figure 5(a)) and **hole boundary start points** (Figure 5(b)). If the point (i, j) satisfies both (a) and (b), then consider (i, j) as **the outer boundary start point**.

NBD: A boundary can be obtained from the boundary start point (i, j) with the bound-

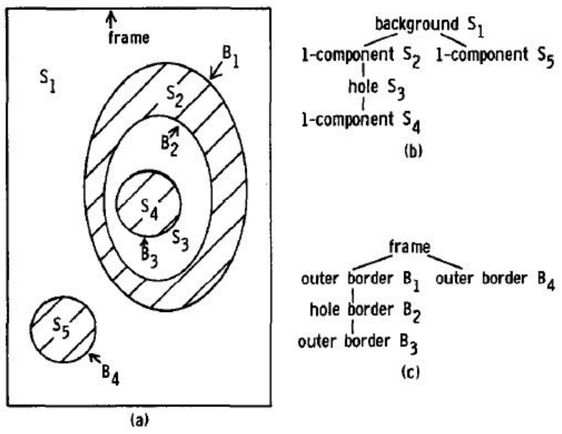


Figure 4 Surroundness among connected components (b) and among borders (c)

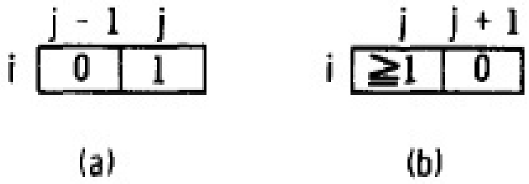


Figure 5 The conditions of the border following starting point (i, j) for an outer border (a) and among border (b)

ary tracking algorithm, and a new unique number is assigned to each newly found boundary **B**. **NBD** denotes the number of the currently tracked boundary.

LNBD: During the raster scan, we also keep the number of the most recently encountered (previous) boundary **B'**, noted as **LNBD**.

3.4.2 *Contour extraction algorithm*

Suppose an image $F = f_{ij}$ is input and the initial NBD is set to 1, that is, the **frame** of F is considered as the first boundary. Use raster scan to scan the image F line by line, and when the gray value of a pixel point (i, j) satisfies $f_{ij} \neq 0$, perform the steps listed below. When the scan reaches the end of a line of the image, the **LNBD** is reset to 1.

(1) Select one of the following cases

(a)If $f_{ij} = 1$ and $f_{i,j-1} = 0$ which be described by (Figure 5(a)), then (i, j) is **the outer boundary start point** and **NBD+=1**, $(i_2, j_2) \leftarrow (i, j-1)$

(b)If $f_{ij} \geq 1$ and $f_{i,j+1} = 0$ which be described by (Figure 5(b)), then (i, j) is **the hole boundary start point** and **NBD+=1**, $(i_2, j_2) \leftarrow (i, j+1)$. If $f_{ij} > 1$, then $LNBD \leftarrow f_{ij}$

(c)If neither case (a) nor case (b) is satisfied, proceed to step (4)

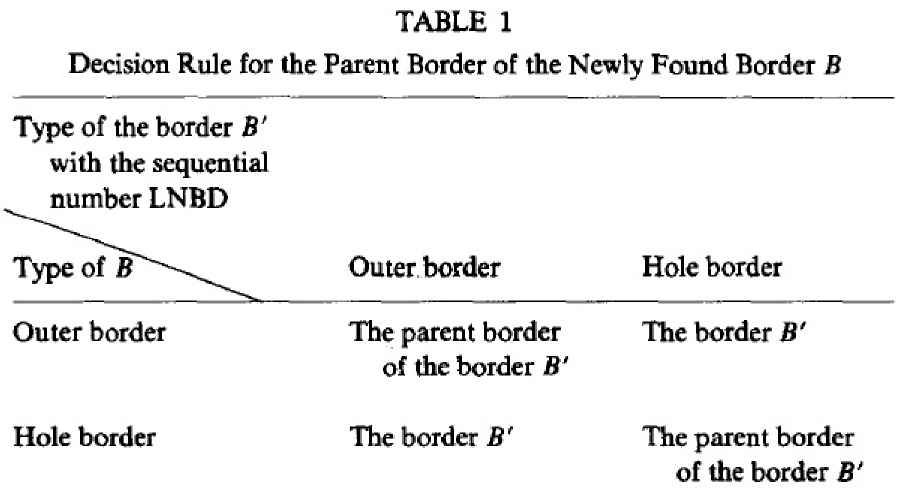


TABLE 1
Decision Rule for the Parent Border of the Newly Found Border B

Type of B \ Type of the border B' with the sequential number LNBD	Outer border	Hole border
Outer border	The parent border of the border B'	The border B'
Hole border	The border B'	The parent border of the border B'

Figure 6 Descision Rule for the Parent Border of the Newly Found Border B

(2) Based on the type of the previous boundary **B’** and the current newly encountered boundary **B**, the parent boundary of the current boundary **B** can be derived from the following table.(Figure 6)

(3) Starting from the boundary starting point (i, j), track the boundary according to the following methods ((3.1) to (3.5)).

(3.1) With (i, j) as the center, (i_2, j_2) as the starting point, look clockwise for (i, j) 4 (8) domains with non-zero pixel points. If a non-zero pixel point is found, make (i_1, j_1) the first non-zero pixel point clockwise, otherwise make $f_{ij} = -NBD$ and proceed directly to step (4).

(3.2) $(i2, j2) \leftarrow (i1, j1)$, $(i3, j3) \leftarrow (i, j)$

(3.3) With (i_3, j_3) as the center, in a counterclockwise direction, to find (i_2, j_2) the next point as the next seven points (i_3, j_3) whether there are non-zero pixel points in the 4 (8) neighborhood, making (i_4, j_4) the first non-zero pixel point in the counterclockwise direction.

(3.4) Take the following steps:

(a) If $(i_3, j_3 + 1)$ is a pixel that has been checked in step (3.3) and is a 0 pixel, then $f_{i_3,J_3} \leftarrow -NBD$.

(b) If $(i_3, i_3 + 1)$ is not the 0 pixel already checked in step (3.3), and $f_{i_3,j_3} = 1$, then $f_{i_3,j_3} \leftarrow NBD$.

(c) If neither case (a) nor case (b) is satisfied, then not to change the value of f_{i_3,j_3}.

(3.5) If $(i_4, j_4) = (i, j)$ and $(i_3, j_3) = (i_1, j_1)$, that means, it returns to the starting point of the boundary, proceed to step (4), otherwise make $(i_2, j_2) \leftarrow (i_3, j_3)$,

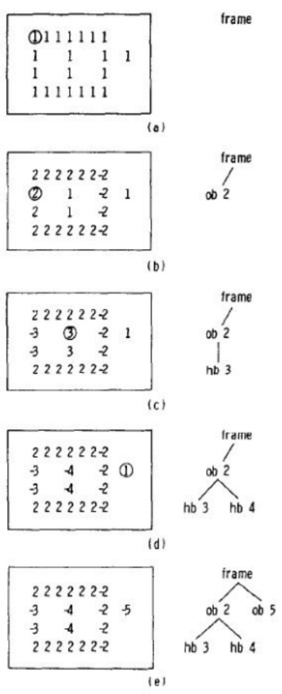


Figure 7 An illustration of the process of Algorithm

$(i_3, j_3) \leftarrow (i_4, j_4)$, and then proceed to step (3.3).

(4) If $F_{ij} \neq 1$, $LNBD \leftarrow \mid f_{ij} \mid$, continue raster scanning from point $(i, j+1)$, and end scanning when scanning to the bottom right corner of the picture.

The whole algorithm can be represented by use case diagram (Figure 7).

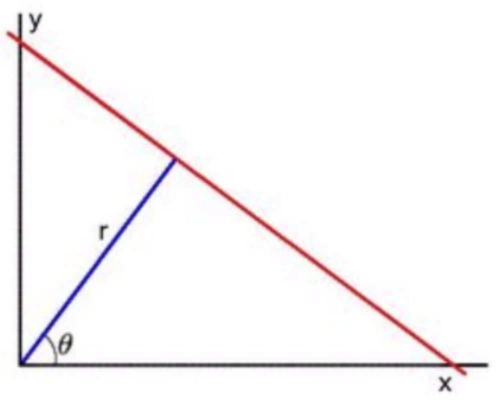
To sum up, the contour extraction algorithm mainly detects a contour and then tracks it to detect all the contours of the whole picture. The whole step is to scan each pixel of the whole picture circularly, judge and deal with various situations in the scanning, so as to realize the algorithm. The findcontours method in opencv is only an implementation of this algorithm.

3.4.3 *Hough transform*

Since the value range of k and b is $(-\infty, +\infty)$ for a straight line, an equivalent transformation of K and B is required.

In the figure (Figure 8), r is the shortest distance from the coordinate origin to a straight line, and θ is the angle between the vertical line from the origin to the target straight line and the x-axis. In this way, we can use θ and r to replace k and b in the original oblique section.

$$k = -\frac{\cos\theta}{\sin\theta} b = \frac{r}{\sin\theta}$$



$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

$$r = x\cos\theta + y\sin\theta$$

Figure 8 Hough tranform

By simplifying and removing the denominator of the converted oblique section, we can get an equation r about θ

$$r = x\cos\theta + y\sin\theta$$

According to the definitions of r and θ, the value of r is $[0, +\infty)$, and the value of θ is $[0, 360)$. At this time, θ can be discretized. For example, the value range of θ is defined as discrete interval $[0, 90, 180, 270]$, and r is divided into n intervals $[0, 1], [1, 2], [2, 3) \cdots [n-1, n)$ in a fixed value. Then θ can be used as row index and r as column index to form a zero matrix. We can understand this matrix as a "ballot box".

Scan the coordinates of all points and calculate them according to the discretized θ to obtain the value of r. then, r corresponds to an interval of the divided interval, and one value of the corresponding matrix is added, that is, "one vote".

Finally, a threshold is set. The straight line composed of the index r and θ of the rows and columns in the matrix exceeding this threshold can be considered as a straight line in the graph. So as to achieve the effect of detecting a straight line.

The circle detection is also similar. Through the transformation of the equation, build a "ballot box" of the zero matrix, then traverse each point for operation, vote for the corresponding position in the "ballot box", and finally select the part whose number of votes exceeds a certain threshold.

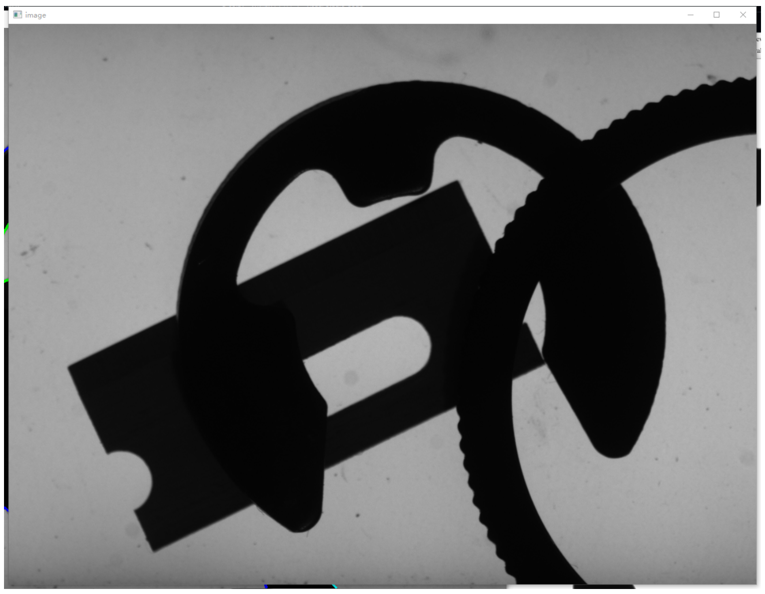


Figure 9 Original image after reading

IV. Calculation method and process

4.1 Contour extraction of binary image

We detect and draw the contours of the three images. We hope we can get the contour line that matches the contour of the real object in our photo as much as possible. In the continuous exploration of methods, we finally decided to use Python opencv as the image processing library, because it can easily process an image. Opencv supports a large number of image transformation and analysis methods, which is well in line with our expectation of problem-solving methods. We hope to get a relatively simple and clear tool for image analysis and processing.

Before the image processing, we determined our needs. We need to find the coordinates of a series of points, which are the contour points of the object photo.

In order to get the contour points, we need to read the picture first. Of course, it is necessary. Without the picture, no data will be analyzed by us. So we use the imread function to read the picture and store it in a variable. In order to see the contour search effect more intuitively in the end, we decided to draw the contour directly on the original drawing, so that we can compare it better. In order to do this, we copy the image and store it in another variable as the final result.

(Figure 9) is the original image after reading.

Before image processing, we need to blur and sharpen it. We use the two-dimensional convolution model to blur and sharpen it (section 3.1).

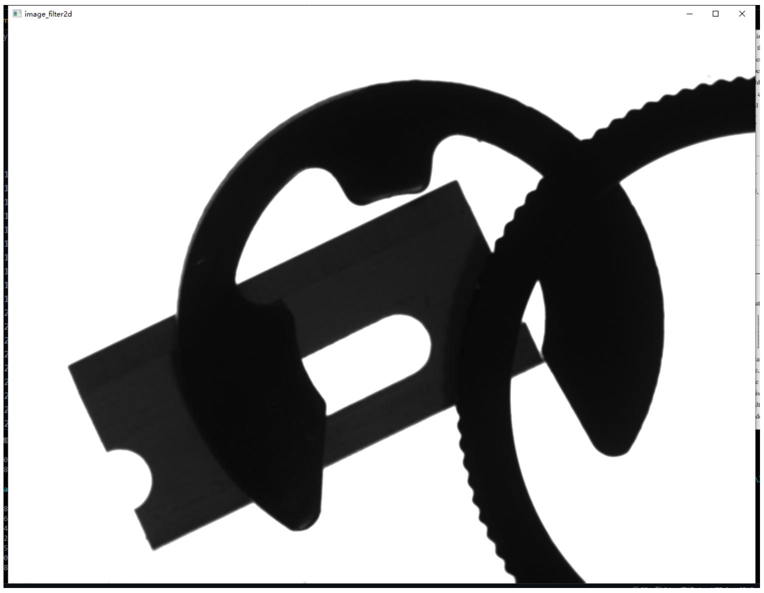


Figure 10 Image after smoothing and sharpening

In the smoothing process, we use K1 matrix as the convolution kernel of our smoothing process, which is a mean convolution kernel.

$$K_1 = \frac{1}{25}\begin{bmatrix} 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

In sharpening, we use K2 matrix as our sharpening convolution kernel, which is a Laplace operator.

$$K_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The purpose of smoothing and sharpening is to denoise the image and extract the eigenvalues, so as to better find the contour points.

(Figure 10) is the image after smoothing and sharpening.

The processed image uses the relationship between RGB and YUV color space to convert each other, and takes y (brightness) directly as the gray value for a simple image graying process (section 3.2). This is to simplify the data size, because the gray image is a single channel, and one pixel has only one value, while the RGB image is a three channel image, and each pixel has three values, representing R, G and B respectively.

(Figure 11) is the grayed image.

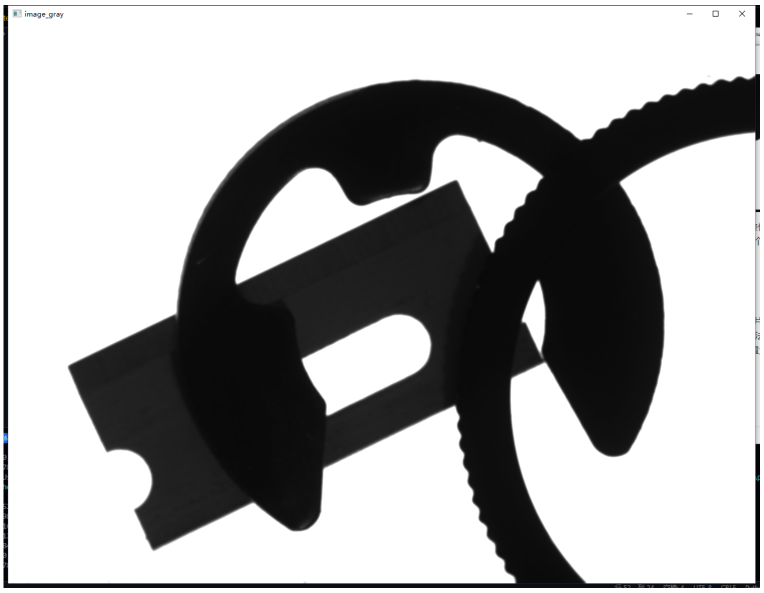


Figure 11 Grayed image

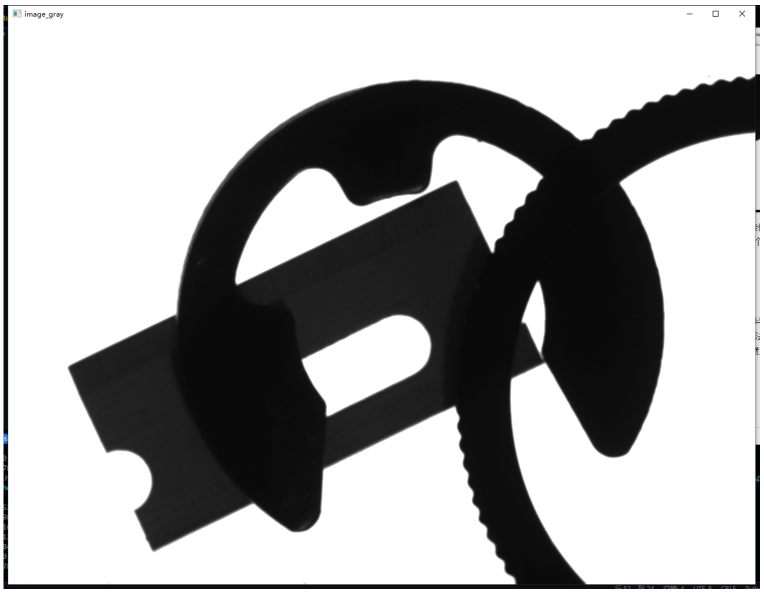


Figure 12 Binary image

For the grayed image, perform a binarization process (section 3.3). This step is to filter the image again and change each pixel of the image from a grayscale value to 0 or 1, so as to be better applied to the contour extraction algorithm of binary image. The method of binarization is to set a threshold and then define a processing mode. The mode we use is above the threshold, the pixel is set to 1, and vice versa.

(Figure 12) is the binary image.

It can be seen from the figure (Figure 12) that after binarization, the contour of the object no longer has sub-pixel contour, either 0 or 1.

So far, the image preprocessing has been completely completed. Next, only the

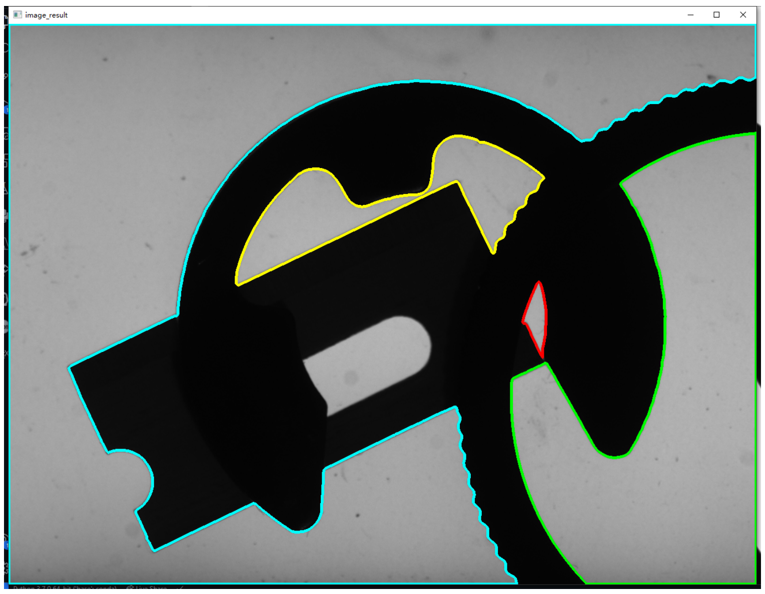


Figure 13 Preliminary results

contour extraction algorithm of binary image needs to be used for the binary image (section 3.4). The principle of the algorithm is to scan the whole picture line by line, detect itself and neighborhood of all pixels, set a variable to track the points that meet the contour point conditions (0 pixel exists in 4 (8) neighborhood of 1 pixel), and finally find all contour point information. Then draw a contour according to the contour points, and finally get the contour map (Figure 13).

It can be seen that the edge of the picture is also mistakenly regarded as contour points. This is because the algorithm assumes that the edge of the whole picture is a circle of contour, so we need to screen out the contour points at the edge of the picture.

The data cleaning process of contour points uses a very simple cycle detection. Firstly, the storage structure of contour points is obtained through debugging, and then it is transformed into a multi-dimensional array. All two-dimensional vectors representing the coordinates of contour points in the data are filtered, and the coordinates meet the edge conditions of the picture (abscissa is equal to 0 or equal to the transverse maximum of the picture, and ordinate is equal to 0 or equal to the longitudinal maximum of the picture). In order to avoid the mismatch between the index and value of the element due to the deletion of the element during the screening process, a variable called rep is used to correct the index. Finally, the contour points are drawn and output, but here we encounter a problem. When we draw the contour, we directly use the contour drawing method. The drawn contour is a closed curve, which will make the picture look like the figure (Figure 14).

This is not what we expect. In order to avoid this situation, we abandon the original

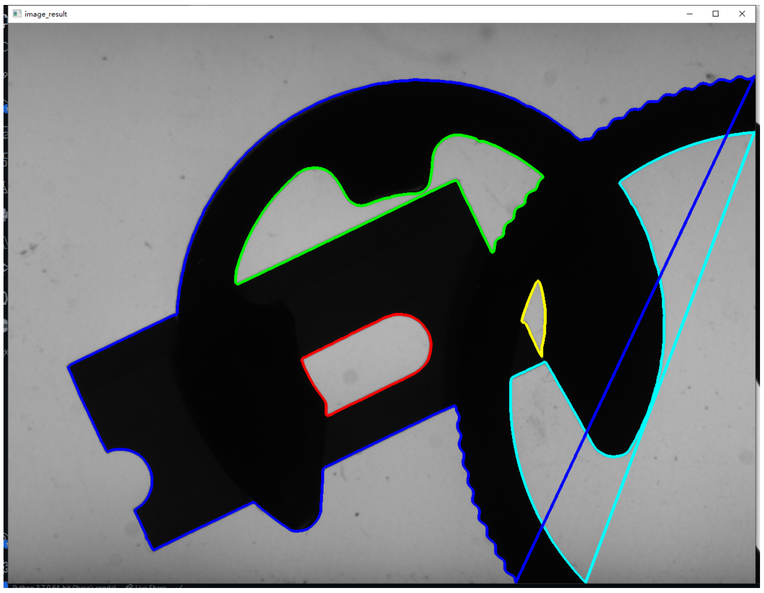


Figure 14 Middle results

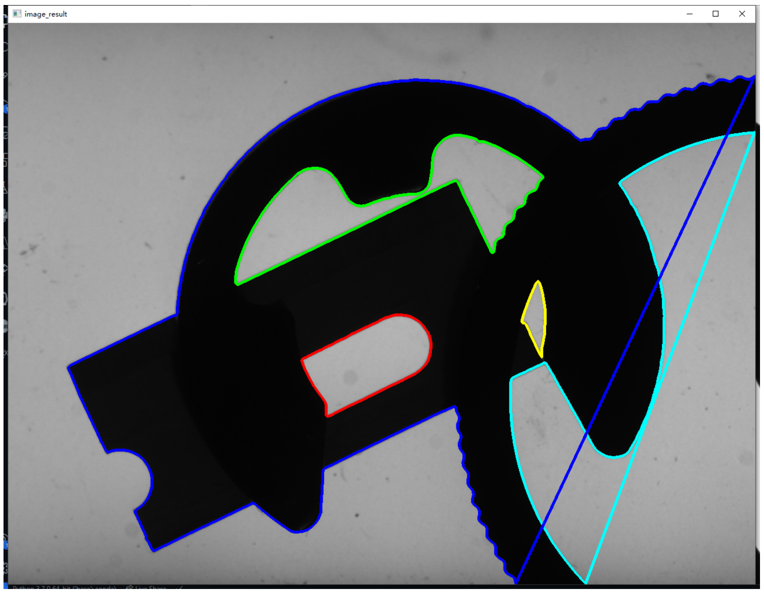


Figure 15 Finally results

contour rendering method and use the method to achieve the desired effect by drawing all contour points to replace the original contour rendering method. Facts have proved that our ideas have been successful and fully meet our expectations.

Finally, the coordinates of contour points, the number of contour points, the number of contours, the total length of contours and the length of each contour are recorded separately. The problem encountered in this process is how to calculate the length of the contour. In order to calculate the length, we calculate the length of each contour through the contour length calculation function before data cleaning, and store it in the array. Then, a counter similar to word frequency statistics is set to count the number of points deleted by each contour. Because the screened contour lines are straight lines, the length of the deleted contour is equal to the number of deleted contour points. Therefore, when we finally output the result, we subtract the number of invalid contour points counted by the counter from each contour based on the counter result, and output the result as an effective contour length.

4.2 Computer ranging based on digital image

Through the processing method of (section 4.1), we successfully obtained various parameter information of the contour. We need to do an operation on these parameter information to measure the actual contour length of the object in the photo.

Because the data measured by the computer is based on digital pictures, the unit of the measured length result is pixels, and what we need is the actual length unit, such as mm, cm, etc. So we need to let the computer know the unit conversion rate between pixel length and actual length. We use a dot matrix calibration plate taken at the same angle as the object, in which the center distance between points and the radius of each point is a predetermined length, the diameter of each point is 1mm, and the center distance between points is 2mm. By processing the calibration plate first, the conversion rate of pixel length and MM is obtained.

We use Hoff circle transformation to analyze the calibration plate picture, and get the radius, perimeter and center coordinates of each point. Then the radius of all points is averaged to obtain the pixel distance between the radius and diameter of the point. **Pixels_per_metric** can be obtained by calculating the pixel distance and the actual preset distance_ per_ Metric, in millimeters per pixel.

$$pixels\_per\_metric = \frac{pixel\_length}{actual\_length}$$

Then the contour of the object to be measured is extracted by the method of (section 4.1), and then pixels are used for the circumference of each contour_ per_ Metric is used for unit conversion to obtain the actual length, and the calculated length results are recorded.

4.3 Contour type detection

Contour type recognition is to analyze the memory category of contour points extracted from the picture to judge whether the contour is line segment, circle, arc or others, so that we can better describe the whole contour. We expect to obtain various attributes of line, circle (or arc), ellipse (or elliptical arc) from the contour point coordinates. For ellipse, we add an attribute called rotation angle[2], so as to fix its standard equation into a format.

Through Hough transform, the contour points in the image can be easily fitted with lines or circles, so as to achieve the purpose of contour type recognition.

V. Conclusions

The conclusions which we get in building our models will be listed as follows:

Conclusions of the problem: Our model basically meets the initial expectation, and also proves that opencv can play an important role in the application of image processing.

Methods used in our models: Image enhancement and smoothing base on 2-d convolution, Image graying Analysis, Image binaryzation Analysis, Edge detection algorithm and Hough circle Transformation are used in our model.

Applications of our models: Finally, the completed model can be applied to the image processing and contour extraction of basic pictures with small scene and less interference.

VI. Future Work

We need to better modify and optimize the anti-interference ability of the model. The current model can not meet the contour extraction of objects under complex lighting conditions, which is obviously a great disadvantage of our model. Only by solving this disadvantage can we apply the model to practical events. That is what we will do in future.

VII. References

[1] Satoshi Suzuki and others, Topological structural analysis of digitized binary images by border following[J], Computer Vision, Graphics, and Image Processing, 1985 , 30(1),32–46

[2] Guiping Hu, Distinguish the centrifugal angle and rotation angle of ellipse[J], Middle school mathematics, 2018, 15, 31-34,

[3] Programming years of Chen January《OpenCv视觉之眼》Python图像处理四 :Opencv图像灰度处理的四种方法及原理 https://blog.csdn.net/qq_42451251/article/details/107783243, 2020.08.04

VIII. Appendix

Listing 1: The python Source code of Contour extraction of binary image and date cleaning

```
import cv2 as cv
import numpy as np
import csv

image = cv.imread('C:\\Users\\25315\\Desktop\\2021 APMCM Problem
    A\\Annex 1\\Pic1_2.bmp')
result = image.copy()
cv.imshow('image',image)
kernel = np.ones((5, 5), np.float32)/10
image = cv.filter2D(image, -1, kernel)
cv.imshow('image_filter2d',image)
imgray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
cv.imshow('image_gray',imgray)
ret, dst = cv.threshold(imgray, 110, 255, 0)
cv.imshow('image_binary',dst)
contours, hierarchy = cv.findContours(dst, cv.RETR_TREE,
    cv.CHAIN_APPROX_NONE)

print(len(contours))

bgr_list = [(0, 0, 255), (0, 255, 255), (0, 255, 0), (255, 255, 0),
    (255, 0, 0), (255, 0, 255)]



contours = [i for i in contours]
dealed = {}
perimeters = []
for i in contours:
    perimeters.append(cv.arcLength(i, True))
print(perimeters)
for key, contour in enumerate(contours):
    rep = 0
    for index, value in enumerate(contour):
        if value[0][0]==0 or value[0][0]==1295 or value[0][1]==971 or
            value[0][1]==0:
            contour = np.delete(contour, index-rep, 0)
            rep += 1
            dealed[str(key)] = dealed.setdefault(str(key), 0) + 1
        dealed[str(key)] = dealed.setdefault(str(key), 0)
    contours[key] = contour
# for index, value in enumerate(contours):
#     perimeter = cv.arcLength(value, True)
#     print(f'{len(value)}, {perimeter:.4f}')
#     cv.drawContours(result, [value], 0, bgr_list[index%6], 3)

total = 0
for index, contour in enumerate(contours):
    with open(f'C:\\Users\\25315\\Desktop\\pic1_2\\{index+1}.csv',
        'w', encoding='utf-8', newline='') as f:
        # perimeter = cv.arcLength(contour, True)
        perimeter = perimeters[index]
        print(f'{len(contour)}, {perimeter-dealed[str(index)]:.4f}')
        total += perimeter-dealed[str(index)]
        for j in contour:
            cv.circle(result, (j[0][0], j[0][1]), 2, bgr_list[index],
                -1)
            w = csv.writer(f)
            w.writerow([j[0][0], j[0][1]])
print(dealed)
print(f'{total:.4f}')

cv.imshow('image_result',result)
# cv.imwrite('C:\\Users\\25315\\Desktop\\2021 APMCM Problem A\\Annex
    1\\result\\pic1_2.bmp', result)
cv.waitKey(0)
```

Listing 2: The python source code of measure actual length

```
import cv2 as cv
import numpy as np
```

```
src = cv.imread('C:\\Users\\25315\\Desktop\\2021 APMCM Problem
    A\\Annex 2\\Pic2_1.bmp')
img = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
cv.imshow('origin', src)

result = src.copy()

circles =
    cv.HoughCircles(img,cv.HOUGH_GRADIENT,1,20,param1=300,param2=30,
  minRadius=0,maxRadius=15)
circles = np.uint16(np.around(circles))

def avg(ls):
    return int(round(sum(ls)/len(ls)))*2

pre = avg(circles[0,:][:,2])
print(pre)

image = cv.imread('C:\\Users\\25315\\Desktop\\2021 APMCM Problem
    A\\Annex 2\\Pic2_4.bmp')
result = image.copy()

kernel = np.ones((5, 5), np.float32)/25
image = cv.filter2D(image, -1, kernel)

imgray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
ret, dst = cv.threshold(imgray, 110, 255, 0)
contours, hierarchy = cv.findContours(dst, cv.RETR_TREE,
    cv.CHAIN_APPROX_NONE)

bgr_list = [(0, 0, 255), (0, 255, 255), (0, 255, 0), (255, 255, 0),
    (255, 0, 0), (255, 0, 255)]

print(len(contours))
for i in contours[1:]:
    print(len(i))
```

```
total = 0
for index, value in enumerate(contours[1:]):
    perimeter = cv.arcLength(value, True)
    print(f'{perimeter/pre:.2f}')
    total += perimeter/pre
    cv.drawContours(result, [value], 0, bgr_list[index%6], 3)
print(f'{total:.2f}')
cv.imshow('image',result)
# cv.imwrite('C:\\Users\\25315\\Desktop\\2021 APMCM Problem A\\Annex
    2\\result\\pic2_4.bmp', result)
cv.waitKey(0)
```