Team Number :	apmcm2102970
Problem Chosen :	A

Abstract

To solve problem 1, the topic requires us to use the sub-pixel edge extraction method and process. Due to the noise and illumination influence of the three images in Annex 1, we first perform Wiener filter denoising and morphology based illumination nonuniformity removal algorithm on the images in Annex 1 to reduce the influence of illumination factors, and then binarize the preprocessed images to obtain black-and-white images, So as to extract its contour. In the extraction process, we first interpolate the image and turn it into a sub-pixel image, then use the improved Sobel operator and OpenCV to detect the straight line and arc segments respectively and mark them with different colors. Finally, the images of the two geometric marks are combined to obtain the extracted contour. At the same time, the topic requires us to output the edge contour data. We use the function CV2. Arclength() in OpenCV to calculate the XY coordinate output of the contour.

For problem 2, the problem requires us to carry out image correction analysis on the product image, and calculate the actual size of the fitting curve segment on the edge of the image. Firstly, according to the pinhole model of the camera and the image acquisition formation of the camera, we first correct the image. Secondly, we use the characteristics of genetic algorithm and camera calibration. We find that the target gradually converges when it converges to about 4000 generations, and calculate the actual ratio value of the template. At the same time, we use the first question method to calculate the contour, Finally, the edge segmentation result is obtained by using the template ratio, and the total length is 2756.129mm.

For problem 3, we first visualize the coordinate data to obtain the image frame. The extraction of each contour is completed based on the model of problem 1. Then, the input contour is approximated by broken lines. The Ramer algorithm is used to segment through the maximum distance of maxlinedist1 and the polygon approximation of the maximum distance of maxlinedist2. Finally, the edge contour curve data of the two pictures are automatically segmented and fitted into straight line segments, arc segments (including circles) or elliptical arc segments (including ellipses), and the parameter forms of 24 contours and segmented curves are extracted from the first picture, and the second picture is divided into 78.

Key words: Image interpolation, Wiener filter denoising, pinhole model, genetic algorithm, camera calibration principle，Ramer two-step maxlinedist

Contents

I. Restatement of the problem

1.1 Background of the problem

With the development of science and technology, people have higher and higher requirements for the measurement accuracy of various workpieces and parts, and higher and higher requirements for measuring instruments. After the camera is calibrated, the distortion of the image can be corrected according to the lattice or chessboard feature information of the calibrated image. The edge of the target object is very useful for image recognition and computer analysis. Edge contains rich internal information (such as direction, step attribute and shape), which is an important attribute to extract image features in image recognition. Image edge contour extraction is not only a very important processing in boundary segmentation, but also a classical problem in image processing.

1.2 Raise questions

- Establish a mathematical model, analyze the sub-pixel edge extraction method and process with accuracy of 1 / 10 pixel and above, draw the extracted edge contour on the image with different colors, and output it as a color edge contour image, and the output data includes the total number of edge contours, the total length of edge contour in the image landmark space, and the number and length of each contour curve, And the X and Y coordinate data of each contour point, the number of contour curves, and the number of points and length data on each curve
- The product image is corrected and analyzed by using the image information of the calibration plate, and how to calculate the actual physical size of the edge segmentation fitting curve segment on the product image as accurately as possible is considered. Please calculate the length of each edge profile (mm), and finally calculate the total length of the edge profile (mm).
- Analyzes the automatic segmentation and fitting of edge contour curve data into straight line segment, arc segment (including circle) or elliptical arc segment (including ellipse), and discusses the model method or strategy of automatic segmentation and fitting of edge contour.

II. Problem analysis

- For question1， because the three images in Annex 1 are affected by noise and illumination, first, the image in Annex 1 needs to be denoised by Wiener filter and the illumination nonuniformity removal algorithm based on morphology to reduce the influence of illumination factors, and then the preprocessed image is binarized to obtain a black-and-white image, so as to extract its contour. In the extraction process, we first interpolate the image and change it into a sub-pixel image, then detect the straight line and arc segments respectively, and mark them with different colors to obtain the extracted contour.

- For question2,according to the pinhole model of the camera and the image acquisition formation of the camera, we can correct the image first. Secondly, we can optimize and find a reasonable correction position by using the characteristics of genetic algorithm and camera calibration.

- For question,the coordinate data is visualized to obtain the image frame. The extraction of each contour is completed based on the model of problem 1, and then the input contour is approximated by broken lines. The Ramer algorithm is used to segment through the maximum distance of maxlinedist1 and the polygon approximation of the maximum distance of maxlinedist2. Finally, the edge contour curve data of the two pictures are automatically segmented and fitted into straight line segments, arc segments (including circles) or elliptical arc segments (including ellipses), and the parameter forms of 24 contours and segmented curves are extracted from the first picture, and the second picture is divided into 78.

III. Model hypothesis

- Assume that the noise of all processed images conforms to Gaussian distribution;
- It is assumed that the processed image has little difference in local pixels;
- It is assumed that all images are based on two-dimensional plane images, regardless of deflection angle and three-dimensional images.

IV. Symbol description

Symbols	*Definition*
$f(x,y)$	Gray image
$n(x,y)$	noise
$M(u,v)$	Wiener filter
$g(x,y)$	Gaussian function2
E	Energy gradient
I_h	High resolution image
$C_{x,y}$	Object center of gravity
f	focal length
x_i	Genes on chromosomes
$f(x_1,\ldots,x_n)$	objective function

V. Establishment and solution of model

5.1 solution of problem 1

5.1.1 Image preprocessing

For pic1_ 1 and pic1_ 2. We find that it is doped with fine noise. Here we use Wiener filtering method for pic1_ 3. There are not only noise but also illumination factors. We use the algorithm based on morphology to remove the uneven illumination to reduce the illumination factors.

① Denoising of two-dimensional image

Wiener filtering algorithm is used to process the image, and the processing method is as follows:

Assuming that $f(x,y)$ is not related to noise $n(x,y)$ and $n(x,y)$ is zero, the transfer function of Wiener filter is[1]:

$$M(u,v)=\frac{H^*(u,v)}{|H(u,v)|^2+R_{NS}(u,v)}$$

Where $H^*(u,v)$ is the complex conjugate of $H(u,v)$, R_{NS} is the signal-to-noise power ratio, $R_{NS}(u,v)=\frac{P_n(u,v)}{P_j(u,v)}$, and $P_n(u,v)$ and $P_j(u,v)$ represent the power spectra of $f(x,y)$ and $n(x,y)$ noise respectively.

It can be seen that there are no poles in the Wiener filter. Assuming that h and the denominator of Wiener filter m become R, and the zero point of H is also transformed into the zero point of Wiener filter, it has an obvious inhibitory effect on noise.
Using Wiener filtering, we analyze pic1 respectively_ 1、Pic1_ 2 and pic1_ 3. Wiener filtering is carried out. See Appendix 1 for the code. The filtering effect is as follows

Figure 5-1 filter and denoise the three pictures in Annex 1

It can be clearly found that most of the noise can be removed, pic1_ 2 more obvious.

② Illumination change

It mainly affects the imaging process of two-dimensional color image. The specific methods to eliminate the influence of illumination are as follows:

In order to make the image evenly distributed in the whole dynamic range of gray value, improve the brightness distribution of the image and enhance the visual effect of the image. Because the algorithm of removing uneven illumination based on morphology has the problem of blurred edge details, it needs to be improved. The main core of the improved algorithm is to detect the edge of the original image. After extracting the edge information, the extracted edge information is used to make up for the edge loss caused by the top hat transform. Compared with other edge detection operators, Gaussian Laplacian (log) operator has the advantages of small amount of calculation, simple operation and high speed. On the premise of effectively avoiding the influence of noise, it can effectively protect the edge detail information, smooth the image, extract finer edge information, and obtain continuous edges. Therefore, LOG operator can better extract the edge information of the collected workpiece image. First filter the image with Gaussian function, and then perform Laplace operation on the filtered image. The boundary point is the point whose value is equal to zero [2]. Log operation:

$$h(x,y)=\nabla^2[g(x,y)]*f(x,y)$$

$f(x,y)$——--image

$$g(x,y)=\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]$$ ————Gaussian function

Find the second partial derivative of the Gaussian function with respect to x, and you can get:

$$\frac{\partial^2 g(x,y)}{\partial x^2}=\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(-\frac{x^2}{\sigma^4}\right)+\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(-\frac{1}{\sigma^2}\right)=\frac{1}{2\pi\sigma^4}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(\frac{x^2}{\sigma^2}-1\right)$$

Similarly, find the second-order partial derivative of G (x, y) with respect to y

$$\frac{\partial^2 g(x,y)}{\partial x^2}=\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(-\frac{x^2}{\sigma^4}\right)+\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(-\frac{1}{\sigma^2}\right)=\frac{1}{2\pi\sigma^4}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\left(\frac{x^2}{\sigma^2}-1\right)$$

$$\nabla^2 g(x,y)=\frac{\partial^2 g(x,y)}{\partial x^2}+\frac{\partial^2 g(x,y)}{\partial y^2}=\frac{1}{2\pi\sigma^4}\left(\frac{x^2+y^2}{\sigma^2}-2\right)\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]$$

The basic idea of the morphological algorithm for eliminating uneven illumination based on log edge detection is: first, open the operation to extract the background, then use the top hat operation to eliminate the influence of uneven illumination, and finally extract the edge information for edge repair. The specific flow of the algorithm is shown in Figure 5-2.

Figure 5-2 flow chart of light repair

The basic steps are as follows:

1) Obtain the original image, first corrode and then expand, that is, open operation, eliminate small particle noise, smooth the boundary, disconnect the adhesion between shadow and object, and extract the image background;

2) The image is transformed by the top hat to get the image after removing the background, which highlights the areas brighter than the original contour, makes up for the cracks and low brightness areas caused by operation, and eliminates the influence caused by uneven illumination;

3) Compare and study various traditional edge detection operators, and select LOG operator to detect and extract the edge of the original image;

4) The extracted edge information is used to make up for the edge loss caused by the top hat transform, and the edge information is repaired. Finally, the image processed by the improved algorithm is obtained.

Due to pic1_ 3 there is brightness information, so the target is obtained by de brightness and binarization. See Appendix 2 for diamagnetic operation, and the results are as follows

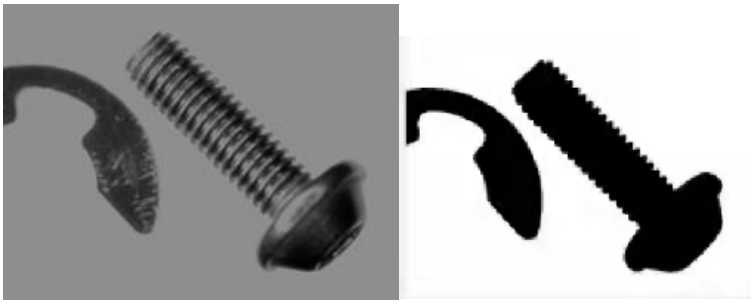

Figure 5-2 De brightening and binarization

5.1.2 modeling of sub-pixel image with edge information

The transformation of digital image resolution is often realized by various interpolation algorithms. However, in these traditional interpolation algorithms, due to insufficient consideration of image edge or texture and other details, the edge of the interpolated image often appears fuzzy or sawtooth. In this paper, the edge detection method is used to preserve the edge information and classify and interpolate the target pixels.

The essence of image interpolation is to find the information of missing pixels in the adjacent pixels. Because human eyes are sensitive to edges, it is necessary to maintain the sharpness and direction of edges during interpolation to avoid large visual errors. Therefore, in order to better preserve the details of the image and maintain the sharpness of the edge, our interpolation algorithm adopts the method of edge detection, considers the influence of the edge on the pixels to be inserted around it, and gives different weights to the edge and non edge pixels around the target pixel for interpolation. The algorithm is mainly divided into the following three steps:

- 1. Perform edge detection on the original image and save the results in the edge marking matrix;
- 2 classify the edge points according to the gray value, enlarge the edge marking matrix and classification matrix for the third step interpolation;
- 3 classify and interpolate the target pixels by using the edge marking matrix and classification matrix.Edge detection:

Firstly, Sobel operator is used to calculate the energy gradient of each pixel of the image on the neighborhood of 3x3. The calculation method of energy gradient e is as follows:

$$E(i, j) = \sqrt{G_x^2 + G_y^2}$$

Where g and G are calculated by Sobel convolution template respectively:

$$G_x(F) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * F \qquad G_y(F) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * F$$

Next, draw the edge line. Taking detecting the edge in the vertical direction as an example, scan line by line from the first line, and mark the pixel points in each line

whose energy gradient is greater than the high threshold Th1 as the starting point of the edge line. Because the gray value changes gently along the direction of the edge line, a relatively low threshold Th2 is used to track the edge.

Finally, the image is denoised because the salt and pepper noise points are scattered in the image and will not be connected into lines. Therefore, if there is no edge point in the eight neighborhoods of an edge point, it is considered as a noise point and marked as a non edge point.

- **Edge point classification**

Because the gray levels of different edges may vary greatly, the weights that should be taken in interpolation are different. Therefore, we classify the edge points according to the gray value: for the two edge points P1 (i.j) and P2 (U, V), if the difference of their gray values is less than a certain threshold K, they are divided into the same kind of edge points.

- **Interpolation**

In order to accurately evaluate the edge and prevent interpolation across the edge region, an algorithm of classified interpolation in small window is proposed in this paper. Assume a size of 2N directly from one 2N×2M high resolution image I. The problem to be solved by interpolation is to use the low resolution image with the size of nxm to reconstruct the size of 2N×2M high resolution image.

Firstly, according to the characteristics of the image, the target pixels are divided into the following three categories, as shown in Fig. 5-3:

Next, the pixels are classified and interpolated.

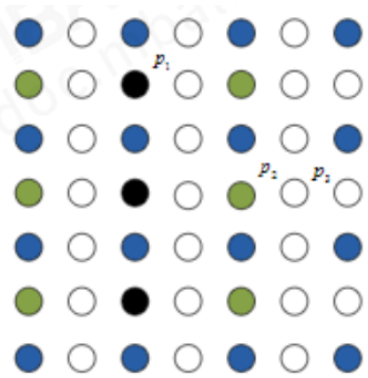

Fig. 5-3Classification interpolation of pixels

Blue represents the original pixels in the low resolution image, black represents the edge points to be inserted, green represents the pixels to be inserted by the edge line, and white represents other pixels to be inserted.For linear pixels, its value is calculated from the surrounding similar edge image turbulence points. The interpolation formula is as follows[3]:

$$P_i = \sum_{P_j \in \Omega_1} \frac{P_j}{d_j} \sum_{P_j \in \Omega_i} \frac{1}{d_j}$$

Where, represents the edge point set of the same category as the point to be inserted PI in the window of NN, dj represents the distance between the reference point and the

point to be inserted; d. The calculation method is as follows:

$$d_j = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$$

X and Y represent the coordinate positions of image pixels.

5.1.3 edge contour detection and calibration

Therefore, Sobel oblique operator is used for edge detection in this paper, and its operator template is shown in Figure 5-4.

Firstly, the edge position of the image (such as the upper left corner) is judged by the two-point characteristics of the arc edge, and then

The oblique Sobel operator corresponding to the change direction of the position pixel detects the image. This method can effectively retain the real edge information of the arc and eliminate some noise points at the same time.

According to Hough transform, the linear contour and circular contour of the constructed sub-pixel image are calibrated and combined. The calibration results are as follows:

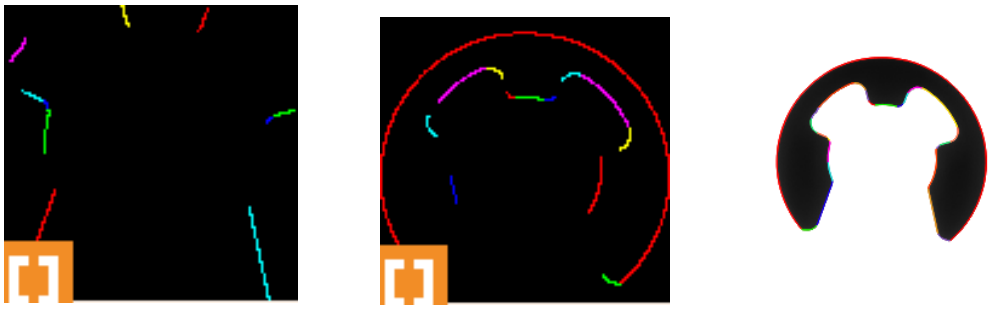

Figure 5-4 Pic1_ 1 linear contour, circular contour calibration and combined calibration

Figure 5-5 Pic1_ 2 linear contour, circular contour calibration and combined calibration

Figure 5-6 Pic1_ 3 linear contour, circular contour calibration and combined calibration

Using OpenCV, the image moment of its calibrated contour is calculated in the worksheet, and the center of gravity of the object is

$$C_x = \frac{M_{10}}{M_{00}}, C_y = \frac{M_{01}}{M_{00}}$$

The second parameter of the function can be used to specify whether the shape of the object is closed (true) or open (a curve).

In the worksheet, we output the XY coordinates of the marked contour line. The output data is shown in the table below. See Annex 3 for the complete results.

Table 5-1.1_ 1 edge profile data output format

Total Edge Contours Count		24
Total Edge Contours Length		2002.0011
Edge Contour 1	Length	151.211
	points	149
Edge Contour 2	Length	71.247
	points	77
…	…	…

Table 5-2.1_ 2 edge contour data output format

Total Edge Contours Count		90
Total Edge Contours Length		4664.67
Edge Contour 1	Length	13.75
	points	16
Edge Contour 2	Length	129.047
	points	137
…	…	…

Table 5-3. 1_ 3 edge contour data output format

Total Edge Contours Count	28
Total Edge Contours Length	28.88

5.2 solution of problem 2

According to the characteristics of genetic algorithm and camera calibration, a calibration method based on genetic algorithm is proposed. This method makes full use of the global optimization of genetic algorithm and the local convergence of neural network. Finally, the actual ratio of the template is obtained, and then the contour is obtained by the first question method. Finally, the result is obtained by the template ratio.

5.2.1. Pinhole camera model

Before introducing the camera calibration parameters, we need to briefly talk about the principle of pinhole camera. The distance from the projection plane to the small hole is the focal length f, and the distance from the object to the small hole is Z. the relationship between the object and the projection is inverted and similar. The following figure is the projection diagram of the pinhole camera:
If the coordinate system is established according to the actual projection relationship, the symbols of the projection coordinates and the object coordinates are always opposite, which is inconvenient to consider. Therefore, translate the projection plane to its symmetrical position about the small hole, so that the symbols of the projection coordinates and the object coordinates are the same. According to the principle of triangle similarity, the following equation can be listed:

$$\frac{f}{Z}=\frac{l_{投影}}{l_{物体}}$$

5.2.2. Coordinate system in camera

There are four coordinate systems in the camera, all of which are right-hand coordinate systems, respectively recorded as world, camera, picture, pixel.

The coordinates of world, camera, picture, pixel coordinate systems are distinguished by subscripts, which are w, C, P and pix respectively.

The unit of world, camera, picture coordinate system is length, generally mm; The unit of pixel coordinate system is pixel, generally pix.

The world coordinate system is a world coordinate system, which can be specified arbitrarily. Other coordinate systems have clear definitions.

The camera coordinate system is the camera coordinate system, the origin is at the position of the small hole, the Z axis coincides with the optical axis, and the XC axis

and YC axis are parallel to both sides of the projection plane respectively.

The picture coordinate system is the image coordinate system, the intersection of the optical axis and the projection plane is the origin, and the XP axis and YP axis are parallel to both sides of the projection plane respectively.

The pixel coordinate system is a pixel coordinate system. When viewed from the small hole to the projection plane, the upper left corner of the projection plane is the origin opix, and the xpix axis and ypix axis coincide with both sides of the projection plane.

5.2.3 From {world} to {pixel}

If the coordinate of a point in the {world} coordinate system is, and the coordinate of the point in the {camera} coordinate system is PC, there is

$$P_c = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_W$$

According to the triangle similarity principle,

$$\begin{cases} x_p = f \dfrac{x_c}{z_c} \\ y_p = f \dfrac{y_c}{z_c} \end{cases}$$

Remember

$$\begin{cases} f_x = f \cdot s_x \\ f_y = f \cdot s_y \end{cases}$$

Represents the equivalent focal length of focal length f in xpix and ypix directions respectively, and the unit is pix,

$$z_c \begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_W \\ y_W \\ z_W \\ 1 \end{bmatrix}$$

Radial distortion will produce "fish eye" phenomenon. The radial distortion at the imaging center is 0. The radial distortion increases with the increase of the distance from the imaging center, and reaches the maximum radial distortion at the image edge. Even power Taylor formula is often used to describe radial distortion

Tangential distortion is caused by non parallelism between lens and imaging plane. It is commonly described by the following formula

$$\begin{cases} x_{\text{cerrected}} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{\text{cerrected}} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{cases}$$

It is commonly described by the following formula

$$\begin{cases} x_{\text{cerrected}} = x + \left[2p_1 y + p_2 \left(r^2 + 2x^2 \right) \right] \\ y_{\text{cerrected}} = y + \left[p_1 \left(r^2 + 2y^2 \right) + 2p_2 x \right] \end{cases}$$

5.2.4 camera calibration based on genetic algorithm

From the mathematical model of nonlinear programming, we can see that the final solution is the single objective calibration solution, which meets the constraints and makes the objective function reach the minimum (or maximum). Take this set of solutions as chromosomes, each of which is equivalent to a gene on the chromosome, which is the coding object. The subsequent selection, crossover and mutation are the calculation of the encoded chromosome genes.

By punishing infeasible solutions, the constrained problem is transformed into an unconstrained problem. In genetic algorithm, penalty technique is used to keep some infeasible solutions in each generation of population, so that genetic search can reach the optimal solution from both feasible and infeasible regions.

Genetic algorithm uses genetic operator to eliminate the fittest. We use the most common proportional selection operator, which is a playback random sampling method. The basic idea is that the probability of each individual being selected is directly proportional to its fitness.

In most cases, the runner method is used as the selection method. It is a proportional selection strategy, which can select a new population according to the probability proportional to the fitness value. The runner method consists of the following steps

- step 1. Calculate fitness value for each chromosome;

$$eval(v_k) = f(x); \quad \text{k=1,2,…,size}$$

- step 2. Calculate the fitness and of all chromosomes in the population;

$$F = \sum_{k=1}^{pop\_size} eval(v_k)$$

- step3. Calculate the selection probability for each chromosome;

$$p_k = \frac{eval(v_k)}{F}; \quad \text{k=1,2,…,size}$$

- step4. Calculate the cumulative probability for each chromosome;

$$q_k = \sum_{j=1}^{k} p_j; \quad \text{k=1,2,…,size}$$

Table 5-4 Operation parameter table of genetic operator

Coding length	Population size	Crossover probability PC	Variation probability PM	Termination algebra T

13	160	0.8	0.2	150

5.2.5 Model solving

The convergence results of single target calibration obtained by genetic algorithm are as follows

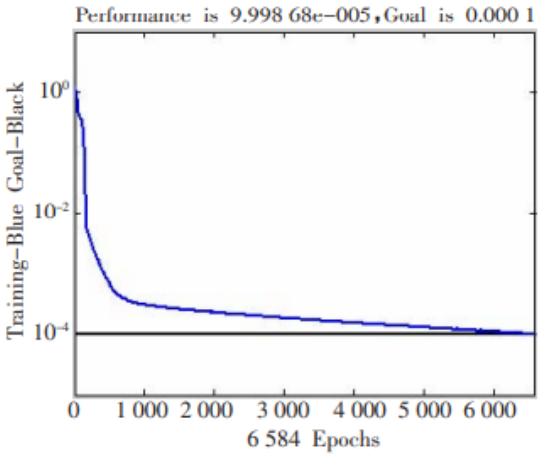

Fig. 5-7 variation curve of optimal fitness solved by genetic algorithm

The obtained parameter results are shown in the figure. The correction results of operation are as follows. See the appendix for the code

u_c	v_c	x_c	y_c
118.03	252.38	0.888 89	−1.388 90
147.99	354.02	1.388 90	−1.388 90
147.13	385.58	1.388 90	−0.888 89
116.98	383.78	0.888 89	−0.888 89
287.56	301.41	3.555 60	−2.277 80
320.15	302.37	4.055 60	−2.277 80
319.20	334.83	4.055 60	−1.777 80
286.64	333.97	3.555 60	−1.777 80
347.44	244.17	4.444 40	−3.166 70
380.30	244.68	4.944 40	−3.166 70
379.62	277.51	4.944 40	−2.666 70
346.61	276.63	4.444 40	−2.666 70
76.167	129.06	0.000 00	−4.944 40
104.59	128.84	0.500 00	−4.944 40
102.88	158.96	0.500 00	−4.444 40
74.544	158.75	0.000 00	−4.444 40
130.72	74.897	0.888 89	−5.833 30
159.80	74.441	1.388 90	5.833 30
158.56	104.18	1.388 90	−5.333 30
129.17	104.50	0.888 89	−5.333 30
295.99	18.781	3.555 60	−6.722 20
326.99	18.469	4.055 60	−6.722 20
325.81	48.099	4.055 60	−6.222 20
294.53	48.457	3.555 60	−6.222 20

Fig. 5-8The first two columns are virtual x, y, and the last two columns are real values

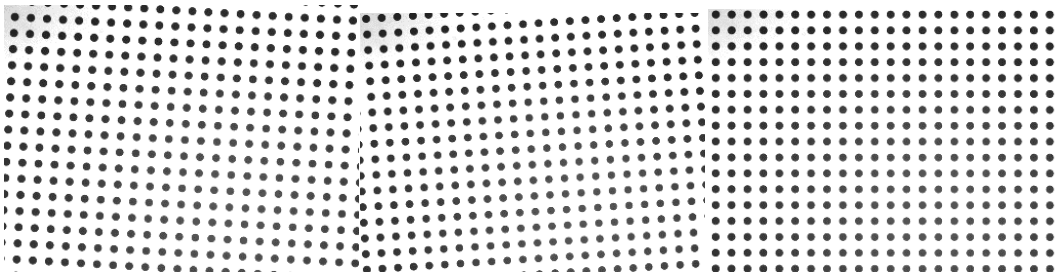

Fig. 5-9 The correction results of operation

5.2.6 contour segmentation results

We use the model of problem 1 for contour segmentation. After some column preprocessing such as sub-pixel processing, we respectively carry out geometric edges on the 2.4 images in Annex 2, and finally merge the geometric contour extraction results to obtain the results. The specific results are as follows

Fig. 5-10 Pic2_ 4 edge contour extraction

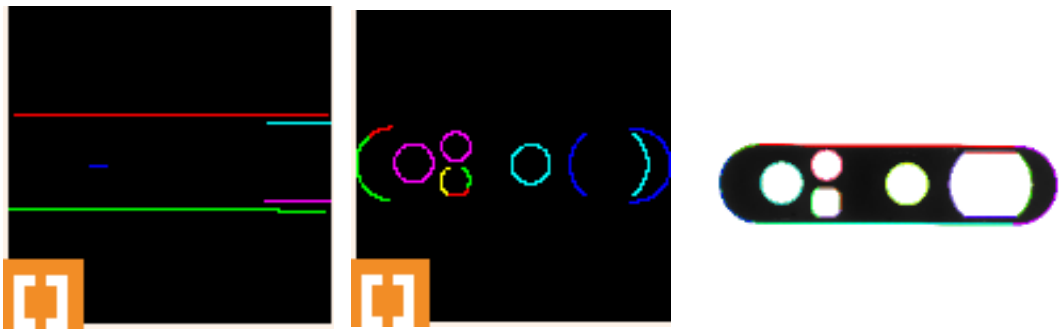

Fig. 5-11 Pic2_ 4 linear contour, circular contour calibration and combined calibration

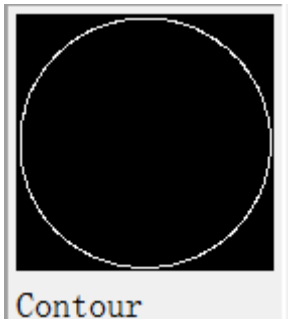

Fig. 5-12 splitting individual examples

At the same time, we finally fuse the single target calibration results with the contour segmentation results, as shown in the table, and the complete results are shown in the annex.

Table 5 - 5 edge profile length output format (mm)

Outline of the logo	Length(mm)	Outline of the logo	Length(mm)
Total Edge Contours		2756.129	
Edge Contour 1	817.018	Edge Contour 11	49.5999
Edge Contour 2	828.041	Edge Contour 12	52.9546
Edge Contour 3	44.9331	Edge Contour 13	49.2823
Edge Contour 4	167.042	Edge Contour 14	54.3006
Edge Contour 5	166.992	Edge Contour 15	49.7909
Edge Contour 6	44.6967	Edge Contour 16	55.3085
Edge Contour 7	51.3794	Edge Contour 17	42.382
Edge Contour 8	43.2656	Edge Contour 18	50.6533
Edge Contour 9	41.736	Edge Contour 19	52.0006
Edge Contour 10	39.9583	Edge Contour 20	54.7943

5.3 problem solving

5.3.1 picture pixel coordinate visualization

According to the two sub-pixel contour edge data (edgecontour1.xls and edgecontour2. XLS) provided in question 3 and the picture 5 in the question, we label the pixel information on the xoy plane with MATLAB according to the pixel information, and the results are shown in the following ：

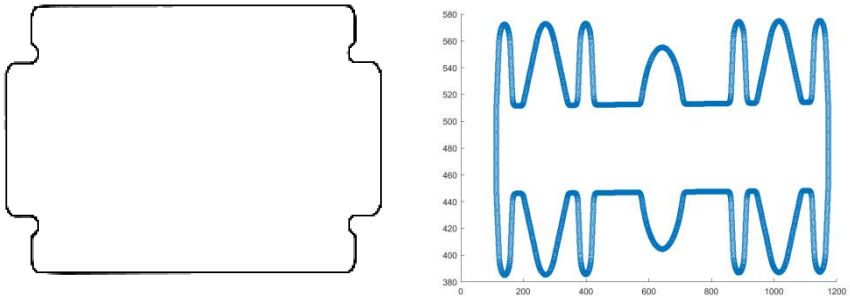

Figure 5 -13 image pixel coordinate visualization

5.3.2 model solution

➢ Step 1:

Firstly, according to the characteristics of the image, the target pixels are divided into the following three categories

Next, the pixels are classified and interpolated.

For linear pixels, its value is calculated from the surrounding similar edge image turbulence points. The interpolation formula is as follows:

$$P_i = \sum_{P_j \in \Omega_1} \frac{P_j}{d_j} \sum_{P_j \in \Omega_i} \frac{1}{d_j}$$

Where, represents the edge point set of the same category as the point to be inserted PI in the window of NN, DJ represents the distance between the reference point and the point to be inserted; d. The calculation method is as follows:

X and Y represent the coordinate positions of image pixels.

➢ Step 2:

Firstly, the polyline is used to approximate the input contour, so that the contour will be excessively segmented at the bending place. If the contour can be better approximated by an arc, the arc or elliptical arc will be used to replace the adjacent line segments respectively. If smoothcont is set to > 0, it is necessary to smooth the input contour first. Because smoothing suppresses the abnormal values on the contour, on the one hand, it can prevent the abnormalities caused by segmentation of particularly short lines. On the other hand, it can achieve more robust segmentation when using circle or ellipse segmentation;

The initial polyline approximation is completed by using the Ramer algorithm through the maximum distance of maxlinedist1. After that, the circular or elliptical arc

is matched to the adjacent line segments. If the maximum distance from the generated arc to the contour line is less than the maximum distance of the two line segments, the two line segments are replaced by arcs, and the process is iterated until no change occurs;

After that, the contour part still approximated by the line segment is segmented again with the polygon approximation of the maximum distance maxlinedist2, and the newly created line segment is merged into a circular or elliptical arc if possible. Obviously, this will only change the output when maxlinedist2 < maxlinedist1. This two-step method is more effective than the one-step method using maxlinedist2. Since there are fewer segments generated in the first step, there must be less circle or ellipse fitting. Therefore, it is more efficient to use the long arc to approximate the partial input contour; Then, the short arc is used to approximate the input contour, and finally the end of the contour approximated by the long arc is refined;

The resulting contour has a length of at least 3 pixels and includes at least 6 consecutive points of the input contour; All input contours with a length of less than 3 pixels or less than 6 contour points will be copied to the output contour without any modification.

5.3.3 result analysis

First, the first mock exam is used to extract the edges and to segment the contour.

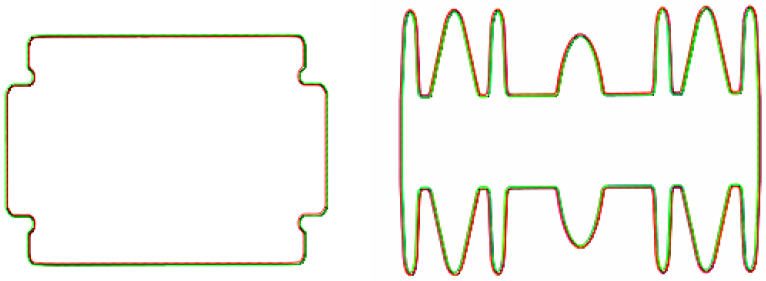

Figure 5 -13 Edge extraction graph

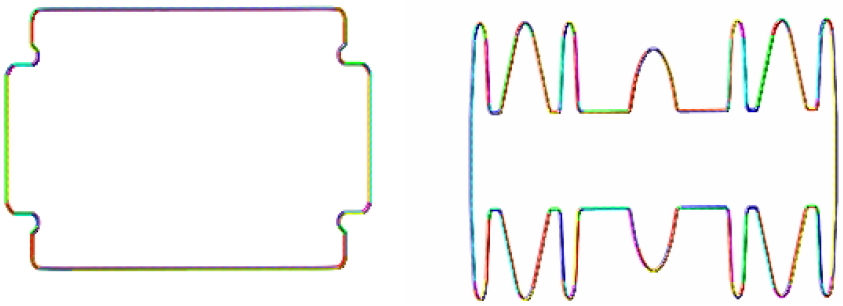

Figure 5 -14 Contour segmentation map

Specific segmented extraction:

Here, take the first three and the last one in the specified direction as an example

(there are 24 pictures and 78 pictures in total):

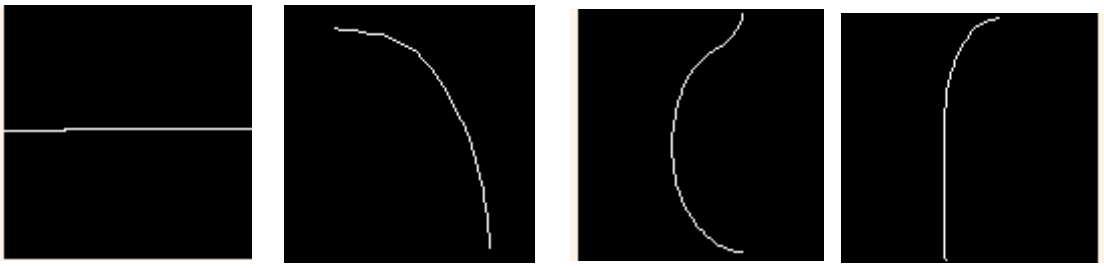

Figure 5 -15 Picture one segment

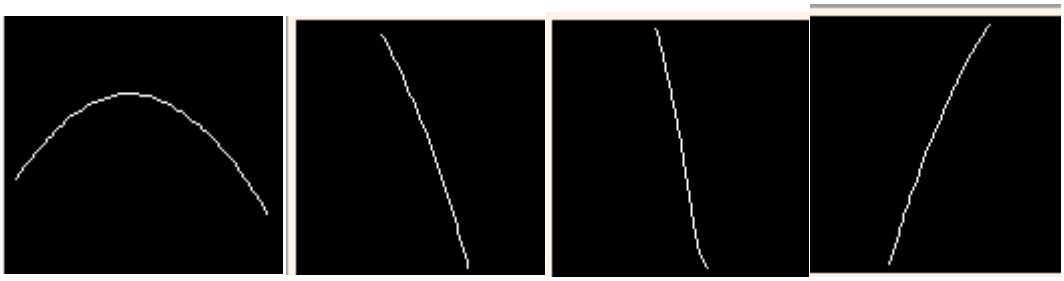

Figure 5 -16 Picture two segment

The parameter results of each segment are as follows:

Edge Contour 1							
NO	TYPE	PARAM					
S1	Line	(133.075, 134.284)	(859.018, 224.98)	634.17pixel			
S2	CircularArc	(152.013, 856.226)	(155.042, 130.671)	(877.284, 865.651)	-5.4		
S3	Line	(155.042, 130.671)	(877.284, 865.651)	39.05pixel			
S4	EllipticArc	(217.463, 877.824)	[21.0626, 15.5335]	(155.042, 130.671)	(877.284, 865.651)	-3.2	1.36721
S5		(239.061, 239.048)	(914.91, 874.894)	40.19pixel			
...							
S24	CircularArc	(172.31, 272.045)	(135.627, 194.987)	(213.055, 211.74)	1.1		

Edge Contour 2					
NO	TYPE	PARAM			
S1	ircularAr	(153.753, 610.053)	(629.842, 595.314)	(145.379, 137.912)	2.8
S2	Line	(191.607, 152.608)	(644.027, 628.925)	41.93pixel	
S3	Line	(264.362, 191.816)	(657.217, 643.998)	74.16pixel	
S4	Line	(265.036, 266.037)	(763.858, 658.974)	105.45pixel	
S5	Line	(248.025, 741.489)	(245.094, 264.149)	(769.675, 766.133)	5.57
...					
S78	Line	(147.536, 186.567)	(591.784, 575.887)	42.32pixel	

VI. Advantages and disadvantages of the model

6.1 model evaluation

Advantages

- The model is widely used in computer vision;

- The results of the model are generated programmatically and have credibility;
- Problem three two-step method is more efficient than single-step method using MaxLineDist2 because the first step generates fewer line segments and therefore circle or ellipse fitting is performed fewer times;

Disadvantages

- problem 1 light image noise reduction is more troublesome, more troublesome to deal with;
- The picture may be slightly different from the one in question.

6.2 model promotion

More complex images can be tried for edge detection and segmentation, which can be processed using neural networks.

VII. References

[1] Support Vector Machines; Studies from Xi'an Jiao Tong University Yield New Data on Support Vector Machines (Partial Discharge Source Localization In Gis Based On Image Edge Detection and Support Vector Machine)[J]. Journal of Robotics & Machine Learning,2019:

[2]Jufriadif Na`am,Johan Harlan,Sarifuddin Madenda,Eri Prasetio Wibowo. The Algorithm of Image Edge Detection on Panoramic Dental X-Ray using Multiple Morphological Gradient (mMG) Method[J]. International Journal on Advanced Science, Engineering and Information Technology,2016,6(6):

[3]Zahra Zareizadeh,Reza P.R. Hasanzadeh,Gholamreza Baghersalimi. A recursive color image edge detection method using Green's function approach[J]. Optik - International Journal for Light and Electron Optics,2013,124(21):

[4]Qirong Lu,Qianmin Liang,Jiqiu Chen,Jiwei Xia. Image Edge Detection Method Based on Ant Colony Algorithm[C]//Abstracts of the 5th International Conference of Pioneering Computer Scientists,Engineers and Educators(ICPCSEE 2019)Part II.,2019:37.

[5]Jinping Cai,Cong He. Study on image edge detection based on GAACA[C]//Proceedings of 2014 International Conference on Simulation and Modeling Methodologies,Technologies and Applications(SMTA 2014 VI).,2014:495-

501.

The appendix

Appendix 1 Code

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

def localStd(img):
    # img = img / 255.0

    img_blur = cv.blur(img, (21, 21))
    reslut_1 = img_blur ** 2

    img_2 = img ** 2
    reslut_2 = cv.blur(img_2, (21, 21))

    reslut = np.sqrt(np.maximum(reslut_2 - reslut_1, 0))
    return reslut


def get_reflect(img, img_illumination):
    # get_img_illumination = get_illumination(img)
    get_img_reflect = (img + 0.001) / (img_illumination + 0.001)
    return get_img_reflect


def enhancement_reflect(img):

    gaussian_blur_img = cv.GaussianBlur(img, (21, 21), 0)
    enhancement_reflect_img = img * gaussian_blur_img
    return enhancement_reflect_img


def get_enhancment_img(img_enhance_illumination, img_enahnce_reflect):
    img = img_enhance_illumination * img_enahnce_reflect
    img = img.astype('uint8')
    return img


def read_img_from_disk(file_path):
```

```
    img = cv.imread(file_path, cv.IMREAD_COLOR)
    return img


def get_illumination(img):
    return cv.GaussianBlur(img, (15, 15), 0)


"""
enhancment_illumination
"""


def enhancment_illumination(img_illumination):
    img_hsv = cv.cvtColor(img_illumination, cv.COLOR_BGR2HSV)
    img_hsv = (img_hsv - np.min(img_hsv)) / (np.max(img_hsv) - np.min(img_hsv))
    h, s, v = cv.split(img_hsv)
    wsd = 5
    gm = np.mean(v) / (1 + wsd * np.std(v)) #  一个数字
    cst = localStd(v)     # 300 * 400  的矩阵
    lm = gm * v /(1 + wsd * cst)       # 300 * 400  的矩阵
    c = np.exp(gm)          #  一个常数
    wg = v ** 0.2           # 300 *400
    wl = 1- wg
    outM = v**c / (v**c +(wl * lm)**c + (wg * gm)**c + 0.001)
    outM = 1.5 * outM - 0.5 * cv.GaussianBlur(outM, (21, 21), 0)
    outM = (outM - np.min(outM))/(np.max(outM) - np.min(outM))
    paramerter = 0.9
    img_illumination[:, :, 0] = outM * (img_illumination[:, :, 0] / (v +
0.01))**paramerter
    img_illumination[:, :, 1] = outM * (img_illumination[:, :, 1] / (v +
0.01))**paramerter
    img_illumination[:, :, 2] = outM * (img_illumination[:, :, 2] / (v +
0.01))**paramerter
    return img_illumination


if __name__ == '__main__':
    file_name =  r'C:\Users\wsq love sy\Desktop\yatai\2021 APMCM Problem
A\Annex 1\Pic1_3.bmp'
    img = read_img_from_disk(file_name)
    # img = cv.resize(img, (0, 0), fx=0.2, fy=0.2, interpolation=cv.INTER_NEAREST)
```

```
    img_illumination = get_illumination(img)
    img_reflect = get_reflect(img, img_illumination)
        img_enhancement_reflect = enhancement_reflect(img_reflect)
    img_enhancement_illumination = enhancment_illumination(img_illumination)
img_done = get_enhancment_img(img_enhancement_illumination, img_reflect)
    cv.imwrite("./img_reflect.png", img_reflect)
    cv.imshow("src", img)
    cv.imshow("img_reflect", img_reflect)
    cv.imshow("img_illumination", img_illumination)
    cv.imshow("done", img_done)
    cv.waitKey(0)
    # cv.imwrite("./data/done/matlab/src_img.png", img)
    # cv.imwrite("./data/done/matlab/img_illumination.png", img_illumination)
    # cv.imwrite("./data/done/matlab/img_reflect.png", img_reflect)
    #                         cv.imwrite("./data/done/matlab/illumination_done.png",
img_enhancement_illumination)
    #                 cv.imwrite("./data/done/matlab/img_enhancement_reflect.png",
img_enhancement_reflect)
    cv.imwrite("./data/done/matlab/img_done.png", img_done)
    print("done")
    cv.destroyAllWindows()
    print("done for everthong")
X1=imread('Pic1_3.bmp');

figure
subplot(131)
imshow(X1)
subplot(132)
imhist(X1)
XX1=im2bw(X1,254/255);
subplot(133)
for kk=1:2
m=30;
for i=m+1:size(XX1,1)-m
    for j=m+1:size(XX1,2)-m
        a=XX1(i-m:i+m,j-m:j+m);
        if length(find(a(1:m,m+1)==0))>0 & length(find(a(m+1:end,m+1)==0))>0
& length(find(a(m+1,1:m)==0))>0 & length(find(a(m+1,m+1:end)==0))>0
            XX1(i,j)=0;
        end
    end
end
end
imshow(XX1)
```

```
Ig=XX1;
s=GetStrelList();
e=ErodeList(Ig,s);
f=GetRateList(Ig,e);
Igo=GetRemoveResult(f,e);
psnr1=PSNR(XX1,e.eroded_co12);
psnr2=PSNR(XX1,e.eroded_co22);
psnr3=PSNR(XX1,e.eroded_co32);
psnr4=PSNR(XX1,e.eroded_co42);
psnr5=PSNR(XX1,Igo);
psnr_list=[psnr1 psnr2 psnr3 psnr4 psnr5];
M{1,1}=e.eroded_co12;
M{1,2}=e.eroded_co22;
M{1,3}=e.eroded_co32;
M{1,4}=e.eroded_co42;
M{1,5}=Igo;

figure
plot(1:5,psnr_list,'r+-');
set(gca,'XTick',0:6,'XTickLabel',{"'"});
grid on;
title('PSNR')

[~,b]=max(psnr_list);
result=M{1,b};


figure
subplot(131)
imshow(result)
subplot(132)
imhist(result)
result=im2bw(result,0.3);
subplot(133)
Y1=bwperim(result);
imshow(Y1)
```

Appendix 2 Code

```
#include "opencv2/opencv.hpp"
#include <string>
#include <iostream>
```

```
using namespace cv;
using namespace std;

int main()
{
    VideoCapture inputVideo(0);
        //inputVideo.set(CAP_PROP_FRAME_WIDTH, 320);
    //inputVideo.set(CAP_PROP_FRAME_HEIGHT, 240);
    if (!inputVideo.isOpened())
    {
        cout << "Could not open the input video " << endl;
        return -1;
    }
    Mat frame;
    string imgname;
    int f = 1;
    while (1) //Show the image captured in the window and repeat
    {
        inputVideo >> frame;                    // read
        if (frame.empty()) break;               // check if at end
        imshow("Camera", frame);
        char key = waitKey(1);
        if (key == 27) break;
        if (key == 'q' || key == 'Q')
        {
            imgname = to_string(f++) + ".jpg";
            imwrite(imgname, frame);
        }
    }
    cout << "Finished writing" << endl;
    return 0;
}
read_image (Image, 'C:/Users//Desktop/yatai/2021 APMCM Problem A/Annex 1/Pic1_1.png')
edges_sub_pix (Image, Edges, 'canny', 1.5, 15, 40)
segment_contours_xld (Edges, ContoursSplit, 'lines_circles', 5, 4, 2)
count_obj (ContoursSplit, Number)
gen_empty_obj (Lines)
gen_empty_obj (Circles)
for I := 1 to Number by 1

  select_obj (ContoursSplit, Contour, I)

  get_contour_global_attrib_xld (Contour, 'cont_approx', Type)
```

```
    if (Type = -1)

        concat_obj (Lines, Contour, Lines)

        get_contour_xld(Contour, Row, Col)
    else

        concat_obj (Circles, Contour, Circles)

    endif

endfor
fit_line_contour_xld (Lines, 'tukey', -1, 0, 5, 2, RowBegin, ColBegin, RowEnd, ColEnd,
Nr, Nc, Dist)
```

Appendix 3 Code

```
read_image (Image, 'C:/Users/y/Desktop/yatai/2021 APMCM Problem A/Annex
1/Pic1_1.png')
get_image_size(Image,Width, Height)
lines_gauss (Image, Lines, 2.3, 0, 0.7, 'dark', 'true', 'parabolic', 'true')
select_contours_xld (Lines, RelLines, 'length', 0.5, 999, 0, 0)
dev_display (RelLines)
count_obj (RelLines, Number)


read_image (Image, 'C:/Users//Desktop/yatai/2021 APMCM Problem A/Annex
1/Pic1_1.png')
edges_sub_pix (Image, Edges, 'canny', 1.5, 15, 40)
segment_contours_xld (Edges, ContoursSplit, 'lines_circles', 5, 4, 2)
count_obj (ContoursSplit, Number)
gen_empty_obj (Lines)
gen_empty_obj (Circles)
for I := 1 to Number by 1

    select_obj (ContoursSplit, Contour, I)

    get_contour_global_attrib_xld (Contour, 'cont_approx', Type)

    if (Type = -1)

        concat_obj (Lines, Contour, Lines)
```

```
        get_contour_xld(Contour, Row, Col)
    else

        concat_obj (Circles, Contour, Circles)

    endif

endfor
fit_line_contour_xld (Lines, 'tukey', -1, 0, 5, 2, RowBegin, ColBegin, RowEnd, ColEnd,
Nr, Nc, Dist)
gen_polygons_xld (Lines, Polygons, 'ramer', 10)
get_lines_xld (Polygons, BeginRow, BeginCol, EndRow, EndCol, Length, Phi)
read_image (Image, 'C:/Users/y/Desktop/322.png')
edges_sub_pix (Image, Edges, 'canny', 1.5, 15, 40)
segment_contours_xld (Edges, ContoursSplit, 'lines_circles', 5, 4, 2)
count_obj (ContoursSplit, Number)
select_obj (ContoursSplit, Contour, 81)
get_contour_global_attrib_xld (Contour, 'cont_approx', Type)
fit_line_contour_xld (Contour, 'tukey', -1, 0, 5, 2, RowBegin, ColBegin, RowEnd,
ColEnd, Nr, Nc, Dist)
fit_circle_contour_xld (Contour, 'atukey', -1, 2, 0, 3, 2, Row, Column, Radius, StartPhi,
EndPhi, PointOrder)
*fit_ellipse_contour_xld (Contour, 'fitzgibbon', -1, 0, 0, 200, 3, 2, Row, Column, Phi,
Radius1, Radius2, StartPhi, EndPhi, PointOrder)
```