Team Number:	2100148
Problem Chosen:	A

2021 APMCM summary sheet

To meet the high-precision requirements of workpiece and part measurement, the sub-pixel level extraction of image edge is realized by selecting appropriate parameters through Canny algorithm. The distorted image is calibrated by dot matrix, the mapping relationship between image coordinate space and world coordinate space is calculated. The distorted image is restored truly, and the accurate scale and size are obtained. Aiming at the problem of primitive segmentation of plane contour, the polygon algorithm is used to determine the location of segmentation points, identify the line segment types between segmentation points, and carry out primitive fusion. Finally, the least square fitting algorithm is used to suppress the maximum outliers and form the corresponding geometric primitives. The detection accuracy, detection speed and primitive recognition ability of the method are verified. The results show that the algorithm has good accuracy and portability.

Keywords: Machine Vision; Polygonal Approximation; Image Denoising; Edge Detection; Canny Operator; Image Pose Correction; Image Calibration; Edge Segmentation Fitting; Primitive Fitting

Contents

I. Introduction

1 . 1 Background

With the development of science and technology, the accuracy of the measurement of the different pieces and parts is increasing and the requirements for measuring instruments are growing. Different imaging instruments, such as digital imaging measuring instruments, gradually replace traditional manual measuring applications. Generally, after calibrating the camera, the image is distorted and corrected according to the point matrix or the function of the chessboard of the calibrated image and the mapping relationship between the space of the image coordinate map and the global coordinating space.

The edges of the target object are very useful in image recognition and computer analysis. The edge of the image is a reflection of the discontinuity of the local features of the image. Edges can outline the target object, giving the observer a clear view. Edges contain rich intrinsic information (such as direction, step properties, shapes, etc.), which is an important attribute in image recognition to extract image features. Image edge contour extraction is a very important process in image boundary segmentation, and it is also a classic problem in image processing. Contour extraction and profile tracking are designed to obtain the outer contour features of the image. Applying certain methods to express the characteristics of contours when necessary, preparing for image shape analysis, has a significant impact on performing advanced processing, such as feature description, identification, and understanding.

1 . 2 Restatement of the problem

The question requires a mathematical model to analyze the method and process of sub-pixel edge extraction. There are three pictures with 1 / 10 pixel accuracy and above, and the edge part of the main object of sub-pixel edge extraction contour boundary. When converting sub-pixel edge point data into command edge contour curve data, we need to consider how to eliminate the interference effects of edge burrs and shadow parts, especially those taken under relatively complex lighting conditions.

After establishing a mathematical model, use the image information of the calibration board to correct and analyze the product image, and consider how to accurately calculate the actual physical size of the edge segmentation fitting curve segment on the product image. Please calculate the length of each edge profile, and finally calculate the total edge profile length.

Establish a mathematical model, analyze the automatic segmentation and fitting of edge contour curve data into straight line segment, arc segment and elliptical arc segment, and discuss the model method or strategy of automatic segmentation and fitting of edge contour.

II. Model building and solution of question one

2.1 The algorithm flow of canny

2.1.1 *Grayscale*

This part is usually processed according to Canny algorithm. The image is a gray image. If the color image is obtained, it must be grayed first. Taking RGB color image as an example, the general graying formula is:

$$\text{Gray } = 0.299\text{R} + 0.587\text{G} + 0.114 \text{ B} \tag{1}$$

2.1.2 *Filtering and noise reduction*

The perfect image information is noise-free and the image quality is very good, but in reality, due to various reasons such as acquisition equipment and environmental interference, the collected image information contains a lot of noise information. The most common noise is salt and pepper noise and Gaussian noise.

Canny operator is an edge detection method that comprehensively seeks the best compromise between anti-noise interference and accurate positioning. Gaussian filtering is generally used to remove noise. The following is a common 3x3 convolution kernel template:

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix} \tag{2}$$

Gaussian filtering can filter out the noise part of the image, so as to avoid mistakenly identifying the wrong noise information as the edge in the later edge detection.

The dimension of the filter kernel should not be too large, otherwise the edge information may be smoothed out, so that the edge detection operator cannot correctly identify the edge information.

2.1.3 *Differential calculation amplitude and direction*

Using the first-order finite difference to calculate the gradient, two matrices of the partial derivatives of the image in the X and Y directions can be obtained. Sobel operator is used as the gradient operator in Canny operator. The following is an example of Sobel operator to calculate the amplitude and direction of gradient:

x direction:

$$Sy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

y direction:

$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4}$$

Preset H (x, y) as the calculated image:

$$H(i,j) = \begin{bmatrix} A0 & A1 & A2 \\ A3 & C & A5 \\ A6 & A7 & A8 \end{bmatrix} \tag{5}$$

Point C (I, J) is the gradient to be calculated.

Y-direction gradient:

$$Gy = 2 \times A7 + A6 + A8 - (2 \times A1 + A0 + A2) \tag{6}$$

X-direction gradient;

$$Gx = 2 \times A5 + A2 + A8 - (2 \times A3 + A0 + A6) \tag{7}$$

Gradient amplitude of point C:

$$G_{C(i,j)} = \sqrt{Gx^2 + Gy^2} \tag{8}$$

Gradient direction of point C

$$\theta = \text{arctan}(Gy/Gx) \tag{9}$$

2.1.4 *Non maximum suppression*

Suppression of non maximum data can also be understood as excluding the possibility that non maximum data is an edge. 8. The larger the element value in the image gradient amplitude matrix in the neighborhood, the larger the gradient value of the point in the image. Combined with the gradient direction of the detection point, the approximate edge information can be located.

Non maximum suppression has two characteristics:

• the gradient value of the current position is compared with the gradient value on both sides in the gradient direction

• the gradient direction is perpendicular to the edge direction

The gradient direction contains multiple gradient values at the same time, so it is necessary to linearly interpolate the gradient values on both sides of the gradient direction to obtain the interpolation coefficient β There are the following requirements: the closer the gradient value to the gradient direction, the greater its proportion.

After the non maximum suppression is completed, a binary image will be obtained. The gray values of non edge points are 0, and the gray values of possible edge points are 255. Such a detection result still contains many false edges caused by noise and other reasons, which also needs double threshold filtering.

2.1.5 *Hysteresis threshold*

Double threshold is used to filter the binary image. By selecting the appropriate large threshold and small threshold, the edge image closest to the real edge of the image can be obtained.

The specific implementation method is as follows: an edge image is obtained according to the high threshold value. Such an image contains few false edges. However, due to the high threshold, the generated image edges may not be closed. Another low threshold is used to solve this problem.

In the high threshold image, the edge is linked into a contour. When reaching the endpoint of the contour, the algorithm will find the point satisfying the low threshold in the 8 neighborhood points of the breakpoint, and then collect new edges according to this point until the whole image edge is closed.

2.2 The basic steps of the algorithm

1)nitialization: Assumes that the particle size is N, the dimension is set to 2, and Initialize to a grayscale value in the 0 to 255 interval, and initially set the speed and set the parameters of the algorithm.

2) Enter the image, turn the image into a grayscale image, and filter the image with median and Gaussian smoothing. (If the image is a complex lighting condition, add a histogram equalization operation to the image)

3) Calculate the 2D histogram of the image and the probability distribution.

4) If the number of iterations t is greater than the specified threshold T, turn to step 7), otherwise turn to step 5).

5)Calculate the fitness value of each particle in the particle according to equation 10,and calculate the temporary position of each particle after the t+1st iteration according to equation 11 12,turn to 6.

$$\operatorname{tr}\left(S_{(s^*,t^*)}\right) = \max_{0\leqslant\ s\ t\leqslant L^{-1}}\left(\operatorname{tr}\left(S_{(s,t)}\right)\right) \tag{10}$$

$$\begin{aligned} v_{ij}(t+1) &= {}_{w} \times v_{ij}(t) + c_1 \times_{r_{1j}(t)} \times (p_{ij}(t) - \\ x_{ij}(t)) &+ {}_{c_2} \times {}_{r_{2j}} \times (p_{gj}(t) - x_{ij}(t)) \end{aligned} \tag{11}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{12}$$

6) Copy the particles with the greatest particle group adaptation, and use roulette algorithm to select the particle group, get a new particle group, turn to step 7)

7) The particle group is mutated and the final position of the t+1st generation particle group is obtained.Updates the current optimal position of each particle and the optimal position of the entire particle group, going back to step 4).

2.3 Histogram equalization

The purpose of equalization processing is to obtain an image with a higher contrast than the original image that extends the dynamic range.

2.4 Principle Suppose

The grayscale level is normalized to a continuous amount within the range of 0,1, and that pr(r) represents the probability density function of the grayscale level in a given image. The following is done for the input grayscale level to get the output grayscale S:

$$S = T(r) = \int_0^r p_r(w)dw \tag{13}$$

The resulting probability density function of the output grayscale level is uniform. By integrating the grayscale probability density function of a given image, a new output grayscale is obtained. The new image grayscale level obtained by this change is more balanced in the range of 0,1.

2.5 Gaussian smooth filtering of the image

2.5.1 *Gaussian smoothing and filtering*

The entire picture transition is smoothed evenly by Gaussian smoothing, removing details and filtering out noise.

2.5.2 *Introduction to Gaussian Smoothing Filter*

Gaussian smoothing filters are used to blur images, similar to mean filters, but unlike mean filters, the nucleus is different. The core of the mean filter is equal to each value, while the number in the core of the Gaussian smooth filter is Gaussian. For the 2D Gaussian distribution:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{14}$$

2.6 Median filtering

2.6.1 *Definition*

The median filter is a nonlinear signal processing method, so it is a nonlinear filter and a statistical sort filter. It sets the grayscale value for each pixel to the median of all pixel grayscale values within a neighborhood window at that point.

2.6.2 *Purpose*

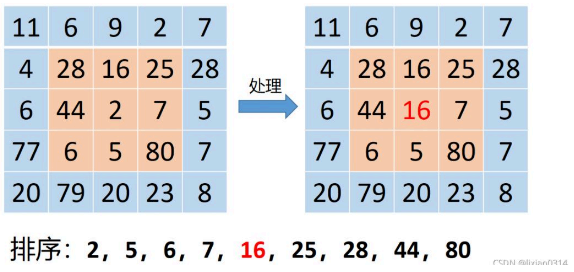
The median filtering has a good filtering effect on isolated noise pixels such as pepper and salt noise and pulse noise, which can maintain the edge characteristics of the image and not cause significant blurring of the image.

2.6.3 *Keystone*

The median filtering of the basic principle/idea is to replace the value of a point in a digital image or sequence of numbers with the median value of each point in a neighborhood of that point, so that the surrounding pixel values are close to the true value, thus eliminating isolated noise points. For example:

2.7 The specific operation process

1. Use a odd number of points to move the window, and the center of the template and a pixel position in the figure coincide; 2. Read the corresponding pixel grayscale values under the template



/ from small to large, from large to small can be; 3. Select the grayscale value of one pixel in the middle of the grayscale sequence; 4. Assign the median value to the pixel at the center of the template.

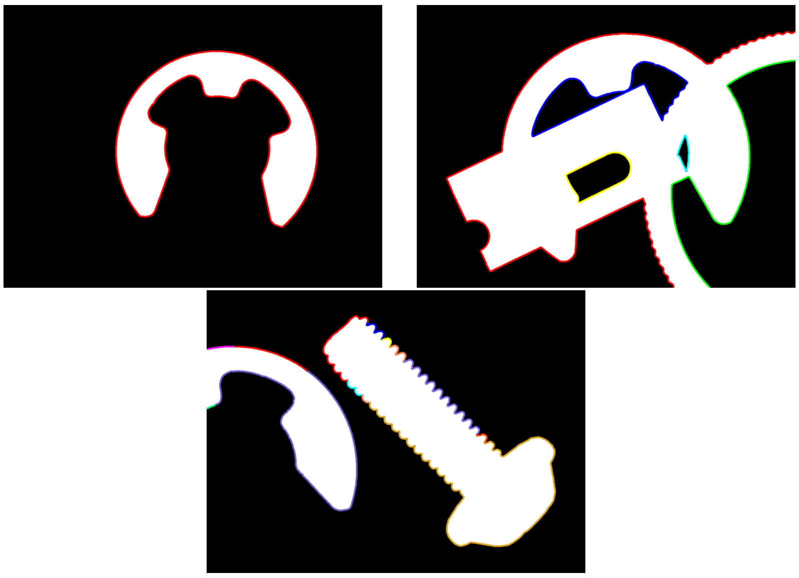


Figure 1 The result of question one

Pic1_2.bmp Edge Contours Output								
Total Edge Contours Count:			5		Total Edge Contours Length:			
Edge Contour 1		Edge Contour 2		Edge Contour 3		Edge Contour 4		Edg
Length	2952.13	Length	1439.95	Length	1502.8	Length	265.813	Length
PointCount	2995	PointCount	1442	PointCount	1504	PointCount	257	PointC

III. Model building and solution of question two

3.1 Calibrate the calibration plate photos

3.1.1 *Homography from calibration plane to image plane*

Because Zhang's calibration is a calibration based on plane checkerboard, if you want to understand Zhang's calibration, you should first start with the homology mapping of two planes. In computer vision, it is defined as the projection mapping from one plane to another. First, let's look at the homography between the image plane and the checkerboard plane of the calibration object.

$$s\tilde{\mathbf{m}} = \mathbf{A}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}\widetilde{\mathrm{M}} \tag{15}$$

The homogeneous coordinates of M represent the pixel coordinates (U, V, 1) of the image plane, and the homogeneous coordinates of M represent the coordinate points (x, y, Z, 1) of the world coordinate system.R represents the rotation matrix, T represents the translation matrix, and s represents the scale factor. A represents the internal parameters of the camera, and the specific expression is as follows:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

α= f/dx，β= F / Dy, because the pixels are not square, γ Represents the scale deviation of pixels in the X and Y directions. Because the calibration object is a plane, we can construct the world coordinate system on the plane of Z = 0. Then the homography calculation is carried out. Let z = 0 to convert the above formula to the following form:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{17}$$

Note H = a [R1 R2 t],now there are:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathrm{H}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{18}$$

H is a three 3 * 3 matrix, and one element is used as homogeneous coordinates. Therefore, h has 8 unknowns to be solved. (x, y) as the coordinates of the calibration object, which can be manually controlled by the designer, is a known quantity. (U, V) are pixel coordinates that can be obtained directly from the camera. For a set of corresponding (x, y) - à (U, V), two sets of equations can be obtained. Now there are eight unknowns to be solved, so at least eight equations are needed. So we need four corresponding points. The homography matrix H from the image plane to the world plane can be calculated by four points. This is also one reason why Zhang's calibration uses the chessboard with four corners as the calibration object. Here, the homography matrix can be written in the form of three column vectors, namely:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} \tag{19}$$

3.1.2 *Solving internal parameter matrix A with constraints*

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda\mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{20}$$

Through the above formula, R1 and R2 can be replaced by the combination of H1, H2 which is R1 = h1a-1, R2 = h2a-1. According to the two constraints, the following two formulas can be obtained:

$$\begin{aligned} &\mathbf{h}_1^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2 = 0 \\ &\mathbf{h}_1^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_1 = \mathbf{h}_2^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2 \end{aligned} \tag{21}$$

$$\mathbf{B} = \mathbf{A}^T\mathbf{A}^1 = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma-u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2}+\frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{v_0}{\beta^2} \\ \frac{v_0\gamma-u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma-u_0\beta)}{\alpha^2\beta^2}-\frac{u_0}{\beta^2} & \frac{(v_0\gamma-u_0\beta)^2}{\alpha^2\beta^2}+\frac{v_0^2}{\beta^2}+1 \end{bmatrix} \tag{22}$$

Now B is a symmetric matrix, so there are only six effective elements of B.

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \tag{23}$$

The next step is pure mathematical simplification:

$$\mathbf{h}_i^T\mathbf{B}\mathbf{h}_j = \mathbf{v}_{ij}^T\mathbf{b} \tag{24}$$

It can be calculated as follows:

$$\begin{aligned} \mathbf{v}_{ij} = &[h_{i1}h_{j1}, h_{i1}h_{j2}+h_{i2}h_{j1}, h_{i2}h_{j2}, \\ &h_{i3}h_{j1}+h_{i1}h_{j3}, h_{i3}h_{j2}+h_{i2}h_{j3}, h_{i3}h_{j3}]^T \end{aligned} \tag{25}$$

The following equations can be obtained by using the constraints:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11}-\mathbf{v}_{22})^T \end{bmatrix}\mathbf{b} = \mathbf{0} \tag{26}$$

3.1.3 *Estimation of external parameter array based on internal parameter array*

Then, for the external parameter matrix, it can be solved by the following formula:

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda\mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{27}$$

By simplifying the above formula, we can get:

$$\begin{aligned} \mathbf{r}_1 &= \lambda\mathbf{A}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda\mathbf{A}^{-1}\mathbf{h}_2 \quad 其中 \quad \lambda = 1/\left\|\mathbf{A}^{-1}\mathbf{h}_1\right\| = 1/\left\|\mathbf{A}^{-1}\mathbf{h}_2\right\| \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda\mathbf{A}^{-1}\mathbf{h}_3 \end{aligned} \tag{28}$$

Maximum likelihood estimation is a method to estimate the total unknown parameters. It is mainly used for point estimation problems. The so-called point estimation refers to using the observed value of an estimator to estimate the true value of an unknown parameter.it is to select the parameter in the parameter space that maximizes the probability of the sample obtaining the observed value.

Define likelihood function:

The overall distribution is discrete: (P is the known distribution law)

$$L(\theta_1, \theta_2, \theta_3, \ldots, \theta_k) = \prod_{i=1}^{n} p(x_i; \theta_1, \theta_2, \theta_3, \ldots, \theta_k) \tag{29}$$

The overall distribution is continuous: (F is the probability density function)

$$L\left(\theta_1, \theta_2, \theta_3, \ldots, \theta_k\right)=\prod_{i=1}^n f\left(x_i ; \theta_1, \theta_2, \theta_3, \ldots, \theta_k\right) \tag{30}$$

The maximum likelihood method is to select the estimated value of the unknown parameter that maximizes L within the desirable range

$$\theta_1, \theta_2, \theta_3, \ldots, \theta_k$$

. It is assumed that the corners on the image are disturbed by noise, and these noises are considered as Gaussian noise. Then the amplitude of noise is the error caused to the observation value. However, the probability density of Gaussian noise is known, so the idea of maximum likelihood estimation can be used to ”guess” the true value.

Then we need to construct a likelihood function and find its maximum value:

$$\sum_{i=1}^n \sum_{j=1}^m\left\|\mathbf{m}_{ij}-\hat{\mathbf{m}}\left(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j\right)\right\|^2 \tag{31}$$

When this formula obtains the minimum value, it is the maximum likelihood estimate of the parameter.

Let the noise near the corner obey Gaussian distribution

Then: the sample value of corner mij follows the following probability density function:

$$f\left(m_{ij}\right)=\frac{1}{\sqrt{2 \pi}} e^{\frac{-\left(m\left(A, R_1, t_1, M_t\right)-m_{+}\right)}{b^2}} \tag{32}$$

Now construct the likelihood function:

$$L\left(A, R_i, t_i, M_j\right)=\prod_{i=j=1}^{n, m} f\left(m_{ij}\right)=\frac{1}{\sqrt{2 \pi}} e^{\frac{-\sum_{i=1}^n \sum_{j=1}^m\left(m\left(A, R_i, t_i, M_j\right)-m_j\right)}{\left.e^2\right)}} \tag{33}$$

Now let l get the maximum value, then the following formula can be minimized:

$$\sum_{i=1}^n \sum_{j=1}^m\left\|\mathbf{m}_{ij}-\hat{\mathbf{m}}\left(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j\right)\right\|^2 \tag{34}$$

3.2 Correct the distortion of the calibration image

With such a simple formula, distortion correction is completed:

$$\begin{aligned} & u=\breve{u}+\left(\breve{u}-u_0\right)\left[k_1\left(x^2+y^2\right)+k_2\left(x^2+y^2\right)^2\right] \\ & v=\breve{v}+\left(\breve{\vartheta}-v_0\right)\left[k_1\left(x^2+y^2\right)+k_2\left(x^2+y^2\right)^2\right]\end{aligned} \tag{35}$$

Represents the pixel coordinates after distortion correction, and (u,v) represents the pixel coordinates of the image in the case of actual radial image distortion.

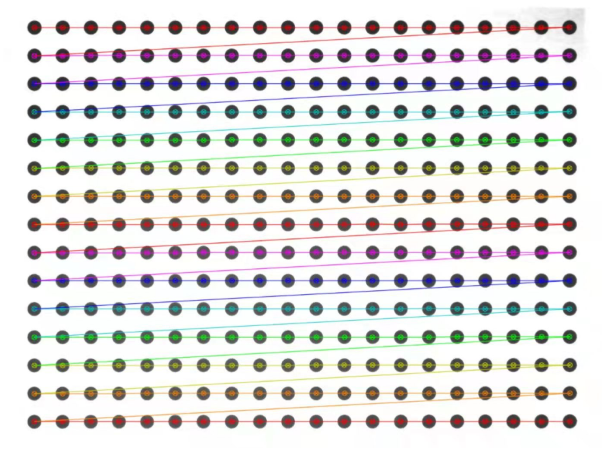


Figure 2 Calibration plate center anchor chart

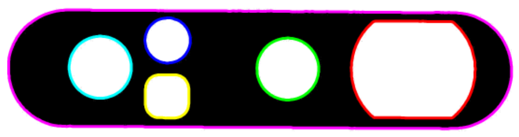


Figure 3 The result of correction

3.3 Calculate the ratio of physical distance to pixel distance

3.4 The pixel length of the image edge contour is calculated according to the first algorithm

3.5 The physical length of the image edge contour is converted according to the ratio

IV. Model building and solution of question three

4.1 The extraction of feature points

Calculate the size of the support area for each data point,the number of times it serves as the endpoint of the support area, and the curvature value, if one attribute is a bureau part extremum,

Total Edge Contours Count:			6			Total Edge Contours Length(mm):	
Edge Contour 1		Edge Contour 2		Edge Contour 3		Edge Contour 4	
Length(mm)	83.1874	Length(mm)	14.2606	Length(mm)	10.3969	Length(mm)	10.8544

which can be judged as a feature.

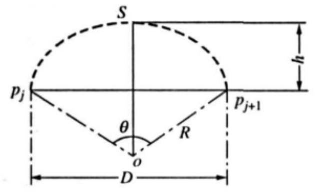
4.2 The judgment of segment linear

(1) Calculate the ratio of the sum of lengths between segmented data points and the distance of the straight segment D determined by the two adjacent feature points The value is .

$$L = \sum_{i=1}^{m-1} d\left(p_i p_{i+1}\right), d\left(p^{\cdot} p_{i+1}\right)$$

Where, the European distance between the point p_i and p_{i+1} is represented leave. If λ is less than a given threshold, the line between the two feature points is judged to be a straight line.

(2) The ratio between the length sum of the segmented data points and the arc length S determined by the two adjacent feature points of L is calculated σ. The picture is as the follow: The



start and end points of the arc segment are the feature points p_j and p_{j+1}, respectively, and the string length D is the straight line distance between the two feature points leave. Set h to string height,which is the maximum distance from the segmented data point to the straight segment $p_j p_{j+1}$.

$$\begin{cases} R = \frac{(D/2)^2 + h^2}{2h} \\ \theta = 4 \arctan(2h/D) \end{cases} \tag{36}$$

If the σ is less than the given threshold, the curve segment type determined by the segmented data point is an arc.

(3) Calculate the maximum absolute error ε. If the type of curve segment determined by the segmented data is a straight line, the two adjacent feature points are true The equation for the fixed line l is: Ax+By+c=0, and the ε is the maximum distance from each point to l.

$$\varepsilon_{\max} = \max_{i \in (1,\ m)} \left\{ \frac{\left| Ax_i + By_i + C \right|}{\sqrt{A^2 + B^2}} \right\} \tag{37}$$

If the segment type determined by the segment data is an arc, the maximum distance from each point to the arc is ε.

$$\varepsilon_{\mathrm{amax}} = \max_{i \in (1,\ m)} \left\{ \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - B \right\}. \tag{38}$$

(4) Calculate fitting error E. If the type of curve segment determined by the segmented data is a straight line , the fitting error is:

$$E_1 = \sum_{i=1}^{m_1} \frac{Ax_i + By_i + C}{\sqrt{A^2 + B^2}} \tag{39}$$

If the curve segment type determined by the segment data is an arc, the fitting error is:

$$E_a = \sum_{i=1}^{\prime\prime\prime} \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R \tag{40}$$

4.3 Accurate judgment of the line

(1) First of all,the value of λ in each segment data point are calculated. If the value of λ is less than the given threshold, the segment data point is initially determined on a straight line.So,it fits a straight line. The maximum absolute error of εlmax and the fitting error El are then calculated separately. Both errors should be less than the given tolerance.

(2) If the value of λ in a segmented data point is greater than the given threshold, it is counted σ value.If the σ is less than a given threshold, the segment data point is initially determined to be on the arc and is rounded to fit it. The maximum absolute error of εlmax and the fitting error El are then calculated separately. Both errors should be less than the given tolerance.

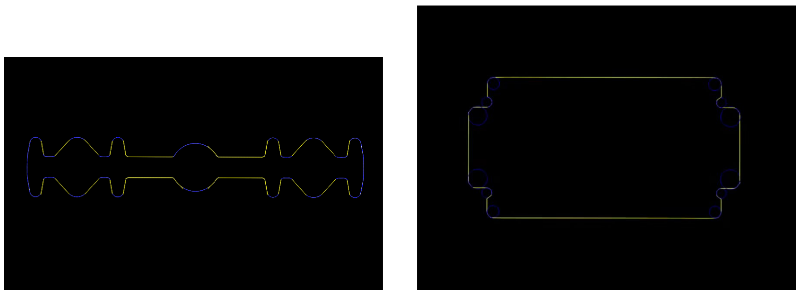


Figure 4 The results of question three

V. Future Work

(1)For the first 1 / 10 sub-pixel edge extraction algorithm, our algorithm is well adapted to simple light sources. However, in the face of overexposed distorted images under complex light sources, the algorithm is difficult to distinguish the highlight distortion part, and the highlight part needs to be preprocessed to better realize edge segmentation.

(2)In the later stage, we intend to improve the universality and robustness of the algorithm. We need to find a suitable and universal image processing algorithm to realize the expectation of processing all pictures with the same algorithm. For the image calibration of the second question, through the efficient application of the given data, the real parameters are accurately restored, and the accurate restoration and high-precision measurement of the workpiece image are realized. However, although our algorithm can compensate for lens distortion and calculate complex image models, the computational cost is high.

(3)For the third question of segmentation by primitives, our algorithm can better realize the edge fitting through primitives such as line segments, ellipses and circles. However, in terms of threshold selection, there is no universal problem. If the threshold is not selected properly, it is easy to realize under fitting or over fitting. In the later stage, the model will be improved for the above problems.

VI. References

[1] Author, Title, Place of Publication: Press, Year of publication.

[2] author, paper name, magazine name, volume number: starting and ending page number, year of publication.

[3] author, resource title, web site, visit time (year, month, day).

VII. Appendix

Listing 1: main1.m (Matlab)

```
clc;
clear;
I = imread('Pic2_4.bmp');
figure(1),imshow(I),title('原圖');
% 中值濾波
I = median_filter(I,3);
%I = histeq(I);
Pso_no=20;   %设种群数量为20个
Max_iteration=10;    %最大迭代次数为
lb=0; ub=255;          %最小阈值为0, 最大阈值为255
% 对图像进行高斯平滑滤波, 濾波器3*3, 標準差0.5
% 抑制服从正态分布的噪声
g_3 = fspecial('gaussian',3,0.5);
I_v = imfilter(I,g_3);
[m,n] = size(I);
I_c=zeros(m,n);
for i=1:m
for j=1:n
I_c(i,j)=abs(I(i,j)-I_v(i,j));
end
end
figure(2),imshow(I_c),title('高斯濾波');
% 粒子群優化
[position_best1,position_best2,D] = PSO(Pso_no,Max_iteration,lb,ub,I,I_c);
I_z = ones(m,n);
for i=1:m
for j=1:n
if I_c(i,j)<position_best2 && I(i,j)<position_best1
I_z(i,j)=0;
end
end
end
I_z = I_z;
% 填洞操作
%J = imfill(I_z,'holes');
imwrite(I_z,'bi21.jpg');
figure(3),imshow(I_z),title('二维分割');
figure(4);
semilogy(D,'Color','r')
xlabel('灰度值');
ylabel('类间方差');
x=1:10;
plot(x,D,'*-');
xlabel('迭代次数');
ylabel('类间方差');
```

Listing 2: madianfilter.m

```
function [ img ] = median_filter( image, m )
%----------------------------------------------
%中值滤波
%输入:
%image: 原图
%m: 模板的大小3*3的模板, m=3

%输出:
%img: 中值滤波处理后的图像
%----------------------------------------------
n = m;
[ height, width ] = size(image);
x1 = double(image);
x2 = x1;
for i = 1: height-n+1
for j = 1:width-n+1
mb = x1( i:(i+n-1),  j:(j+n-1) );
mb = mb(:);
mm = median(mb);
x2( i+(n-1)/2,  j+(n-1)/2 ) = mm;
end
end
img = uint8(x2);
end
```

Listing 3: fillsmallholes.m

```
function new=fillsmallholes(bw,threshold)
% Fill small holes in the binary image bw using the threshold, only holes
% with areas smaller than threshold will be filled

filled = imfill(bw, 'holes');

holes = filled & ~bw;

bigholes = bwareaopen(holes, threshold);

smallholes = holes & ~bigholes;

new = bw | smallholes;
intialize_position_rand.m (Matlab)
function Positions=intialize_position_rand(SearchAgents_no,ub,lb)

Positions=rand(1,SearchAgents_no).*(ub-lb)+lb;
Positions=double(int32(Positions));
end
```

Listing 4: intializepositionrand.m

```
  function Positions=intialize_position_rand(SearchAgents_no,ub,lb)

  Positions=rand(1,SearchAgents_no).*(ub-lb)+lb;
  Positions=double(int32(Positions));
  end
```

Listing 5: Otsuh.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%最大类间方差计算%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Leader_pos,y,d]=Otsu_h(ua11,w11,Positions,p,nd,u)
ua=[];w=[];
n=size(Positions,2);
for i=1:n
flag=Positions(i)>p&&Positions(i)<=nd;
if flag
ua=[ua,ua11(Positions(i)-p+1)];
w=[w,w11(Positions(i)-p+1)];
else
ua(i)=0+eps;
w(i)=0+eps;
end
end
d=(w./(1-w)).*(ua./w-u).^2;
[y,tp]=max(d);
Leader_pos=Positions(tp);
end
```

Listing 6: Otsuq.m

```
function [ua11,w11,p,nd,u]=Otsu_q(a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%最大类间方差改进（适应性函数）%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[m,n]=size(a);
N=m*n;
L=256;
a=double(a);
for i=1:L
count(i)=length(find(a==(i-1)));
f(i)=count(i)/(N);
end

for i=1:L
if count(i)~=0
```

```
st=i-1;
break;
end
end
for i=L:-1:1
if count(i)~=0
nd=i-1;
break;
end
end
p=st;
q=nd-st+1;
u=0;
w11=[];
ua11 = [];
for i=1:q
u=u+f(p+i)*(p+i-1);
ua11 = [ua11,u];
w11(i)=sum(f(1+p:i+p));
end
% w11=w11+eps;
```

Listing 7: PSO.m

```
function [G_best1,G_best2,D] = PSO(Pso_no,Max_iteration,lb,ub,I,image_c)
% Pso_no: 种群数量
% Max_iteration: 最大迭代次数
% lb、ub: 最小阈值、最大阈值
% I: 原始圖像
% image_c: 高斯濾波後圖像
L_a=max(max(image_c));
L_b=min(min(image_c));
V_max=4;
Leader_score1=-inf; %change this to -inf for maximization problems
Leader_score2=-inf;
%统计图像整体灰度平均值u, 背景灰度平均值u11, 背景概率w11
[ua11,w11,pp1,nd1,u1]=Otsu_q(I);
%统计图像整体灰度平均值u, 背景灰度平均值u11, 背景概率w11
[ua22,w22,pp2,nd2,u2]=Otsu_q(image_c);
%%%%粒子群位置及速度的初始化
Positions1=intialize_position_rand(Pso_no,ub,lb);%粒子位置初始化
Positions2=intialize_position_rand(Pso_no,L_a,L_b);%粒子位置初始化
V_pso2=intialize_position_rand(Pso_no,2-V_max,V_max-2);%粒子位置初始化
V_pso1=intialize_position_rand(Pso_no,-V_max,V_max);%粒子位置初始化
D=[];t=0;
P_best1=zeros(1,Pso_no);
P_best2=zeros(1,Pso_no);
d111=zeros(1,Pso_no);
d222=zeros(1,Pso_no);
```

```
while t<Max_iteration
for i=1:Pso_no
%粒子位置重置
if Positions1(i)>ub     %粒子位置超出上界设置为255
Positions1(i)=255;
end
if Positions1(i)<lb     %粒子位置超出下界设置为0
Positions1(i)=0;
end
% Return back the search agents that go beyond the boundaries of the search space
if Positions2(i)>ub     %粒子位置超出上界设置为255
Positions2(i)=255;
end

if Positions2(i)<lb     %粒子位置超出下界设置为0
Positions2(i)=0;
end
%粒子速度重置
if V_pso1(i)>V_max     %粒子速度超出上界设置为V_max
V_pso1(i)=V_max;
end

if V_pso1(i)<-V_max     %粒子速度超出下界设置为-V_max
V_pso1(i)=-V_max;
end
if V_pso2(i)>V_max-2     %粒子速度超出上界设置为V_max-2
V_pso2(i)=V_max-2;
end
if V_pso2(i)<2-V_max     %粒子速度超出下界设置为2-V_max
V_pso2(i)=2-V_max;
end
end
%由最大类间方差公式计算每个位置（灰度值）的类间方差
[Max_jubu1,d1,d11]=Otsu_h(ua11,w11,Positions1,pp1,nd1,u1);
[Max_jubu2,d2,d22]=Otsu_h(ua22,w22,Positions2,pp2,nd2,u2);
%全体最优位置
if d1>Leader_score1
Leader_score1=d1;
G_best1=Max_jubu1;
end
if d2>Leader_score2
Leader_score2=d2;
G_best2=Max_jubu2;
end
for i=1:Pso_no
if d11(i)>d111(i)
d111(i)=d11(i);
P_best1(i) =Positions1(i);
end
```

```
if d22(i)>d222(i)
d222(i)=d22(i);
P_best2(i) =Positions2(i);
end
end
%给各个系数赋值
w=0.5;
r1=rand;      r2=rand;
c1=2;         c2=2;
for i=1:Pso_no
V_pso1(i)=w*V_pso1(i)+c1*r1*(P_best1(i)-Positions1(i))+c2*r2*(G_best1-Positions1(i));%改变粒子的速度
V_pso2(i)=w*V_pso2(i)+c1*r1*(P_best2(i)-Positions2(i))+c2*r2*(G_best2-Positions2(i));
Positions1(i)=Positions1(i)+V_pso1(i);%改变粒子的位置
Positions2(i)=Positions2(i)+V_pso2(i);
end
Positions1=double(int32(Positions1));
Positions2=double(int32(Positions2));
D=[D,G_best1];
t=t+1;
end
end
```

Listing 8: Annex1.hdev (Halcon)

```
*读取图片
dev_update_off ()
dev_close_window ()
dev_open_window (0, 0, 650, 485, 'black', WindowHandle)
read_image (Image, './bi3.bmp')
get_image_size (Image, Width, Height)
dev_set_part (0, 0, Height-1, Width-1)
edges_sub_pix (Image, ContoursSplit, 'canny', 1, 20, 40)
count_obj (ContoursSplit, Number)
dev_set_line_width (3)
dev_display (Image)
dev_display (ContoursSplit)

*写入轮廓点数据
open_file ('轮廓数据3.txt', 'append', FileHandle)
for I := 1 to Number by 1
corx := []
cory := []
*选择一个元素
select_obj (ContoursSplit, ObjectSelected, I)
length_xld(ObjectSelected, len)
*获取轮廓点的个数
contour_point_num_xld(ObjectSelected, num_1)
*获取轮廓点坐标
get_contour_xld(ObjectSelected, Row1, Col1)
```

```
tuple_abs (Row1, Row1)
tuple_abs (Col1, Col1)
fwrite_string (FileHandle, '轮廓'+I+'\n'+'Length '+len+'\n'+'PointCount '+num_1+'\n')
fwrite_string (FileHandle,'X Y\n')
fwrite_string (FileHandle, Col1+' '+Row1+'\n')
endfor

close_file (FileHandle)
*Pic1_1.bmp Edge Contours Output
```

Listing 9: Annex2.cpp (C++/OpenCV)

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2\core.hpp>
#include <opencv2\features2d.hpp>
#include <opencv2\highgui.hpp>
#include <opencv2\imgproc.hpp>
#include <opencv2\imgproc\types_c.h>
#include <opencv2\calib3d.hpp>

#include <iostream>
#include <vector>
#include <string>
#include <fstream>

#include <direct.h>
#include <io.h>

using namespace cv;
using namespace std;
int main() {
   vector<cv::Point2f> image_points;
   vector<std::vector<cv::Point2f>> image_points_seq;
   SimpleBlobDetector::Params params;
   params.maxArea = 10e4;
   params.minArea = 10;
   params.filterByArea = true;
   Ptr<FeatureDetector> blobDetector = SimpleBlobDetector::create(params);

   Mat img = imread("F://2021 APMCM Problems//2021 APMCM Problem A//Annex
       2//Pic2_1_t.bmp", 0);
   int aqXnum = 20;
   int aqYnum = 15;
   bool found = findCirclesGrid(img, Size(aqXnum, aqYnum), image_points,
       CALIB_CB_SYMMETRIC_GRID | CALIB_CB_CLUSTERING, blobDetector);
   image_points_seq.push_back(image_points);
   Mat cimg;
```

```
  cvtColor(img, cimg, COLOR_GRAY2BGR);
  cv::Size square_size = cv::Size(10, 10);
  std::vector<std::vector<cv::Point3f>> object_points;
  cv::Mat cameraMatrix = cv::Mat(3, 3, CV_32FC1, cv::Scalar::all(0));
  cv::Mat distCoeffs = cv::Mat(1, 5, CV_32FC1, cv::Scalar::all(0));
  std::vector<cv::Mat> tvecsMat;
  std::vector<cv::Mat> rvecsMat;

  std::vector<cv::Point3f> realPoint;
  for (int i = 0; i < aqYnum; i++) {
    for (int j = 0; j < aqXnum; j++) {
      cv::Point3f tempPoint;
      tempPoint.x = i * square_size.width;
      tempPoint.y = j * square_size.height;
      tempPoint.z = 0;
      realPoint.push_back(tempPoint);
    }
  }
  object_points.push_back(realPoint);
  calibrateCamera(object_points, image_points_seq, img.size(), cameraMatrix,
      distCoeffs, rvecsMat, tvecsMat);
  std::cout << "tvecsMat:\n" << tvecsMat[0] << std::endl;
  std::cout << "rvecsMat:\n" << rvecsMat[0] << std::endl;

  std::cout << "#cameraMatrix:\n" << cameraMatrix << std::endl;
  std::cout << "#distCoeffs:\n" << distCoeffs << std::endl;
  cv::Mat cb_final;
  cv::Mat mapx = cv::Mat(img.size(), CV_32FC1);
  cv::Mat mapy = cv::Mat(img.size(), CV_32FC1);
  cv::Mat R = cv::Mat::eye(3, 3, CV_32F);
  //initUndistortRectifyMap(cameraMatrix, distCoeffs, R, cv::Mat(), cb_source.size(),
      CV_32FC1,
  //                    mapx, mapy);
  //cv::remap(cb_source, cb_final, mapx, mapy, cv::INTER_LINEAR);

  undistort(img, cb_final, cameraMatrix, distCoeffs);

  cv::imwrite("F://2021 APMCM Problems//2021 APMCM Problem A//Annex
      2//cb_final_1.png", cb_final);

}
```

Listing 10: Annex3.hdev (Halcon)

```
open_file('EdgeContour2.csv','input', FileHandle)

x1:=[]
y1:=[]
x2:=[]
```

```
y2:=[]

IsEOF1:=0
while(IsEOF1==0)
fread_line(FileHandle, OutLine, IsEOF1)

tuple_split(OutLine, ', \n', Substrings)
tuple_length(Substrings, Length)

if(Length!=0)
x1:=[x1,Substrings[0]]
y1:=[y1,Substrings[1]]

endif

endwhile
dev_set_color('red')
dev_set_draw('margin')
tuple_number(x1,row1)
tuple_number(y1,col1)

dev_update_window('off')
dev_close_window()
dev_open_window(0,0,1000,1000,'black',WindowID)
gen_contour_polygon_xld(Contours,row1,col1)
segment_contours_xld (Contours, ContoursSplit, 'lines_ellipses', 20, 7, 3)
* sort_contours_xld (ContoursSplit, SortedContours, 'upper_left', 'true', 'column')
* dev_display(Contours)
count_obj(ContoursSplit,NumSegments)
* dev_display(ContoursSplit)
NumCirCles :=0
NumLines :=0
open_file ('轮廓数据4.txt', 'append', FileHandle)
for i :=1 to NumSegments by 1
select_obj(ContoursSplit,SingleSegment,i)
get_contour_global_attrib_xld(SingleSegment,'cont_approx',Attrib)
length_xld(SingleSegment, len)
contour_point_num_xld(SingleSegment, num_1)
get_contour_xld(SingleSegment, Row1, Col1)
if(Attrib = 1)
fit_circle_contour_xld (SingleSegment, 'atukey', -1, 0, 0, 3, 2, Row, Column, Radius,
    StartPhi, EndPhi, PointOrder)
gen_ellipse_contour_xld (ContEllipse, Row, Column, 0, Radius, Radius, 0, 6.28318,
    'positive', 1.5)
dev_set_color('blue')
dev_display(ContEllipse)
set_tposition(WindowID,Row,Column)
tuple_sub(EndPhi*57.3,StartPhi*57.3,Phi_sub)
tuple_length(Col1,LC)
tuple_length(Row1,LR)
fwrite_string (FileHandle, 'S'+i+' CircularArc '+'('+Row+','+Column+')'+'
    ('+Row1[0]+','+Col1[0]+')'+' ('+Col1[LC-1]+','+Row1[LR-1]+') '+Phi_sub+'\n')
*fwrite_string (FileHandle, Row1+','+Col1+'\n')
elseif(Attrib = -1)
fit_line_contour_xld (SingleSegment, 'tukey', -1, 0, 5, 2, RowBegin, ColBegin, RowEnd,
    ColEnd, Nr, Nc, Dist)
gen_contour_polygon_xld (Line, [RowBegin,RowEnd], [ColBegin,ColEnd])
distance_pp(RowBegin,ColBegin,RowEnd,ColEnd,Length)
dev_set_color('yellow')
dev_display(Line)
set_tposition(WindowID,(RowBegin + RowEnd)/2,(ColBegin +ColEnd)/2)
fwrite_string (FileHandle, 'S'+i+' Line'+' ('+RowBegin+','+ColBegin+')'+'
    ('+RowEnd+','+ColEnd+') '+Length+'\n')
else
fit_ellipse_contour_xld (SingleSegment, 'fitzgibbon', -1, 2, 0, 200, 3, 2.0, \
Row, Column, Phi, Radius1, Radius2, StartPhi, \
EndPhi, PointOrder)
gen_ellipse_contour_xld (ContEllipse, Row, Column, Phi, Radius1, Radius2, \
StartPhi, EndPhi, PointOrder, 1.5)
dev_set_color('blue')
dev_display(ContEllipse)
set_tposition(WindowID,Row,Column)
tuple_sub(EndPhi*57.3,StartPhi*57.3,Phi_sub)
tuple_length(Col1,LC)
tuple_length(Row1,LR)
fwrite_string (FileHandle, 'S'+i+' EllipticArc '+'('+Row+','+Column+')'+'
    ['+Radius1+','+Radius2+']'+ \
' ('+Row1[0]+','+Col1[0]+')'+' ('+Col1[LC-1]+','+Row1[LR-1]+') '+Phi_sub+'
    '+Phi*57.3+'\n')
endif
endfor
close_file (FileHandle)
```