Team Number:	apmcm2106468
Problem Chosen:	A

2021 APMCM summary sheet

Despite the fact that algorithms such as Canny, Sobel could indeed provide some usable edge-detection methods, to reach a higher standard of edge detection under which sub-pixel accuracy as well as global information is also desired, there is still a lot to do. In this work, we applied some relevant algorithms such as least-squares approximation, Deverny algorithm, hyper-parameter optimization, interpolation process, polygon estimation and Gaussian filter, as well as their integrations, to obtain better solution results. All of the algorithms are programmed with C++, python and Matlab.

In the following, we briefly introduce the three proposed questions and explain some of our key analyses based on which the solutions are derived.

Question1: Firstly, we detect the contour of the target image, however, it's full of redundancy points, which may hinder our calculation of the perimeter of the boundary. Hence, we only extract a sub array of points. These points form the vertex of polygon, so we can calculate the perimeter of the polygon to approximate the exact perimeter of the target image.What's more,To raise the accuracy to sub-pixel. We use the Devernay algorithm, which provide a slight revision to each pixel. So we can get a more accurate coordinate of the vertex.To conclude, the contour line detection ensure that we treat the boundary as a whole, while the Devernary algorithm improve the accuracy to sub pixel.

Question2: For that we know the distribution of the calibration board, we can derive the mapping from an undistorted picture to a distorted picture.Thus we can use this mapping to calibrate the product's picture. And through the pixel-distance between the centers of the dots, and its physical distance, we can obtain the scale ratio between pixel space and physical space. Thus, the physical length can be calculated through ratio and length of edge in calibrated picture in pixel space.

Question3: We first sample points and calculate local curvature of each point. Based on the curvature and its differential, we can locate breakpoints to separate different arcs and its type(line segment, circular arc, elliptic arc). After that, we got points sequence of each arc, and through least-square-error optimization algorithm. We could acquire the parameters of fitting curves.

Keywords: Edge Detection Sub-pixel Edge Location and Correction Edge Separation Image Calibration Edge Curvature Estimation High-accuracy Calculation

Contents

I. Introduction

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

1.1 Background

With the development of science and technology, size measuring instruments of digital image are gradually taking the place of traditional manual caliper measurement application in order to improve the measurement accuracy of various workpieces and parts. In general, the key idea is to calibrate the camera in virtue of the dot matrix or checkerboard feature information of the calibrated image to correct the distortion of the image, as well as to map relationship between the image coordinate space and the world coordinate space.

Image edge contains rich intrinsic information attribute for extracting image features in image recognition. Edge detection in digital images is one of the oldest problems in computer vision and continues to be a very active area of research. The most famous method is the one proposed by Canny, a name that has become almost a synonym of edge detection. However, **the Canny algorithm** can barely detect the image edge pixels boundary based on the difference of image gray value, and it can't take the contour as a whole.

To have finer accuracy in the position of the edge points, **sub-pixel correction** was proposed. With a very elegant addition to the original Canny algorithm, the position of an edge point can be refined.

1.2 Restatement of the Problem

This question requires to extract sub-pixel edge with certain pixel accuracy as well as to eliminate the interference effects of edge burrs and shadow parts of the edges. Then with the information and the image of a dot matrix calibration plate given, we are supposed to conduct image rectification analysis of the product image and calculate the actual physical sizes of the edge segmentation fitting curve segments on the product image. After that, we should seek out how to segment edge contour curve automatically and fit the curve data into straight line segments, circular arc segments or elliptical arc segments.

II. Analysis

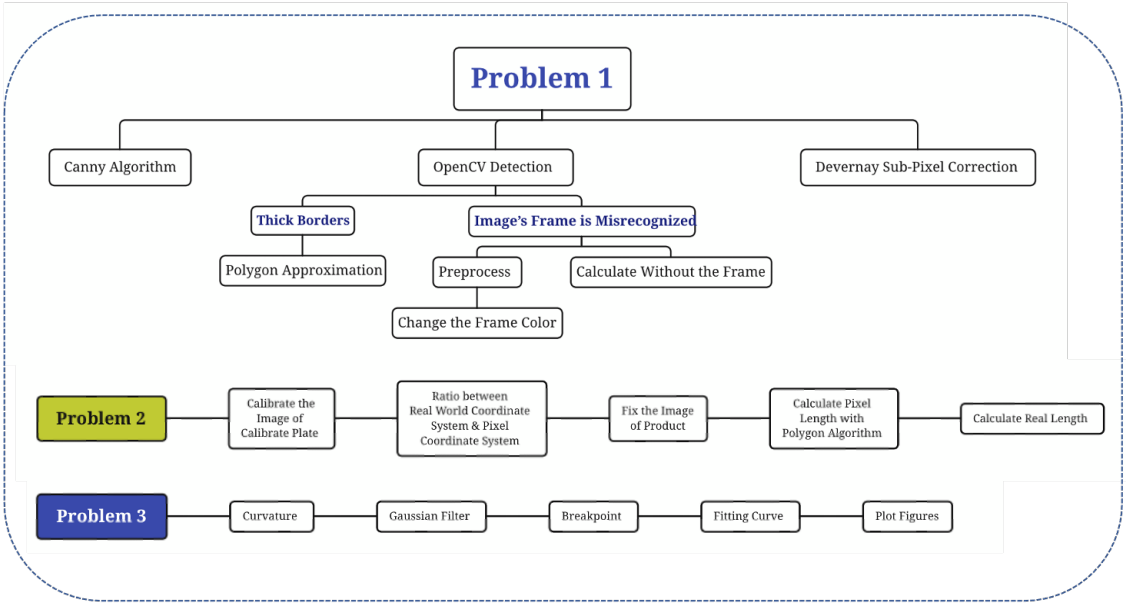

Figure 1 Flow chart of the full text

Claim:we did some pre-processing work on the image, here we first list these here.

- In the problem 1, since some target image is connected with the boundary of the picture. assuming that we implement our algorithm without any pre-processing work, chances are that the picture's boundary will disturb the exact boundary detection.

- In the problem 2, to better illustrate the utility of the our calibration algorithm, we extend the size of the image with some blank pixels, as one can see from the analysis and figure in the page shown after.

III. Assumptions

To conveniently solve the problem, we make the following assumptions:

1. Assuming that the borders of the products are smooth and continuous.
2. Assuming that the width of the borders of product are infinitely small in real world.
3. Assuming that the size of the desired product is large in comparison with the noise

.

IV. Symbol Description

Symbol	Description
P_i	The point on the contour
R_i	The circumscribed circle radius of P_i
Correction()	Correct matrix function relative to C_i
C_i	The radius of curvature of P_i
X_I	Product in **I**mage space
X_I'	Product in **I**mage space. Theimage has been corrected.
X_r	Product in **R**eal space.
η	The offset of the pixel

V. Models and Solutions

5.1 Models and Solution of Question 1

5.1.1 *Canny Algorithm*

Gauss Filter

Gaussian kernel used for Gauss filtering is a **Gaussian function** with both x and y dimension, and the standard deviation on both dimensions is generally the same in the form of:

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

Gauss filtering means convolution of two-dimensional Gaussian kernel with an image. Because the data form of the digital image is a discrete matrix, Gaussian kernel is a discrete approximation of the continuous Gaussian function, which is obtained by discrete sampling and normalization of Gaussian surfaces. For example, Gaussian kernel, which has a size $3 * 3$ and a a standard deviation of 1, is

$$\begin{bmatrix} 0.0751136 & 0.1238414 & 0.0751136 \\ 0.1238414 & 0.2041799 & 0.1238414 \\ 0.0751136 & 0.1238414 & 0.0751136 \end{bmatrix}$$

Algorithm 1 algorithm 1:image gradient (I,S)

Input: An image I, a scale parameter S

Output: gradient vector fild of x coordinate and y coordinate

1: $I_S \leftarrow I \star K_s$ /* convolution with a Gaussian kernel */
2: **for** $(x, y) \in I_S$ **do**
3: $g_x(x, y) \leftarrow I_S(x + 1, y) - I_S(x_1), y$
4: $g_y(x, y) \leftarrow I_S(x, y + 1) - I_S(x, y - 1)$
5: **end for**
6: $g \leftarrow (g_x, g_y)$

Sobel Operator and Image Gradient

Sobel operators are two matrices of size $3 * 3$, $\mathbf{S}_x$ and $\mathbf{S}_y$. The former is used to calculate the pixel gradient matrix in the x direction of the image, and the latter is used

to calculate the pixel gradient matrix in the y direction of the image. The specific form is

$$G_x = S_x \star I = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \star I$$

$$G_y = S_y \star I = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \star I$$

Among them, I is a image matrix, and here $\star$ represents a cross-correlation operation (convolution operation can be regarded as a cross-correlation operation after the convolution kernel is rotated by 180 degree). It should be noted that the origin of the image matrix coordinate system is in the upper left corner, and the positive direction of the x coordinate is from left to right, and the positive direction of the y coordinate is from top to bottom. Then gradient matrix $\mathbf{G}_{xy}$ can be calculated:

$$g_{xy}(i,j) = \sqrt{g_x(i,j)^2 + g_y(i,j)^2}$$

Algorithm 2 algorithm 2:Generate-transform-map (I,g)

Input: An image I,image gradient field g=(g_x, g_y)

Output: tranform map of x and y T=(T_x, T_y)

1: **for** $(x,y) \in I_S$ **do**
2: current=(i,j)
3: near1,near2=two nearest sub-pixel to the current point in the direction of g
4: $\epsilon = \frac{1}{2} \frac{||g(near1)|| - ||g(near2)||}{||g(near1)|| + ||g(near2)|| - 2||g(current)||}$
5: **if** $\epsilon \leq 1$ **then**
6: $(T_x(i,j), T_y(i,j)) = \epsilon * (near2 - current)$ /*Notice that the dividend may be too small, beacuse not all the pixel are boundary*/
7: **end if**
8: **else**$(T_x(i,j), T_y(i,j)) = (0,0)$
9: **end for**

Non-Maximum Suppression

Canny proposed the following scheme to perform **non-maximum suppression** in a 2D grid. As mentioned before, g_{xy}, g_x, g_y respectively denote the norm, x-component and y-component of the image gradient $\nabla(G \star I)$.Suppose we want to decide if pixel B is a local maximum of ||g|| in the direction of the image gradient, see Figure 2. For that aim, $g_{xy}(B)$ needs to be compared to $g_{xy}(A)$and $g_{xy}(C)$; B is a local maximum when $g_{xy}(B) > g_{xy}(A)$ and $g_{xy}(B) > g_{xy}(C)$. A simple finite difference scheme can be used to approximate g_{xy} at the positions of the grid, for example for the points labeled B, D, E, F , and G. Values at points A and C cannot be computed in this way. Canny proposed to approximate the values $g_{xy}(A)$ and $g_{xy}(C)$ by a simple linear interpolation of the two nearest known values

$$g_{xy}(A) \approx \frac{g_x(B) - g_y(B)}{g_x(B)} g_{xy}(D) + \frac{G_y(B)}{G_x(B)} g_{xy}(E),$$

$$g_{xy}(C) \approx \frac{g_x(B) - g_y(B)}{g_x(B)} g_{xy}(G) + \frac{G_y(B)}{G_x(B)} g_{xy}(F).$$

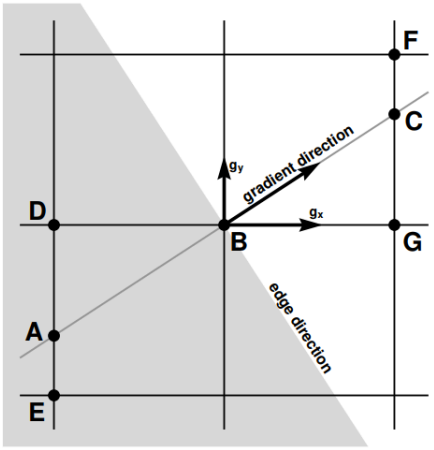

Figure 2 Support of Canny's non-maximum suppression operator

Threshold Hysteresis Processing

Canny proposed to discriminate true edges using two threshold, a high threshold and a low threshold. Pixels with g_{xy} lower than the low threshold are suppressed and not regarded as edge points; pixels with g_{xy} above the high threshold are defined as strong edges and remain as edge points; those between the high and low thresholds are defined as weak edges and are left for further processing.

5.1.2 *The Devernay Sub-Pixel Correction*

To produce sub-pixel accuracy edge points, Devernay computed a quadratic interpolation of the gradient norm between three neighboring positions along the gradient direction. With quadratic interpolation of the gradient norm along the three points used in the Canny operator: $(A, g_{xy}(A)), (B, g_{xy}(B))and(C, g_{xy}(C))$, see Figure 3. Solving the maximum leads to an offset of

$$\eta = \frac{1}{2}\frac{g_{xy}(A) - g_{xy}(B)}{g_{xy}(A) + g_{xy}(C) - 2g_{xy}(B)} \tag{2}$$

relative to the vector $\overrightarrow{BC}$.

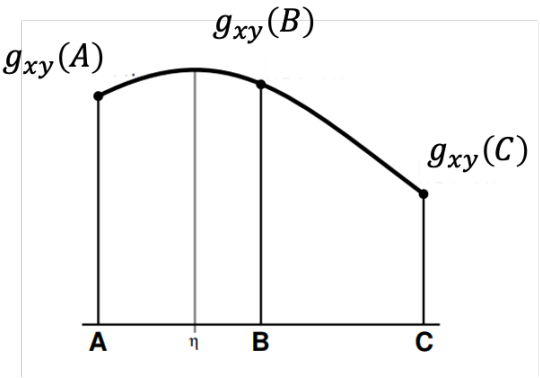

Figure 3 Profile of the norm of the image gradient along the direction of the image gradient (roughly orthogonal to the edge)

Algorithm 3 algorithm 3:Sub-pixel-fix(T,$\{a_i\}$,)

Input: Two transform map T=(T_x, T_y), points list of contour $\{a_i\}$

Output: the new points list after subpixel fix $\{\hat{a_i}\}$

```
len =length of {a_i}
for i = 1; i <= len; i++ do
    â_i.x = a_i.x + g_x
    â_i.y = a_i.y + g_y
end for
```

5.1.3 *Find Continuous Edges Roughly with OpenCV*

OpenCV can directly give the edge curve of the image and the coordinates of the points on the curve with pixel accuracy. However, there are three challenges.

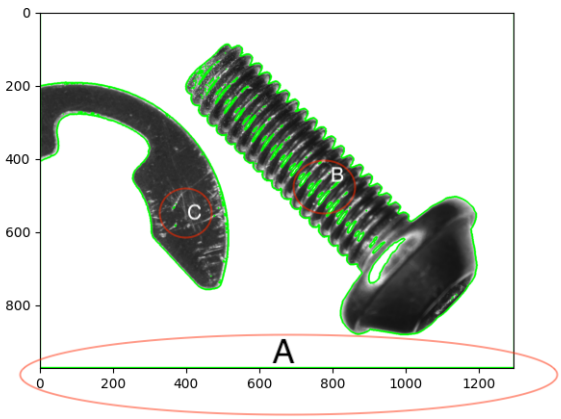

Figure 4 Detect the edges with OpenCV and there're some flows. (in part A and part B for example)

The first challenge. As shown in the B and C part of figure 4, many curves that are not contours are also recognized.

Solution 1. Therefore, to eliminate these interference, we propose to respectively calculate the length (we sum up the count of the points instead, as the count of points is relative to the length) of all the curves that are judged as contours, and to sort the lengths of these curves. Obviously, the edges with the largest length should be contour edges.

Figure 5 Support of the first challenge: OpenCV misunderstand many points as the contour edges, which have to be eliminate

The second challenge. The contour edge detected by OpenCV have a width over one pixel as the figure 6 shows.

Solution 2. To estimate the real length of the product contour, we approximate the arc with some straight lines, as figure 8 presents.Notice that every smooth curve can be fitted by several line segments, and when the $d_s \to 0$, we have $\sum |d_s|$=S, here S denote

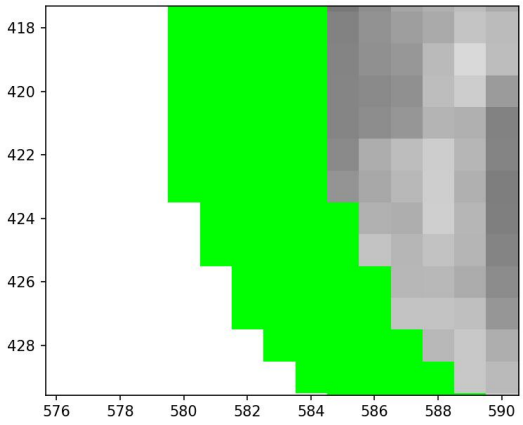

Figure 6 Zoom in the contour edges (green) detected by OpenCV

the perimeter of the curve. So we can only use several control points to represent the whole contour edge detected by OpenCV, say, a_i is the control points list, which form a poly-line(polygon) together, In this case, we can simply calculate the perimeter of the polygon to approximate the perimeter of the curve.

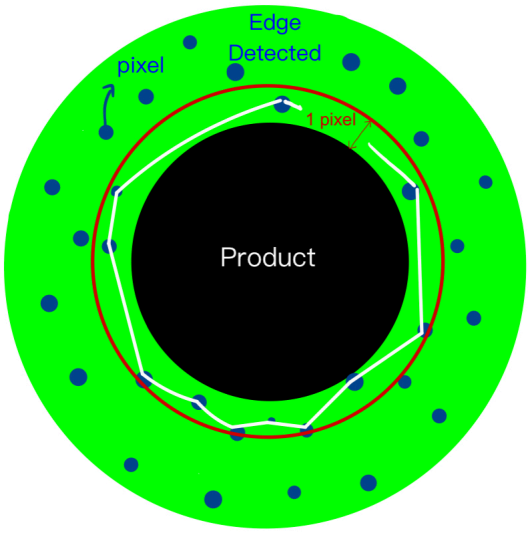

Figure 7 Zoom in the contour edges (green) detected by OpenCV : Blue dots are some of the pixel which recognized as the contour edge. Those pixel which close to the real edge of the product within 1 pixel are linked to approximate the contour edge.

The third challenge. Actually, the frame of the photo is usually recognized as well, as shown in the part A of figure 4. The length of the photo frame is very long among those recognized curves. But it is not required. Therefore, the images have to be preprocessed when the operations above are performed.

Solution 3: preprocessing. We change the color of the edge-pixel to white (we plot it red in figure 8 to present clearly) with computer programs based on whether the nearest pixel in the direction of inside is black enough. And then, we calculated the contour edges in black, and minus the calculation of Ω.

Figure 8 Support of preprocessing

5.1.4 *Get Sub-Pixel Contour Edge with Devernay Correction*

Then, with Devernay sub-pixel correction mentioned in 3.1.2 to process the contours have been recognized. Let $\|\eta\|, \eta_x, \eta_y$denote, respectively, the calculated offset of η. It should be pointed out that the Devernay method is an improvement of Canny algorithm, but OpenCV doesn't rely on Canny algorithm to recognize edge points though it also provides edge points with pixel accuracy. So some of η is over 1, which means the refined sub-pixel isn't between A and C. . To solve the problem, we don't modify those $\eta > 1$ pixel. Then clear distribution of $\|\eta\|, \eta_x, \eta_y$ can be seen in the Figure 9 heat map below. Consequently, the extracted contours are refined as sub-pixel contours and have greater accuracy. Figure 9 is the modified output .

5.1.5 *Results*

The results is showed in figure 10.

5.1.6 *Analysis of the Results*

We calculated both the pixel and sub-pixel length of the contour edges, the results are as followed. We find that our model and algorithm have the best calibration to picture. It could be resulted from the abundant noise in Pic 3.

Table 1 Analysis of the results (Unit: pixel)

Num	Pixel Length	Sub-Pixel Length	Δ	Modify Rate = $\frac{\Delta}{PixelLength}$
1	3185.3498	3286.7304	+101.3806	3.18%
2	8530.3250	8563.6235	+ 33.2985	0.39%
3	5165.2340	5430.3415	+ 265.1075	5.13%

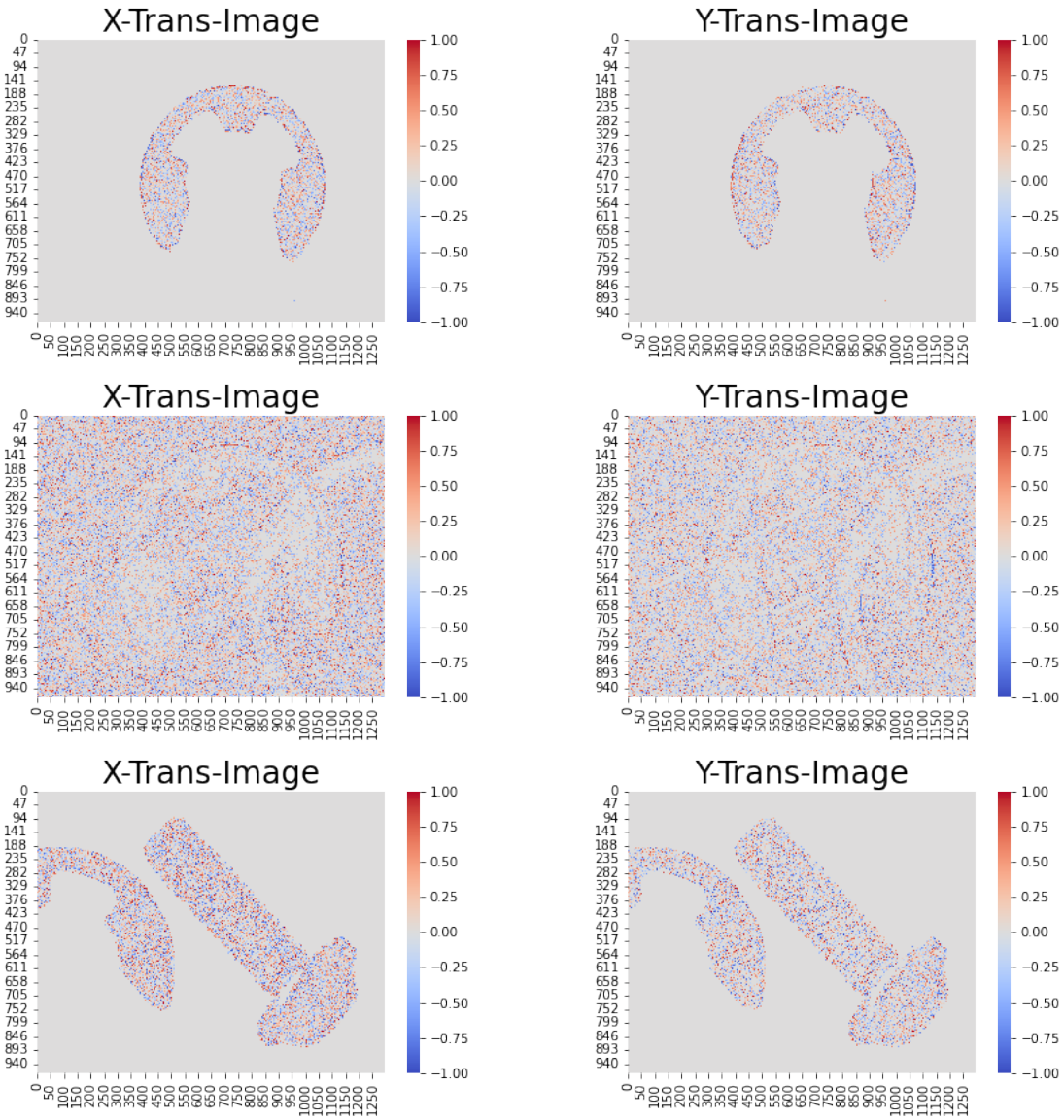

Figure 9 The heat map of the distribution of η

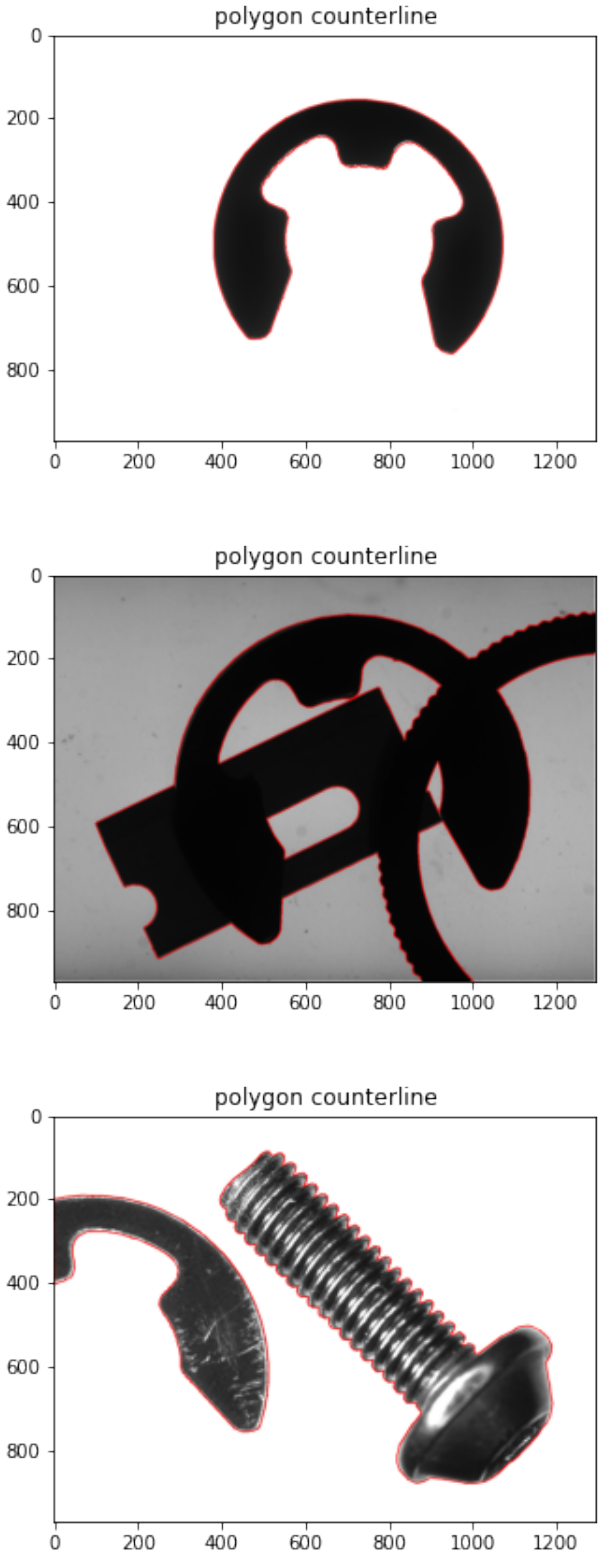

Figure 10 Results of the problem

5.2 Models and Solution of Question 2

5.2.1 *Preprocess: Calibrate the Image*

To eliminate lens distortion of camera, it is necessary to establish the geometric model of the camera and get parameters of camera.

These camera parameters can be used to calibrate the images' distortion and eliminate re-projection error. Round-mesh plate is a popular calibration target design based on a black circle on a white background. During processing, circles can be detected as spots in an image. With some simple limits, such as area, roundness, convexity, etc. applied to these binary speckle areas can remove the candidate's bad feature points. And Images C_I, which are always some photos of calibration plate take at similar place can be calibrated into C_I'. We use a correct matrix function **Correct(X)** to get C_I' , where

$$C_I \cdot \textbf{Correct}(C_I) = C_I'$$

With the help of the Calibrate board, the image X_I of products can be calibrated into X_I'.

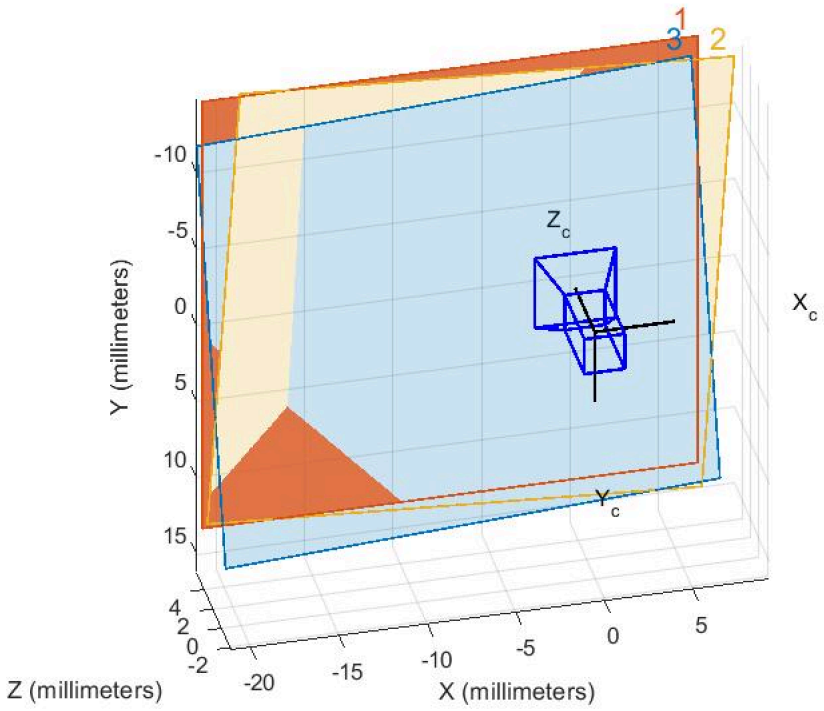

Figure 11 $Correct(C_I)$**: 3D camera centric picture**

5.2.2 *Calculate the Ratio of Real World Coordinate System to Pixel Coordinate System*

After getting the fixed coordinates in pixel coordinate system, the pixel-coordinate of the circles' center on the calibrate plate can be output by algorithm which uses OpenCV to detect the coordinates of the circles' center and performs sub-pixel correction . The output are a set, which is$(P_1, P_2, \ldots, P_n)$.

When the centers of the circles are confirmed, the average distances from circle to circle can be calculated. We calculated the average distances of those neighbor points.

$$\overline{P} = \frac{1}{n} \sum_{P_i, P_j are-neighbors} \|P_i - P_j\|$$

Then we can give the ratio between the real world coordinate system and pixel coordinate system,

$$r = \frac{d}{\overline{P}} \tag{3}$$

where d is the calibration plate's parameter of real distance between two neighbor dots , $d = 2mm$.

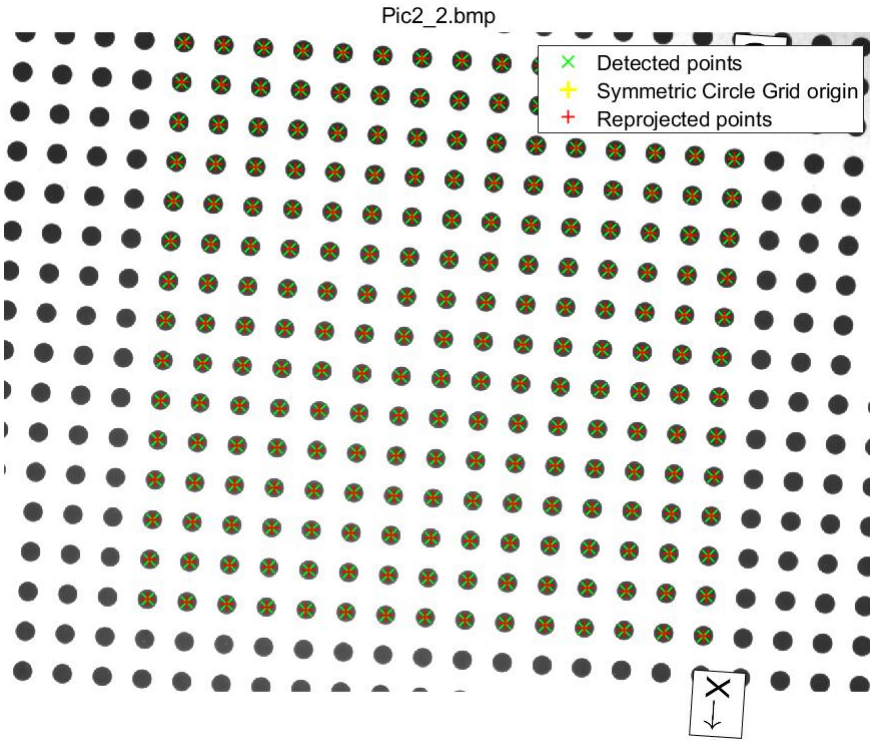

Figure 12 Confirmed centers of circles on the calibrate plate

5.2.3 *Get the Real-World Coordinates of the Product*

The real world coordinates of the product is

$$X_I' = X_I \cdot \textbf{Correct}(C_I) \tag{4}$$

5.2.4 *Calculate the Pixel-Length of the Product's Border with Polygon Algorithm*

The Polygon Algorithm has been mentioned in 4.1.3 Solution 2. And with this algorithm, we can calculate the Pixel length $D(X_I')$.

5.2.5 *Calculate the Real-Length of the Products*

$$D(X_r) = r * D(X_I')$$

5.2.6 *Results*

We can see that the calibration do work by a little experiment. We split the image into two parts.The left side of the image (x<975) is the undistorted image corrected by our algorithm, while right side of the image (x>975) intentionally leave blank.

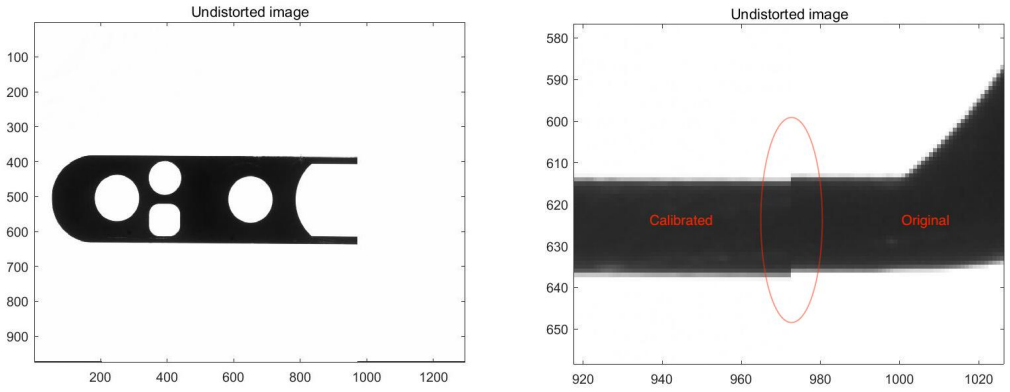

Figure 13 Contrast of image before calibration and the undistorted image

To contrast the image before calibration and the undistorted image, we copy the right side of the old image into our picture. We can see there is an obvious shifting in x= 975, where the left side is a straight line segment after the calibration. However, the right side still keep its distortion.

5.3 Models and Solution of Question 3

5.3.1 *Circumscribed Circle Radius to Estimate the Curvature*

Heron's formula states that the area of a triangle whose sides have lengths a, b, and c is

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where s is the semi-perimeter of the triangle; that is,

$$s = \frac{a+b+c}{2}$$

based on this equation we get the radius of circumscribed circle is as below

$$R = \frac{abc}{4\sqrt{s(s-a)(s-b)(s-c)}} \tag{5}$$

From basic calculus, we know that any continuous curves can be well fitted locally,

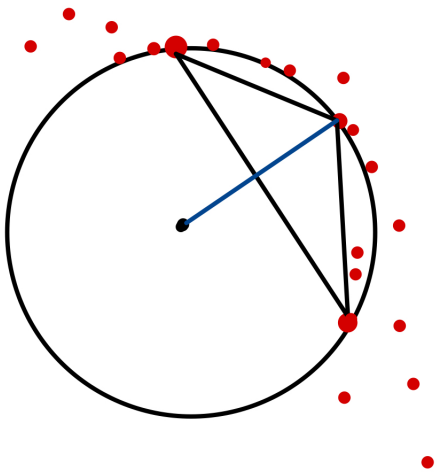

Figure 14 Estimation of curvature

through a small arc of a **circle** with same curvature. So in the same way , we sample three points from the curve to represent the curve and locate a circumscribed circle through three points(a triangle). In this circumstance, for that the edge is a continuous line, the circle fits the arc well . As a result the curvature (the reciprocal of the radius) of the circumscribed circle can be a good estimation of the curve's curvature.

5.3.2 *Gauss Filter to Eliminate The Noise of the Image*

Then we **use Gauss filter to eliminate the noise of the image.** For that, we use Gauss matrix

$$G_1 = [0.05, 0.1, 0.2, 0.3, 0.2, 0.1, 0.05]$$

$$G_2 = [0.03, 0.05, 0.07, 0.2, 0.3, 0.2, 0.07, 0.05, 0.03]$$

to respectively have dot product with the dots near P_i, that is

$$R_i' = G_1 \bullet [R_{i-3}, R_{i-2}, R_{i-1}, R_i, R_{i+1}, R_{i+2}, R_{i+3}, R_{i+4}]^T$$

$$C_i' = G_2 \bullet [C_{i-3}, C_{i-2}, C_{i-1}, C_i, C_{i+1}, C_{i+2}, C_{i+3}, C_{i+4}]^T$$

5.3.3 *Splitting The segment*

As a series of points can have numerous features, many features can indicate its property. Firstly, we considered the feature of radius. However, for any straight line , radius is infinite , which can thus cause a lot of problem. After a careful selection, we choose to split are based on the **curvature and the gradient of the curvature** . That is because, we could simply classify the arcs into circular arcs , elliptical arc and straight line segment as below : If P_i's curvature is close to zero, it will be classified into a straight line.Otherwise if its gradient of curvature is close to zero ,it will be classified into a circle or circular arc. And if it's not these types above, it is an ellipse or elliptic arc.

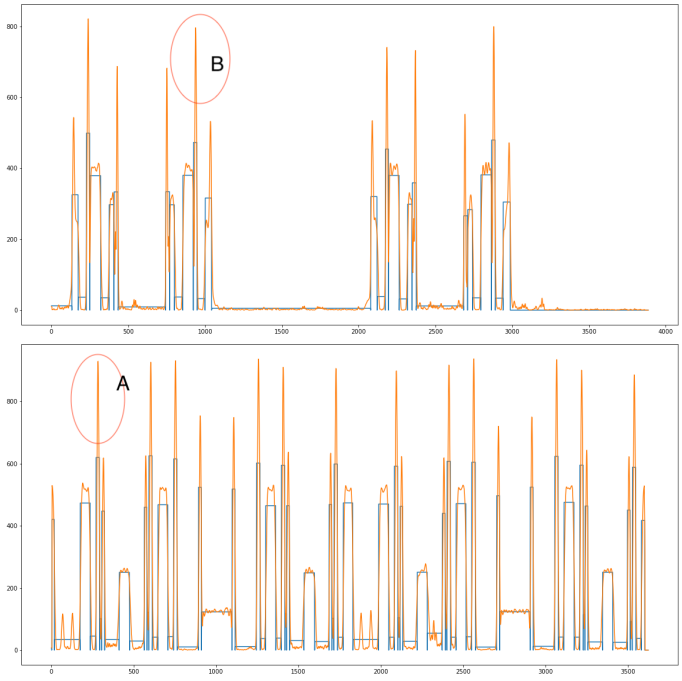

Figure 15 Average curvature of each edge: the implementation of the split of the two sub-pixel contour edge. For example, curvature (orange) of points in A and B is beyond the average curvature (blue) of their group.

Figure 16 (next page) shows the distribution of curvature, curvature after gauss filter, gradient of the curvature of the first sub-pixel contour edge. To observe them

more clearly, part of the distribution is shown in the c of figure16. The points between 2400 and 2450 can be classified into a Line, as their curvatures ≈ 0. The points between 2325 and 2350 can be classified into a Circle or CircularArc, as their curvatures are a constant and the gradient of their curvatures ≈ 0.The points between 2350 and 2375 can be classified into a Ellipse or EllipticArc,because both of their index are changing from points to points.The intact threshold to split is in codes in the appendix.

Then, the detailed implementation of the split is demonstrated in figure 15. Considering that there might be some points whose gradient of curvature and curvature can be significantly abnormal in its group. For example, a few points in a group classified as an Arc may have different curvature with others points in the group. So compare them with the **average curvature** of the group, the abnormal curvature points can be found and smoothed through the averaging the curvature of the point groups.As is shown with the figure below , the stages of the segments can be a very robust feature to represent the curvature of the segment. Figure 18 are the breakpoints.

5.3.4 *Least Squares Fitting of Ellipse*

In analytic geometry, the ellipse is defined as a collection of points (x, y) satisfying the following implicit equation

$$Ax^2 + Bxy + Cy^2 + Dx + Ey = 1$$

Several new notations needs to be introduced to ease our discussion. For two vectors s = $(s_1, s_2, ..., s_m)^T$ and t = $(t_1, t_2, ..., t_m)^T$, the tensor product between them are defined as

$$s \otimes t = (s_1 t_1, s_2 t_2, \ldots, s_m t_m)^T$$

Assuming n measurements $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$are given, we define$x = (x_1, x_2, ..., x_n)^T$and $y = (y_1, y_2, ..., y_n)^T$, then the following cost function needs to be minimized

$$C(\beta) = (\mathbf{X}\beta\text{-}\mathbf{1})^T(\mathbf{X}\beta\text{-}\mathbf{1})$$

where $X = [x \otimes x, x \otimes y, y \otimes y, x, y]$ is a n-by-5 matrix, $\beta = (A, B, C, D, E)^T$ consists of the parameters to be determined, and **1** is a n-dimensional column vector with all 1s. To minimize $C(\beta)$ in matlab,and with a series of operation, we can get the coordinate of the ellipse center in the original coordinate system.

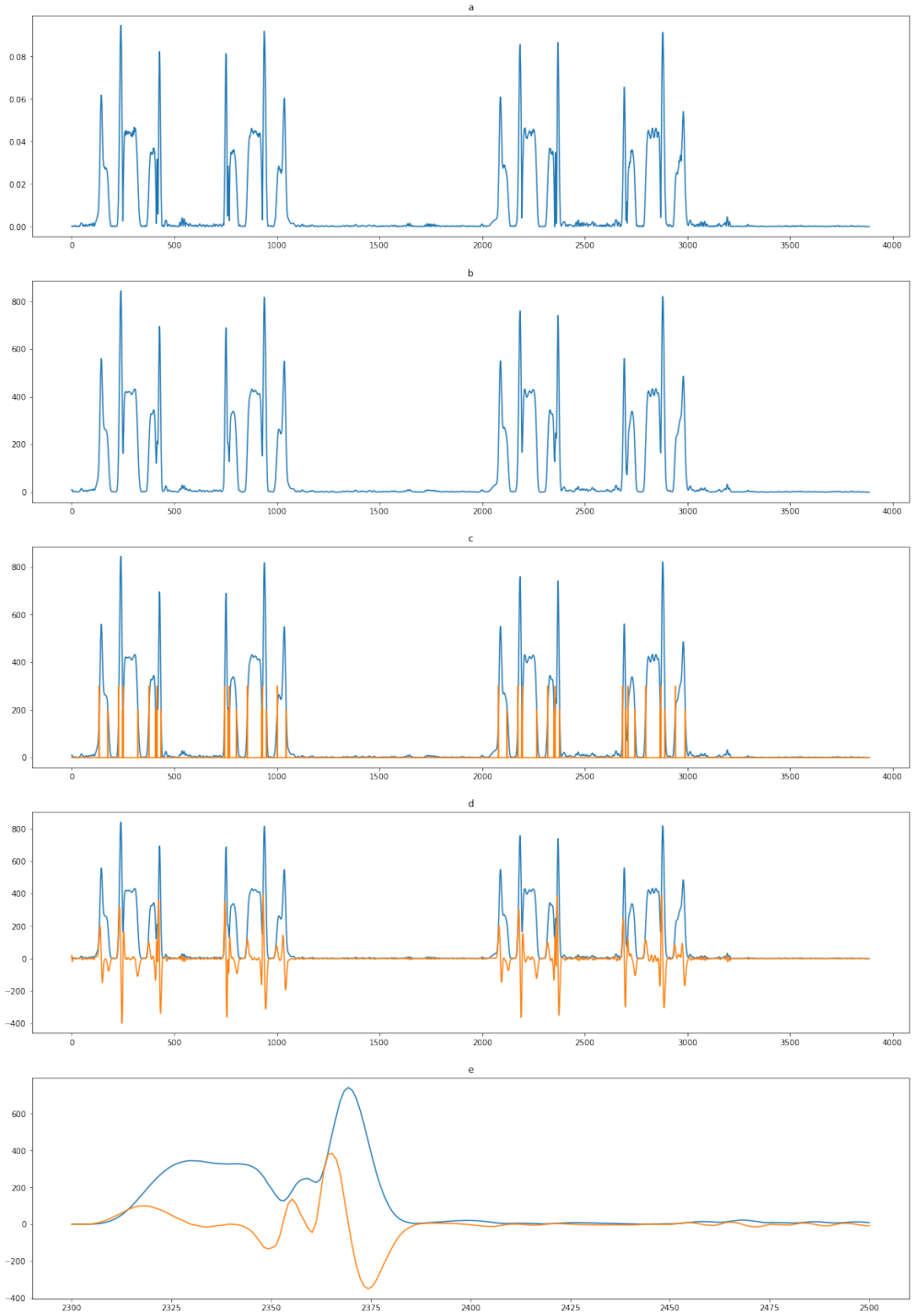

Figure 16 Information of first sub-pixel contour edge

a: Distribution of curvature. **b:** Distribution of curvature after Gauss filter, apparently the curves are more smooth. **c: Breakpoints** (orange). **d:**Distribution of curvature (blue) and its gradient (orange). **e:** Part of the distribution of curvature (blue) and its gradient (orange). Points between 2400 and 2450 can be classified into a Line. Points between 2325 and 2350 can be classified into a Circle or CircularArc. Points between 2350 and 2375 can be classified into a Ellipse or EllipticArc.

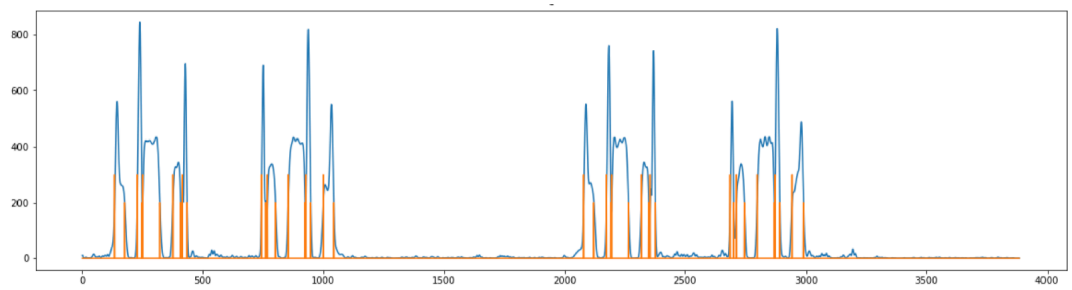

Figure 17 The second sub-pixel contour edge's distribution of curvature after Gauss filter and breakpoints (orange)

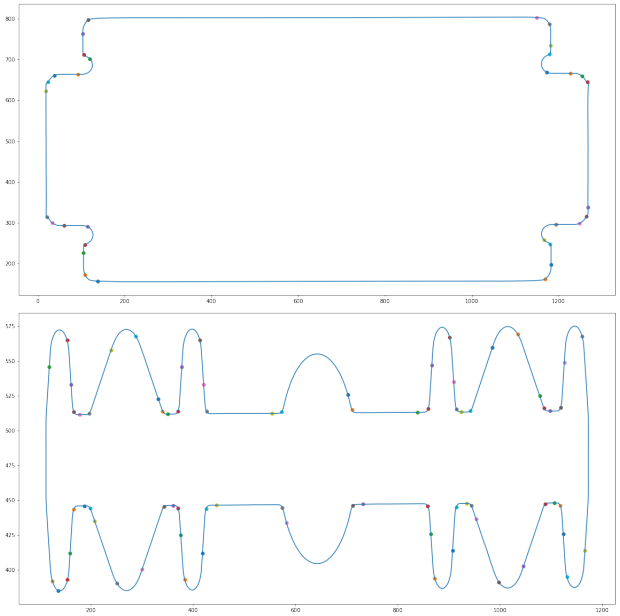

Figure 18 Break points of the two sub-pixel contour egdes

5.3.5 *Results*

We plot the image by coordinates in each group we classified. Specifically, in Figure 20 1st and the 3th figure shows the fitting of the circular and elliptical arc; the rest shows the fitting line segment. (thin blue background is the original edge) As is shown in the figure, the fitting lines suggest highly accurate approximations of the contour.

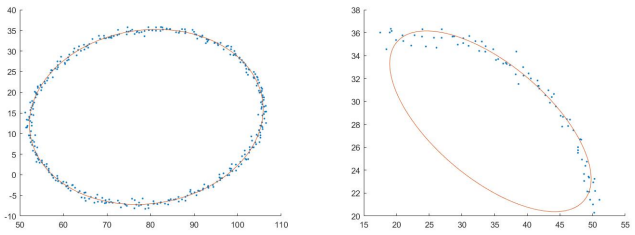

Figure 19 Results of least squares fitting of some of ellipse

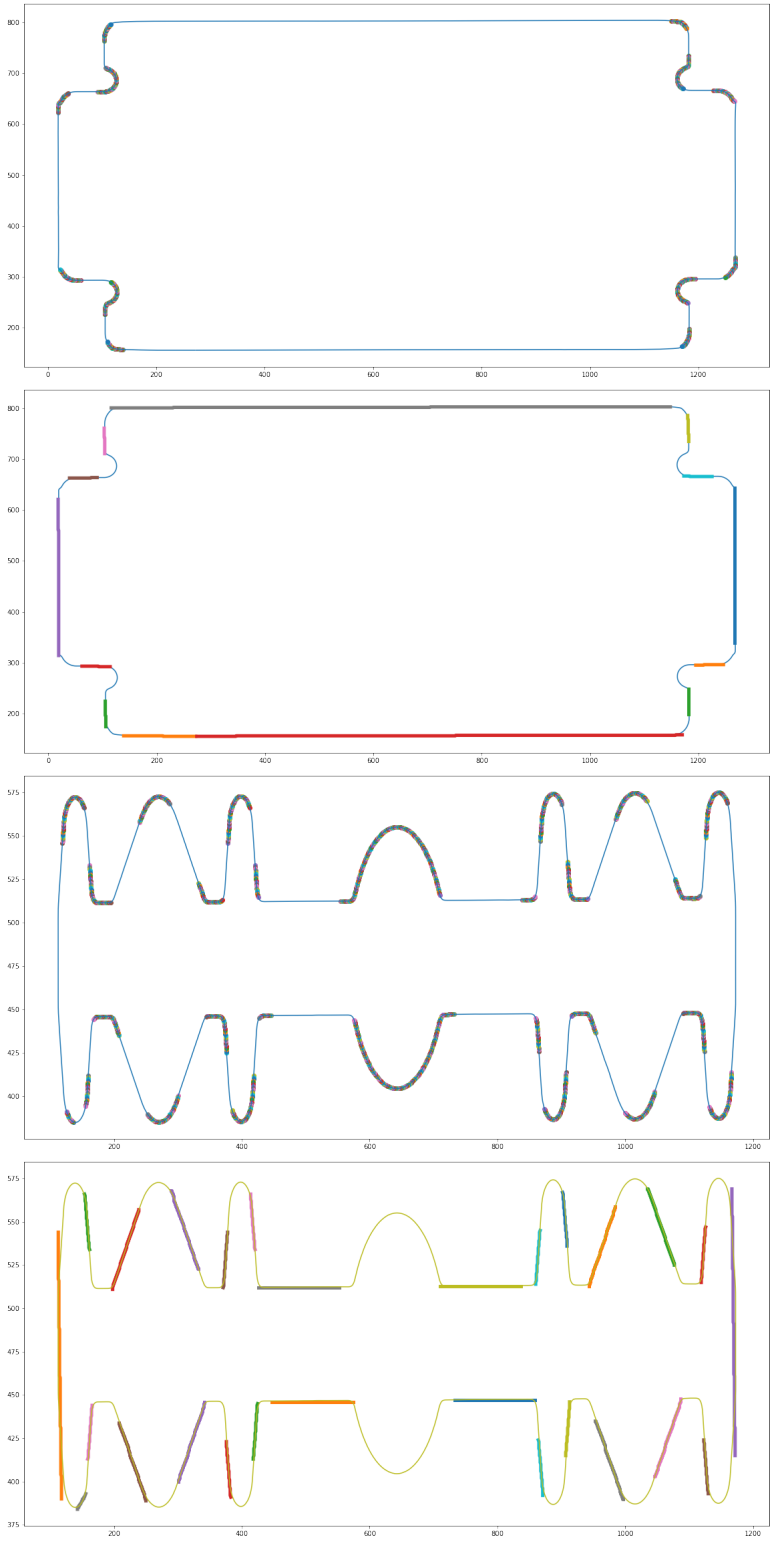

Figure 20 Results plot: fitting curves and fitting line segment of the Edgecontour1 and Edgecontour2

VI. Innovative Works

1. In the problem 1, we combine both advantages of contours and the devernay algorithm. Since the former method treat the whole boundary as a whole, which helps us how to eliminate the interference effects of edge burrs and shadow parts of the edges, while the latter algorithm improve the accuracy of our calculation to sub-pixel.
2. In the problem 2, as Newton has said, *If I have seen further, it is by standing on the shoulders of giants.*We make full use of the the programming language, C++, python, matlab to derive a satisfying result.
3. In problem 3, our innovative work is that we successfully avoid the calculation of curvature, which may divergent since the noise exists. We use three adjacent points to calculate circumcircle determined by the triangle. WhatâĂŹs more, we smartly choosing the appropriate threshold to aggregate some neighbor points to only one to derive the right answer.

VII. References

[1] GROMPONE VON GIOI, Rafael; RANDALL, Gregory. A sub-pixel edge detector: an implementation of the canny/devernay algorithm. IPOL Journal-Image Processing On Line, 2017, vol. 7, pp. 347-372, 2017.

[2] LI, Zhe, et al. Binocular vision system calibration based on HALCON [J]. Development Innovation of Machinery Electrical Products, 2013, 26.3: 95-96.

[3] CANNY, John Francis. Finding Edges and Lines in Images. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1983. Laboratory, 1983.

[4] MAINI, Raman; AGGARWAL, Himanshu. Study and comparison of various image edge detection techniques. International journal of image processing (IJIP), 2009, 3.1: 1-11.

[5] VINCENT, O. Rebecca, et al. A descriptive algorithm for sobel image edge detection. In: Proceedings of informing science IT education conference (InSITE). 2009. p. 97-107.

[6] HARALICK, Robert M. Zero crossing of second directional derivative edge operator. In: Robot Vision. International Society for Optics and Photonics, 1982. p. 91-101.

[7] PRASAD, Dilip K.; LEUNG, Maylor KH. Methods for ellipse detection from edge maps of real images. Machine Vision-Applications and Systems, 2012, 135-162.

[8] SPONTÃŞN, Haldo; CARDELINO, Juan. A review of classic edge detectors. Image Processing On Line, 2015, 5: 90-123.

VIII. Appendix

Listing 1: The python source code: The algorithm to calculate the length

```
p1=mi.image(image_new_path_1,[])
#read the edge
p1.get_origin_contour(image_bound_path_1,-2)

#Calculate the sobel
p1.set_sobel()
#Devernary_smooth (sub-pixel correction)
p1.Devernary_smooth(heat_path_1)
p1.sub_pixel_correct()
p1.write_in_excel(path_1_1,'page_1')
```

Listing 2: The matlab source code :Algorithm of calibration in camera coordinate system

```
load matlab_calib.mat;%Load the parameters
intrinx = cameraParams.IntrinsicMatrix;
I2=imread("new_image.png");

fx = intrinx(1,1);
fy = intrinx(2,2);
fc = [fx fy];
cx = intrinx(1,3);
cy = intrinx(2,3);
cc = [cx cy];
alpha_c = 0;
k21 = cameraParams.RadialDistortion(1);
k22 = cameraParams.RadialDistortion(2);
k23 = 0;
p21=cameraParams.TangentialDistortion(1);
p22=cameraParams.TangentialDistortion(2);

[H,W] = size(I2);%Image Size
W=1171;
I22 = 255*zeros(H,W);
```

```
for x=1:H
    for y=1:W
    %Pixel-coordinate system to Camera coordinate system
        yy=(y-cy)/fy;
        xx=(x-cx)/fx;
    %Calibrate in Camera coordinatesystem
        r=xx^2+yy^2;
        xxx=xx*(1+k21*r+k22*r^2+k23*r^3)+2*p21*xx*yy+p22*(r+2*xx^2);
        yyy=yy*(1+k21*r+k22*r^2+k23*r^3)+2*p22*xx*yy+p21*(r+2*yy^2);
    %Camera coordinate system to Pixel-coordinate system after
        calibrated
        xxxx=xxx*fx+cx;
        yyyy=yyy*fy+cy;
        if (xxxx>1 && xxxx<=W && yyyy>1 && yyyy<=H)
            h=yyyy;
            w=xxxx;
           %Interpoloation
           I22(y,x)=(floor(w+1)-w)*(floor(h+1)-h)*I2(floor(h),floor(w)
           )+...(floor(w+1)-w)*(h-floor(h))*I2(floor(h+1),floor(w))
           +(w-floor(w))*...(floor(h+1)-h)*I2(floor(h),floor(w+1))+(w-
           floor(w))*(h-floor(h))
                *I2(floor(h+1),floor(w+1));
        end
    end
end
imwrite(uint8(I22),'undistort.png')
figure();
image(I22);
colormap(gray(256));
```