

LAB: Grayscale Image Segmentation

Date: 2025-March-28

Author: JongHyeon Kim 21900179

Github: [repository link](#)

Introduction

1. Objective

Goal: Image analysis of the gear determines whether the gear is defective or not.

There is a gear with a defective gear tooth part. We can detect whether it is defective by calculating the area it occupies.

2. Preparation

Software Installation

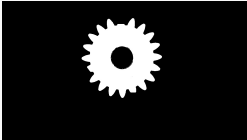

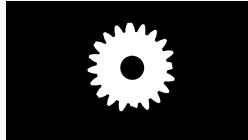
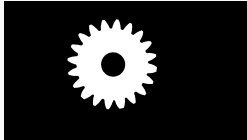
- OpenCV 4.9.0, Visual Studio 2022

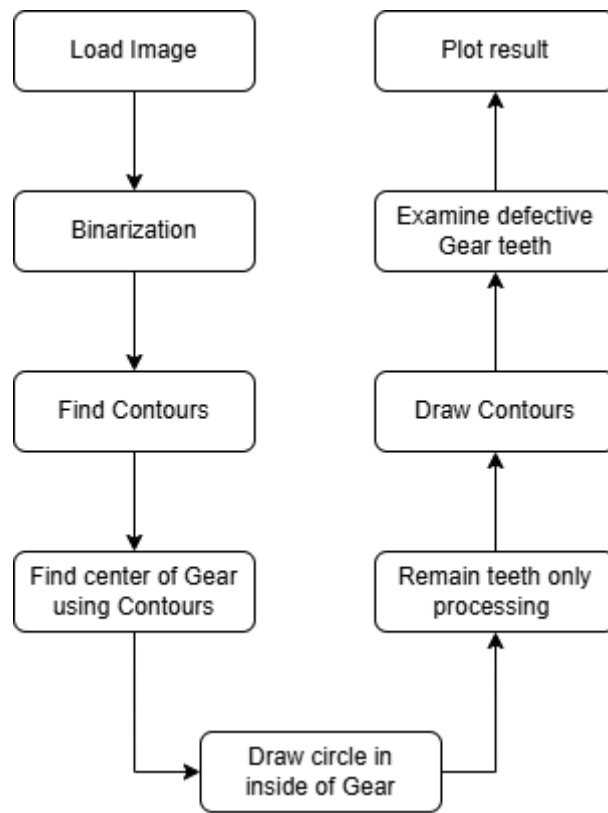
Dataset

Dataset link: [Download the test image](#)

Algorithm

1. Overview

			
Gear 1	Gear 2	Gear 3	Gear 4



We will load Gear Images first. after processing, we can find out which Gear teeth are defective by the Plot results.

2. Procedure

Binarization

To find Contours, we have to convert an image to Binary data. So, we have to use the Threshold function in openCV.

Before Binarization										After Binarization									
0	0	2	0	1	0	255	255	254		0	0	0	0	0	0	255	255	255	
0	0	0	0	0	2	254	253	255		0	0	0	0	0	0	255	255	255	
0	0	2	0	1		252	255	255	253	0	0	0	0	0	255	255	255	255	
0	0	0	0	1		255	254	255	255	0	0	0	0	0	255	255	255	255	
0	0	0	5		251	255	253	252	255	0	0	0	0	255	255	255	255	255	
0	0	0	0		255	255	255	255	255	0	0	0	0	255	255	255	255	255	
0	0	1		255	255	255	251	255	255	0	0	0		255	255	255	255	255	
0	0	0		253	255	254	255	255	254	0	0	0		255	255	255	255	255	
0	1		255	255	255	255	255	255	255	0	0		255	255	255	255	255	255	

As we can see image above, after binarization, we can only see the intensities 0 and 255. I gave threshold value as 128.

```
threshold(Gears[i], GearBinaries[i], g_thresh, 255, THRESH_BINARY); // Convert
Raw Image to Binary Image.
```

Find Contours

To make contours teeth only, we have to find contours twice, and this is the first processing to find outer contours.

```
findContours(GearBinaries[i], contours1[i], RETR_EXTERNAL, CHAIN_APPROX_SIMPLE,
Point(0, 0));
drawContours(drawings[i], contours1[i], -1, Scalar(0, 255, 0), 2, 4, noArray());
```

Find the center of Gear using Contours

To find center of Gear, we have to find a circle that surrounding Gear and find the center. So, we can use minEnclosingCircle function in openCV to in order to find the center point. Find outer circle is not important for this lab. We need Gear inside of circle.

```
minEnclosingCircle(contours1[i][0], centers[i], mECradii[i]);
```

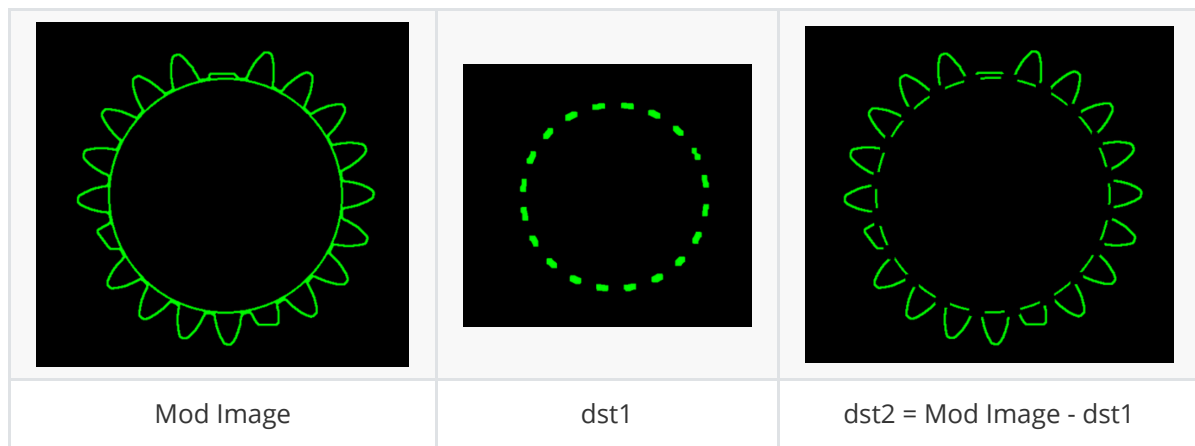
Draw a circle inside of the Gear

After earn the center point, we have to make an inner circle line to make contours that surround teeth only. By using RadiusMeasure function that personally make, we can find the inner circle line.

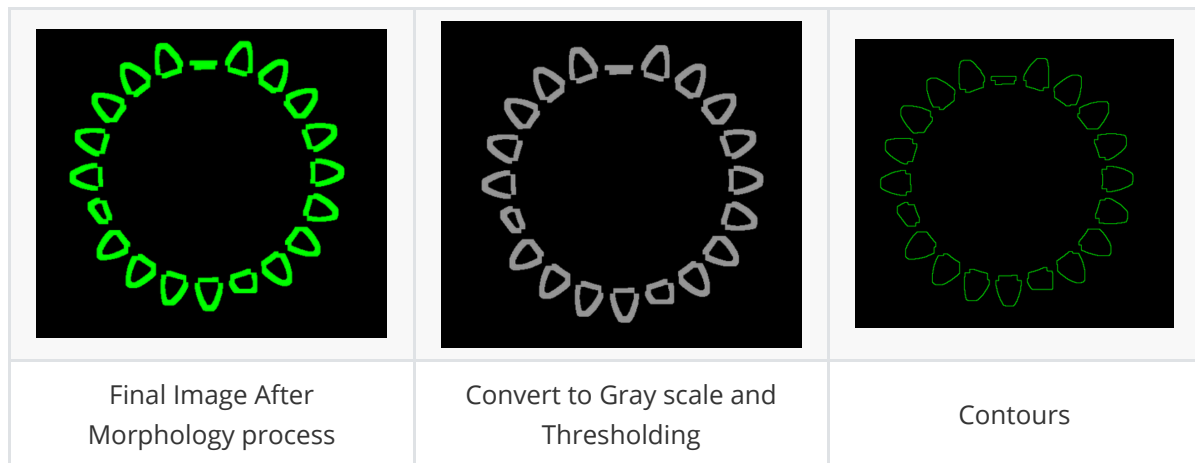
```
void RadiusMeasure(Mat &target, Point2f &center, float &Radius) {
    float value = 0.0;        // Space for checking distance.
    Radius = 1000.0;          // Initial Value.

    for (int i = 0; i < target.rows; i++) {
        for (int j = 0; j < target.cols; j++) {
            Vec3b pixel = target.at<Vec3b>(i, j);
            if (pixel[0] == 0 && pixel[1] == 255 && pixel[2] == 0) {
                value = sqrt(pow((center.x - j), 2) + pow((center.y - i), 2));
                if (Radius > value)
                    Radius = value;
            }
        }
    }
}
```

Remain teeth only process and Draw Contours



After making the inner circle, we use Mod Image source to make dst1 using morphology to erase line between each teeth. So that we can gain an image like dst2. After this, we have to make contours to gain area, but the dst2 images' line is disconnected.



By the morphology process in the dst2 image, we can gain an image like the first figure above, and to make this image as contours, convert image as gray scale, and binarization. Then, make contours like the third figure above.

```
// Morphology
morphologyEx(drawings[i], drawing1Mod, MORPH_CLOSE, KernelELL3x3, Point(-1, -1), 1);
morphologyEx(drawing1Mod, drawing1Mod, MORPH_CLOSE, KernelELL3x3, Point(-1, -1), 1);
erode(drawing1Mod, dst1, KernelELL5x5, Point(-1, -1), 1);
morphologyEx(dst1, dst1, MORPH_CLOSE, KernelELL5x5, Point(-1, -1), 4);
dilate(dst1, dst1, Kernelrect5x5, Point(-1, -1), 2);
morphologyEx(dst1, dst1, MORPH_CLOSE, KernelELL5x5, Point(-1, -1), 2);

dst2 = drawing1Mod - dst1;

// Morphology
erode(dst2, dst2, KernelELL3x3, Point(-1, -1), 1);
dilate(dst2, dst2, Kernelrect5x5, Point(-1, -1), 2);
```

```
morphologyEx(dst2, dst2, MORPH_CLOSE, Kernelrect3x3, Point(-1, -1), 1);
erode(dst2, dst2, Kernelrect3x3, Point(-1, -1), 1);
```

Examine defective Gear teeth

To check defective Gear teeth, we have to know the area of each contours. After, I decided to use the median method to find the Examine value. I am going to use the sort() function then, pick the value which locate in half of array size index.

```
for (int j = 0; j < contours2[i].size(); j++) { // Number
of Conturs : 20
    arr[j] = contourArea(contours2[i][j]);
}
```

To use the sort() function, we have to store the area value in vector space.

```
sort(arr.begin(), arr.end()); // Soring by upward.
medianVal = arr[(int)(contours2[i].size() / 2)]; // Pick the median value.
detectValMin = medianVal - tuningval; // Add tuning value to
check defective tooth (MaxValue).
detectValMax = medianVal + tuningval; // Add tuning value to
check defective tooth (MinValue).
```

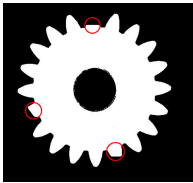
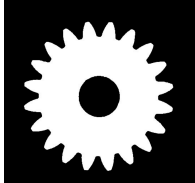
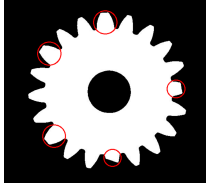
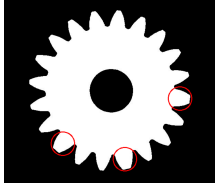
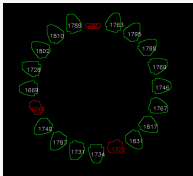
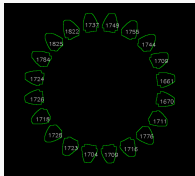
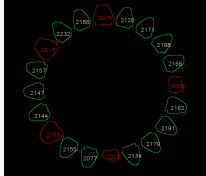
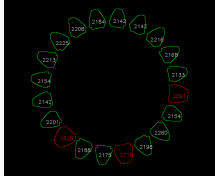
After gaining the median Value, we have to tune the value because it is a median value. I chose a tuning Value of 200. So, if the area value is upper than detectValuMax or lower than detectValMin then, mark as defective tooth.

Plot Result

Text will be inserted using the putText() function, and putText will be used to obtain each context coordinate. The coordinates were manipulated and changed to output in the middle. In addition, in order to display the gear teeth in which the problem was found, the outline of the defect was inserted into the raw gear image to be displayed.

Result and Discussion

1. Final Result

	Gear #1	Gear #2	Gear #3	Gear #4
Output Images 1				
Output Images 2				
Teeth Numbers	20	20	20	20
Avg. Teeth Area	1647.6	1734.72	2176.18	2267
Defective Teeth	3	0	5	3
Diameter of Gear	328.132	327.207	364.957	366.203
Quality	Fail	Pass	Fail	Fail
Output window	<pre>===== Gear1 ===== Teeth number: 20 Avg.Teeth Area: 1647.6 Diameter of the gear: 328.132 Defective Teeth: 3 Quality: Fail</pre>	<pre>===== Gear2 ===== Teeth number: 20 Avg.Teeth Area: 1734.72 Diameter of the gear: 327.207 Defective Teeth: 0 Quality: Pass</pre>	<pre>===== Gear3 ===== Teeth number: 20 Avg.Teeth Area: 2176.18 Diameter of the gear: 364.957 Defective Teeth: 5 Quality: Fail</pre>	<pre>===== Gear4 ===== Teeth number: 20 Avg.Teeth Area: 2267 Diameter of the gear: 366.203 Defective Teeth: 3 Quality: Fail</pre>

2. Discussion

After image processing, it can be seen that the gear teeth are slightly dented. This may cause a problem in obtaining the exact area of the object, but it is judged that there is no big problem because it is now the discrimination of defective products. In addition, it took a lot of time because the best algorithm had to be selected through a lot of morphology work to obtain the final outline. However, it seems highly likely to be applied in automation because no matter what image comes in through the overall algorithm, it automatically obtains the inscribed circle, obtains the area, and determines the defect.

Conclusion

Through image processing, the program was able to detect and display defective parts. If deep learning image processing is applied, it is expected that more accurate results can be produced.

Appendix

- opencv Documentation (<https://docs.opencv.org/4.9.0/>)
- Gary R. Bradski(2017). Learning Opencv 3: Computer Vision in C++ with the Opencv Library
- <https://velog.io/@hyesoup/%EB%91%90-%EC%A0%90-%EC%82%AC%EC%9D%B4%EC%9D%98-%EA%B1%B0%EB%A6%AC-%EA%B3%B5%EC%8B%9D>
- <https://hwan-shell.tistory.com/119>
- <https://developer-cat.tistory.com/19>
- <https://blockdmask.tistory.com/178>
- <https://ansohxxn.github.io/cpp/chapter7-2/>

```
double meanCal(vector<double>& arr,int size) {
    double result = 0;
    for (int i = 0; i < size; i++) {
        result += arr[i];
    }
    return result / size;
}
```

```
void RadiusMeasure(Mat &target, Point2f &center, float &Radius) {
    float value = 0.0;        // Space for checking distance.
    Radius = 1000.0;          // Initial value.

    for (int i = 0; i < target.rows; i++) {
        for (int j = 0; j < target.cols; j++) {
            Vec3b pixel = target.at<Vec3b>(i, j);
            if (pixel[0] == 0 && pixel[1] == 255 && pixel[2] == 0) {
                value = sqrt(pow((center.x - j), 2) + pow((center.y - i), 2));
                if (Radius > value)
                    Radius = value;
            }
        }
    }
}
```

Discription in CPP file as comment.