

LAB: Grayscale Image Segmentation -Gear

Introduction

1. Objective

Date: 2025-Mar-26

Author: Yechan Kim 22100153

Goal: Find the number of defective teeth from a gear image.

There are 4 gear images and some of them have defective teeth. You're required to find the number of the defective teeth, the diameter, and the quality inspection from each of gear image.

2. Preparation

Software Installation

- OpenCV 4.9.0, Visual Studio 2022

Dataset

Dataset link: [https://github.com/ykkimhgu/DLIPsrc/blob/main/images/Lab GrayScale Gears.zip](https://github.com/ykkimhgu/DLIPsrc/blob/main/images/Lab%20GrayScale%20Gears.zip)

Algorithm

1. Overview

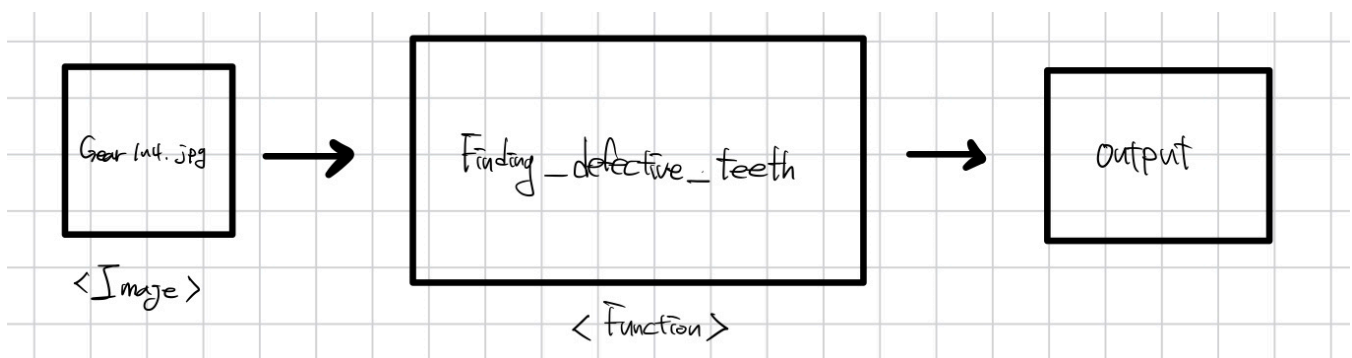


Figure1. Whole algorithm of the project

In general, by inputting a specific gear image into the `findin_defective_function`, the following results are produced:

- Contours of the teeth

- The number of defective teeth
- The location of defective teeth
- Quality inspection of the gear
- The diameter of the gear

2. Procedure

Given Gear images

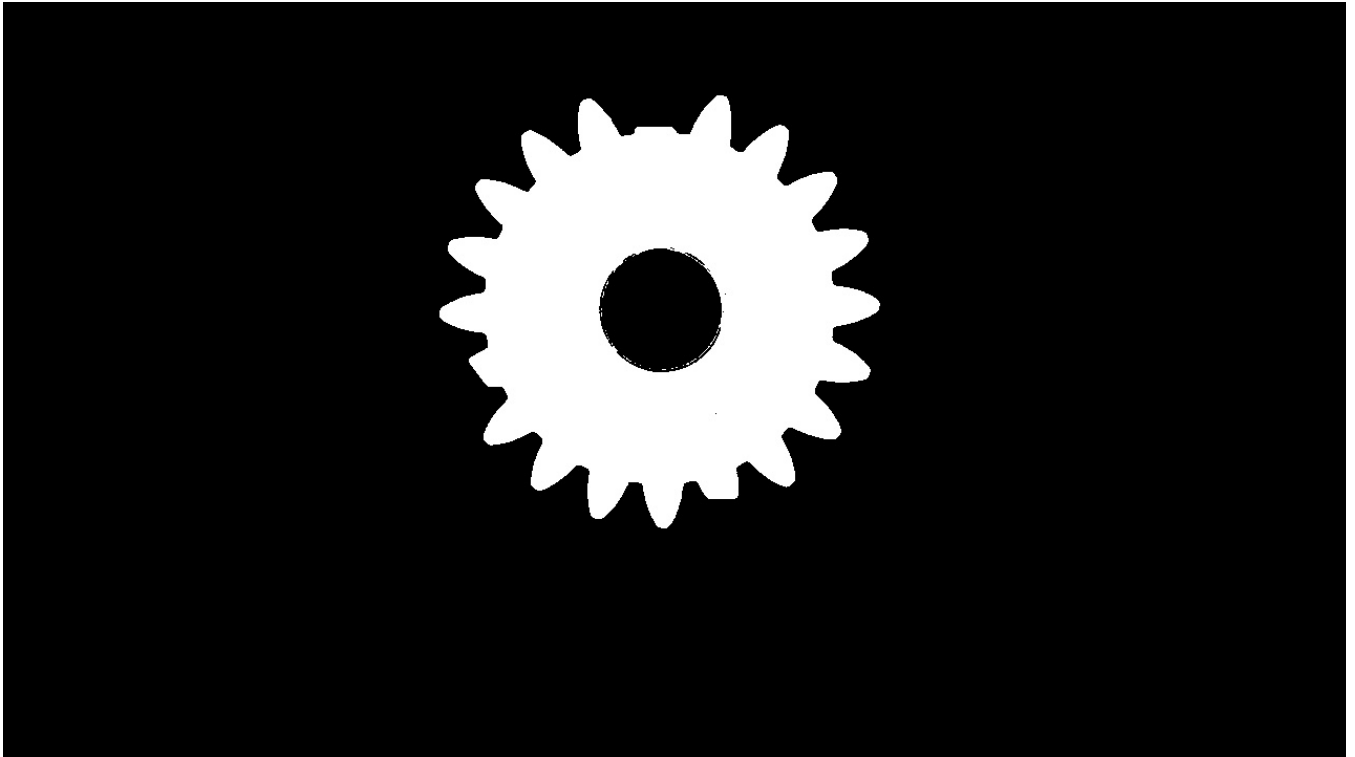


Figure2. Gear1.jpg

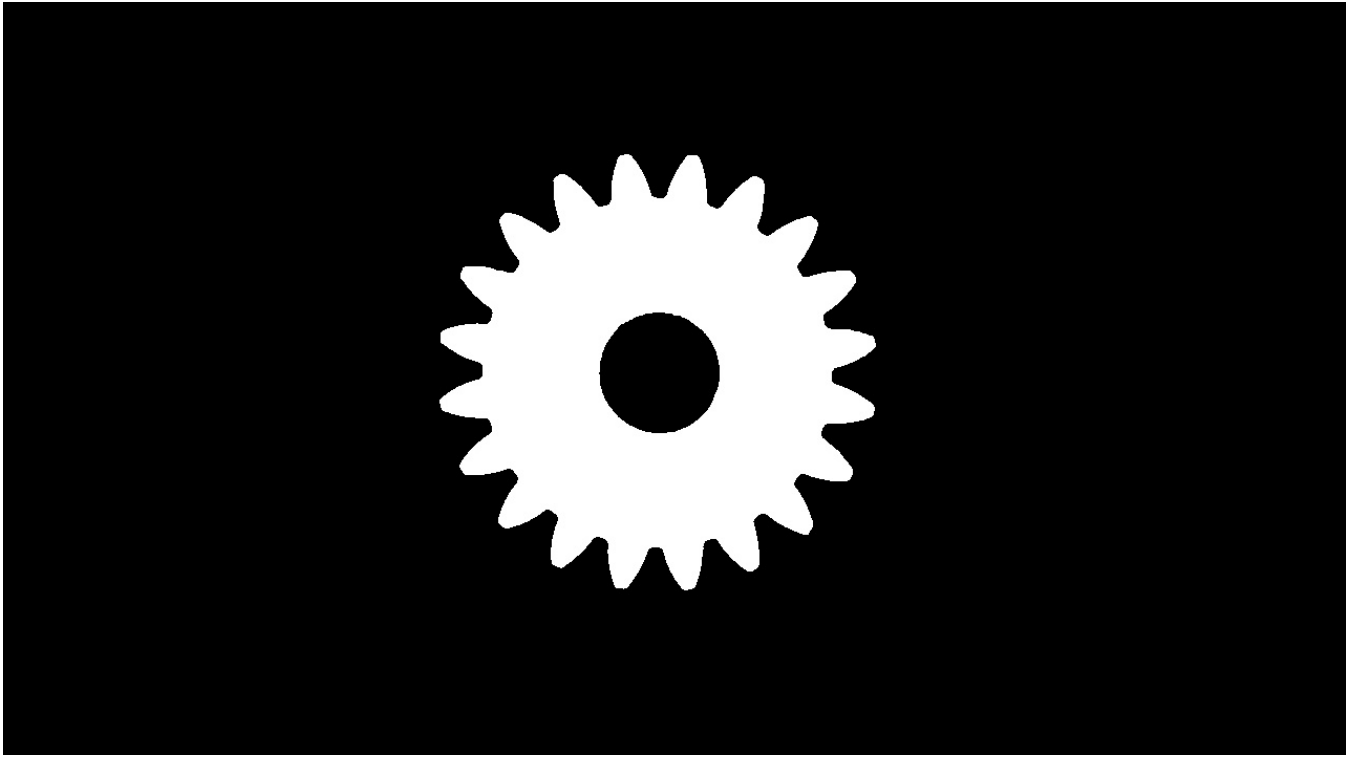


Figure2. Gear2.jpg

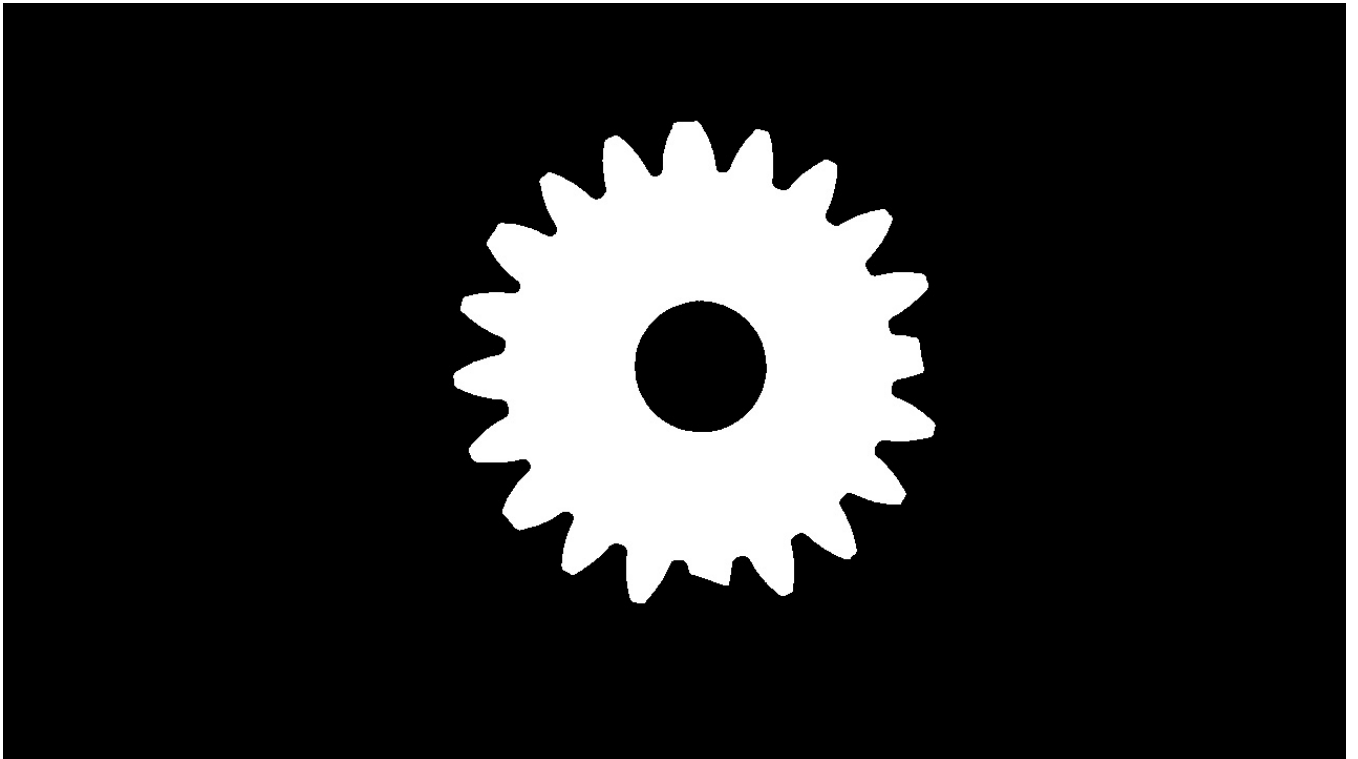


Figure3. Gear3.jpg

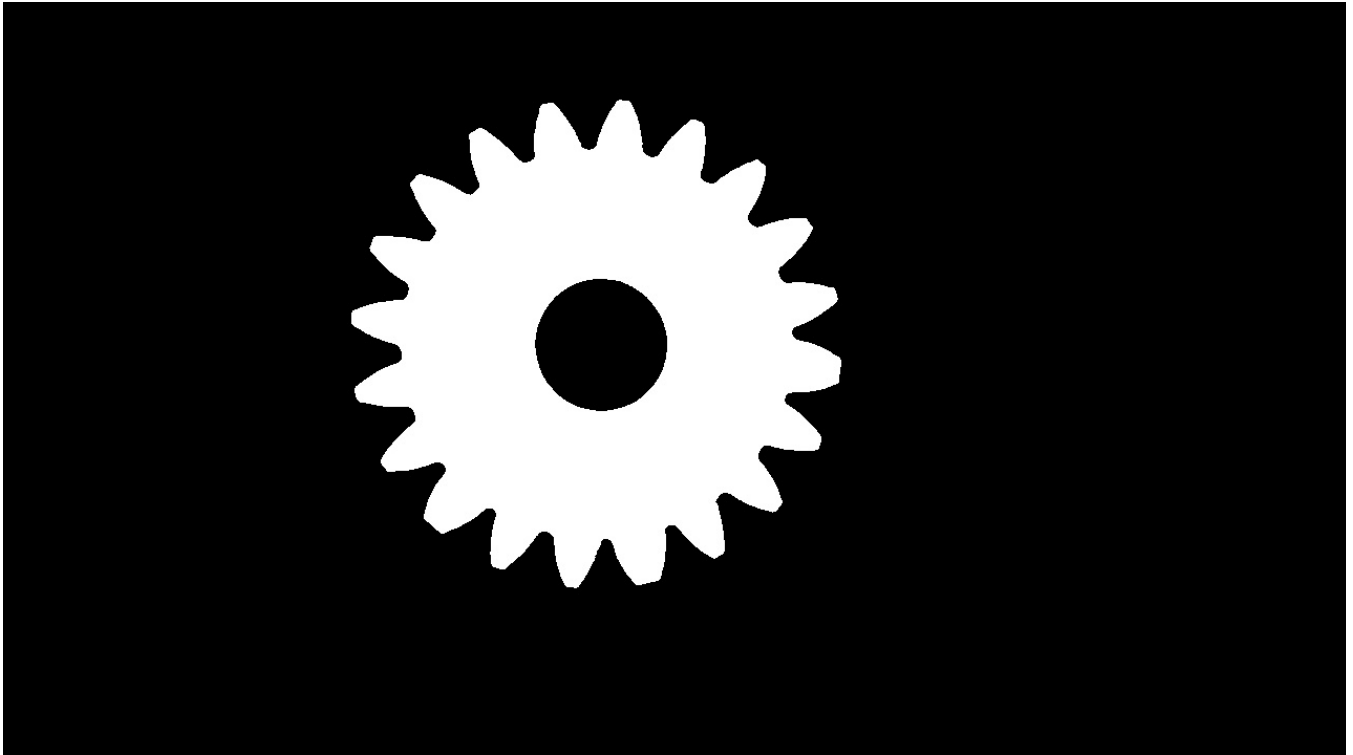


Figure4. Gear4.jpg

Expected quality inspection

Gear number	gea1	gear2	gear3	gear4
Number of defective teeth	3	0	5	3
Quality inspection	Fail	Pass	Fail	Fail

Main algorithm to find defective teeth

1. To extract contours, the original image must be converted into a binary image by applying a threshold. The optimal threshold value provided by OpenCV is used for this purpose.

Gear	Threshold Value
Gear1	7
Gear2	8
Gear3	7
Gear4	8

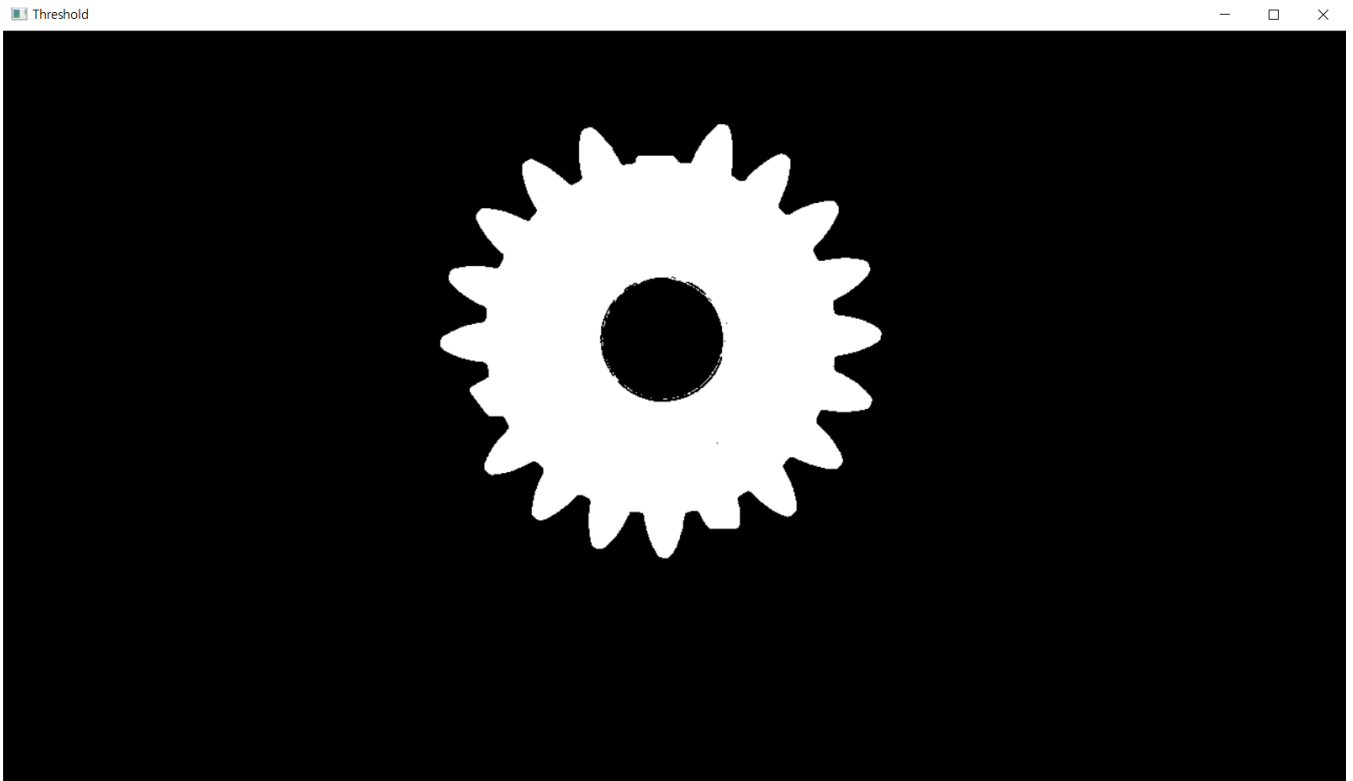


Figure5. Gear1.jpg after threshold

2. To use function `minEnclosingCircle(InputArray points, Point2f& center, float& radius)`, find the external contour of the whole gear image.

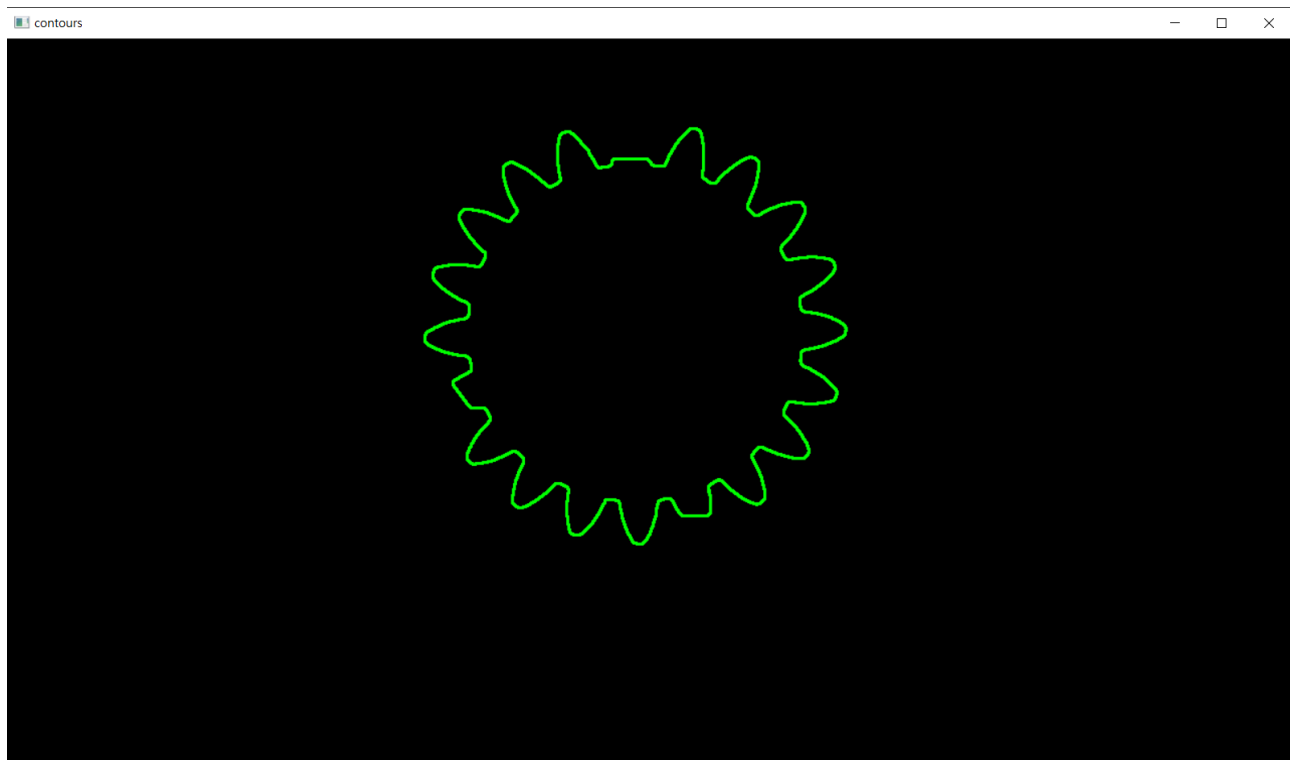


Figure6. The external contour of Gear1.jpg

3. Find the Circumscribed circle by using function 'minEnclosingCircle(InputArray points, Point2f& center,float& radius)'

minEnclosingCircle(InputArray points, Point2f& center,float& radius)

4. Calculate the diameter of the Circumscribed circle

5. In order to extract only the teeth from the entire gear, the tooth length is computed using the formula:
tooth length = module \times 2.25.

$$m = \frac{d_{outer}}{n_{teeth} + 2} \quad m * 2.25 = l_{teeth}$$

Figure 7. Finding the length of the teeth using the module of the gear

6. Calculate the root diameter using module and length of the teeth

7. To isolate the gear teeth, the inscribed circle is removed from the original image and erode the tooth image to accurately draw the contour of each tooth

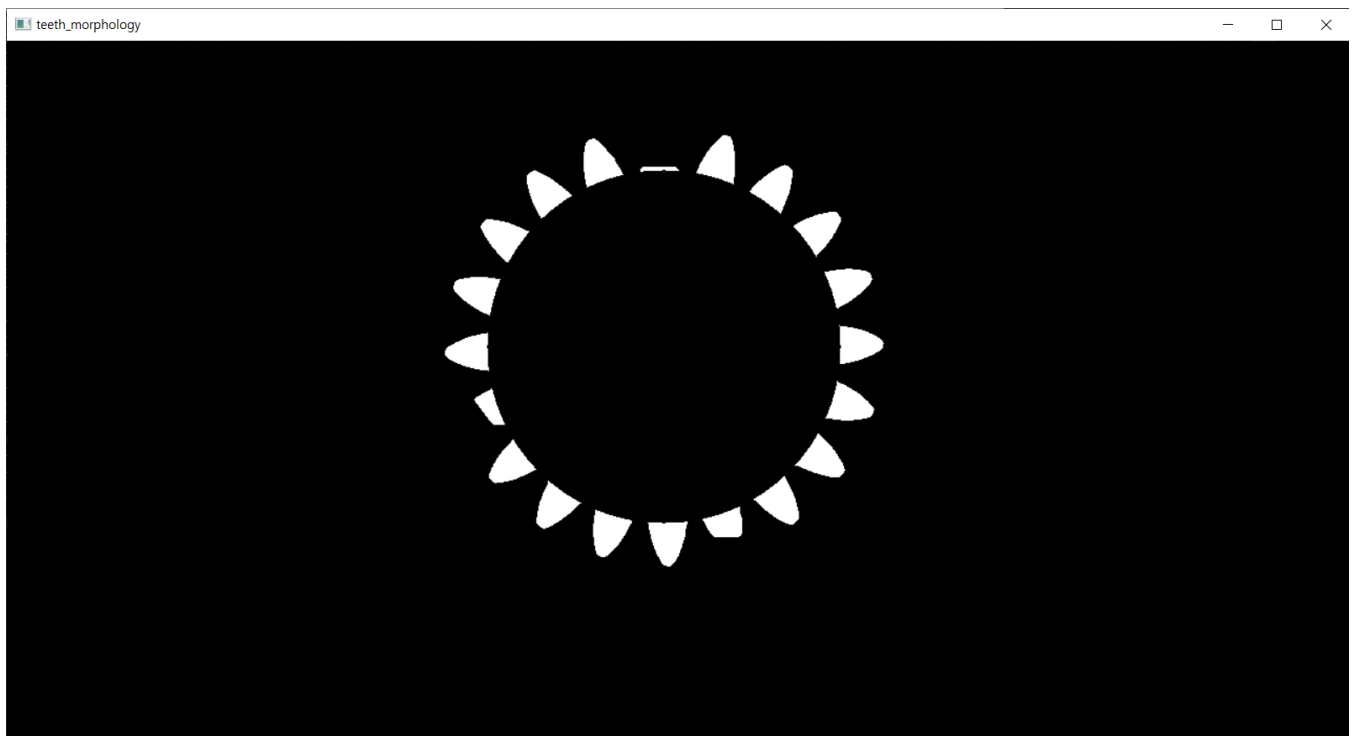


Figure 8. The result of erosion on teeth

8. Calculate the average of each teeth and use them as a reference to determine whether a tooth is defective or not.

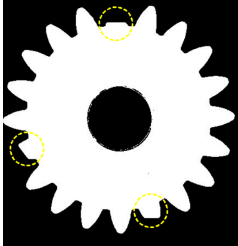
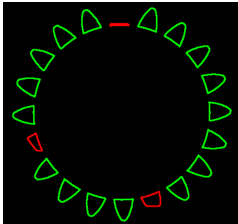
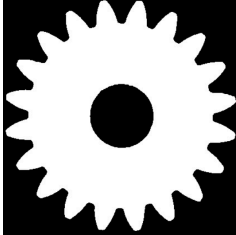
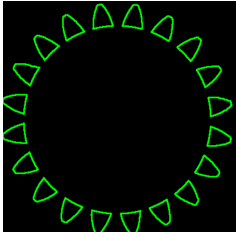
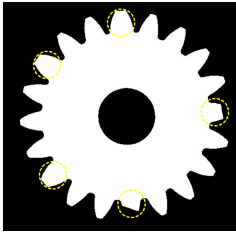
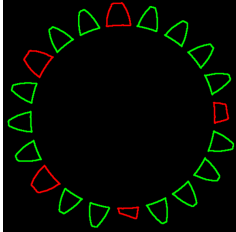
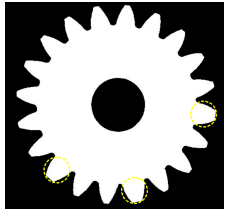
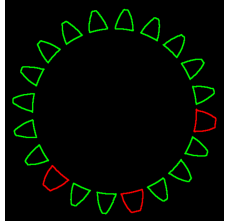
9. A defect inspection is performed on the gear teeth.

- If a teeth area is smaller than $(Avg-200)$ or bigger than $(Avg+200)$, it is a defective teeth. --> red color
- else, the teeth has no problem. --> green color

10. Print the other values (Number of teeth, Number of defective teeth, Diameter of the gear, Quality Inspection (Pass or Fail))

Result and Discussion

1. Final Result

	Sample#1	Sample#2	Sample#3	Sample #4
Output Images	 />>  />>	 />>  />>	 />>  />>	 />>  />>
Teeth numbers	20	20	20	20
Avg. Teeth Area	916.775	975.3	1295.75	1361.53
Defective Teeth	3	0	5	3
Quality	Fail	Pass	Fail	FAIL

2. Discussion

The purpose of this study was to identify defective teeth in specific gear images. As a result, Gear 1 had 3 defective teeth, Gear 2 had none, Gear 3 had 5, and Gear 4 had 3 defective teeth. Using the algorithm described above, the number of defective teeth was accurately determined.

Conclusion

In this study, a tooth was considered defective if its area was smaller than the average minus 200 or larger than the average plus 200. This criterion was consistently applicable to all gear images.

Apendix

```
#include<iostream>
#include<opencv2/opencv.hpp>
#include <iomanip>

using namespace std;
using namespace cv;

// 함수 선언
void finding_defective_teeth(Mat& std);

int main()
{

    // 각 기어 이미지를 불러온다.
    Mat gear1 = imread("Gear1.jpg", IMREAD_GRAYSCALE);
    Mat gear2 = imread("Gear2.jpg", IMREAD_GRAYSCALE);
    Mat gear3 = imread("Gear3.jpg", IMREAD_GRAYSCALE);
    Mat gear4 = imread("Gear4.jpg", IMREAD_GRAYSCALE);

    //이미지가 로드되지 않을 시에 에러 메시지를 출력한다.
    if (gear1.empty() || gear2.empty() || gear3.empty() || gear4.empty()) {
        cerr << "One or more images failed to load. Check the file paths." << endl;
        return -1;
    }

    //

    /*
    gear1~4에 대해 결함이빨 개수와 위치를 출력한다.

    !!기어 이미지에 대해 스페이스바를 누를 시 다음 이미지가 출력된다!!!

    */
    cout << "Gear1" << endl;
    finding_defective_teeth(gear1);

    cout << "Gear2" << endl;
```



```

finding_defective_teeth(gear2);

cout << "Gear3" << endl;
finding_defective_teeth(gear3);

cout << "Gear4" << endl;
finding_defective_teeth(gear4);

return 0;
}

void finding_defective_teeth(Mat& std)
{
    //결손이빨의 위치를 점선으로 표시하기 위해 입력된 이미지를 컬러로 변환하여 새로운 행렬에 저장한다.
    Mat std_colored;
    cvtColor(std, std_colored, COLOR_GRAY2BGR);

    //OpenCV에서 제공하는 최적의 threshold 임계값을 제공하는 함수를 사용한다.
    //threshold value: gear1-->7 gear2-->8 gear3-->7 gear4-->8
    Mat after_threshold;
    double otsu_thresh_val = threshold(std, after_threshold, 0, 255, THRESH_BINARY |
cv::THRESH_OTSU);

    //이빨의 개수를 지정한다.
    int teeth_number = 20;

    //외접원을 구하기 위해 기어 가장 바깥쪽의 컨투어를 찾는다.
    vector<vector<Point>> contours_whole;
    findContours(after_threshold, contours_whole, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

    //외접원의 파라미터로 지정될 변수들을 선언한다.
    Point2f center;
    float radius;

    //2D 이미지에서 가장 바깥쪽에 있는 부분을 기준으로 하여 외접원을 구한다.
    int maxIndex = 0;
    double maxArea = 0;
    for (int i = 0; i < contours_whole.size(); i++) {
        double area = contourArea(contours_whole[i]);
        if (area > maxArea) {
            maxArea = area;
            maxIndex = i;
        }
    }
    minEnclosingCircle(contours_whole[maxIndex], center, radius);

    //기어 전체에서 이빨만 남기기 위해 이빨의 길이를 구한다. (이빨길이=모듈*2.25)
    double module = (2 * radius) / (teeth_number + 2);

```

```

double teeth_length = module * 2.25;
float inner_radius = radius - teeth_length;

//root diameter를 구한다.
double root_diameter = module * teeth_number - 2 * teeth_length + 2 * module;

//본 이미지에서 내부원을 빼 이빨만 남긴다.
Mat teeth = after_threshold.clone();
circle(teeth, center, inner_radius, Scalar(0), -1);

//이빨이 서로 연결되어있는 등의 노이즈가 있을 시 계산이 정확해지지 않기에 erode를 진행한다.
Mat teeth_morphology;
Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3));
erode(teeth, teeth_morphology, kernel);

// 이빨에 대한 컨투어를 구한다.
vector<vector<Point>> contours_teeth;
findContours(teeth_morphology, contours_teeth, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

//이빨 면적의 평균을 구한다.
float teeth_area_sum = 0;
for (const auto& contour : contours_teeth) {
    teeth_area_sum += contourArea(contour);
}
float teeth_area_average = teeth_area_sum / teeth_number;

//이빨의 컨투어를 그려낼 행렬을 선언한다.
Mat drawing_with_text = Mat::zeros(std::size(), CV_8UC3);
Mat drawing = Mat::zeros(std::size(), CV_8UC3);

//결함이빨의 개수를 세기 위한 변수를 지정한다.
int defective_teeth_count = 0;

//이빨 하나하나에 대해 결함검사를 한다.
for (int i = 0; i < contours_teeth.size(); i++) {
    double teeth_area = contourArea(contours_teeth[i]);

    //만약 특정 이빨의 면적이 평균-200보다 작거나, 평균+200보다 클 경우 결함이 있다고 판단한다.
    if ((0 < teeth_area && teeth_area < (teeth_area_average - 200)) || (teeth_area >
(teeth_area_average + 200))) {

        // 텍스트를 넣을 이미지와 빨 이미지 모두 결함 이빨은 빨간색으로 컨투어를 그린다.
        drawContours(drawing, contours_teeth, i, Scalar(0, 0, 255), 2);
        drawContours(drawing_with_text, contours_teeth, i, Scalar(0, 0, 255), 2);

        // 텍스트로 면적을 표기한다.
        putText(drawing_with_text, to_string((int)teeth_area), contours_teeth[i][0],
FONT_HERSHEY_SIMPLEX, 0.5, Scalar(255, 255, 255), 1);
    }
}

```

```

// 노란 점선으로 컬러 이미지 위에 결손이빨의 위치를 표시한다.
Moments M = moments(contours_teeth[i]);
int cx = int(M.m10 / M.m00);
int cy = int(M.m01 / M.m00);
Point center_teeth(cx, cy);
for (int angle = 0; angle < 360; angle += 20) {
    ellipse(std_colored, center_teeth, Size(30, 30), 0, angle, angle + 10,
Scalar(0, 255, 255), 2);
}

//결함이빨의 개수를 추가한다.
defective_teeth_count++;
}

//평균-200보다 크고, 평균+200보다 작은 이빨은 정상이빨로 처리한다.
else if ((teeth_area > (teeth_area_average - 200)) && (teeth_area <
(teeth_area_average + 200))) {

    // 텍스트를 넣을 이미지와 빨 이미지 모두 정상 이빨은 초록색으로 컨투어를 그린다.
    drawContours(drawing, contours_teeth, i, Scalar(0, 255, 0), 2);
    drawContours(drawing_with_text, contours_teeth, i, Scalar(0, 255, 0), 2);

    //이빨의 면적을 기입한다.
    putText(drawing_with_text, to_string((int)teeth_area), contours_teeth[i][0],
FONT_HERSHEY_SIMPLEX, 0.5, Scalar(255, 255, 255), 1);
}
}

//결손부위가 표시된 컬러이미지를 표시한다.
namedWindow("gear", WINDOW_AUTOSIZE);
imshow("gear", std_colored);

//결손이빨과 일반 이빨의 색깔을 구분한 컨투어 이미지를 표시한다.
namedWindow("contours with text", WINDOW_AUTOSIZE);
imshow("contours with text", drawing_with_text);

namedWindow("contours", WINDOW_AUTOSIZE);
imshow("contours", drawing);

//이빨 개수, 평균, 결손이빨 개수, 기어의 root diameter를 출력한다.
cout << "Teeth numbers: " << teeth_number << endl;
cout << "Avg. Teeth Area: " << teeth_area_average << endl;
cout << "Defective Teeth: " << defective_teeth_count << endl;
cout << "Diameter of the gear:" << root_diameter << endl;

// 퀄리티 검사를 출력한다.
if (defective_teeth_count > 0)
    cout << "Quality : Fail" << endl;

```

```
else
    cout << "Quality : Pass" << endl;
cout << endl;

waitKey(0);
}
```