DLIP : LAB1

Author : HanMinung

Student ID : 21800773

School of Mechanical and Control Engineering

LAB : Grayscale and segmentation

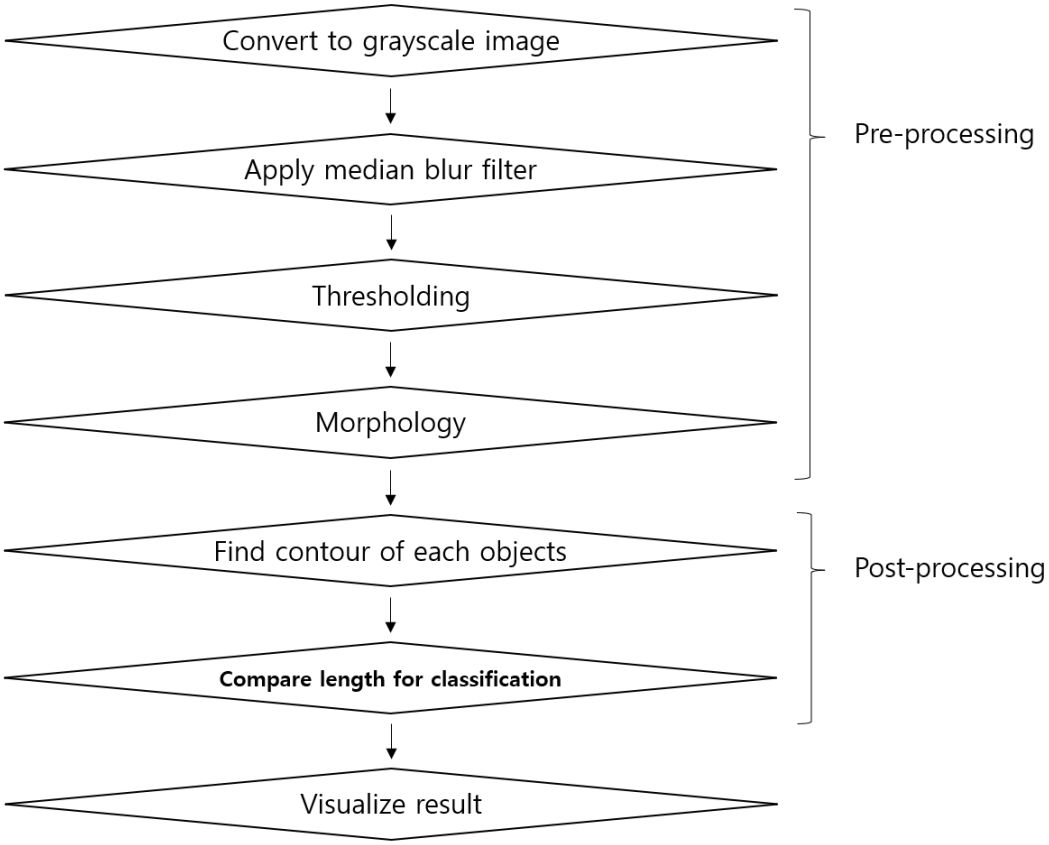
Date : 2023.03.24

---

1. Introduction

- The purpose of this experiment was to develop a system that can automatically distinguish bolts and nuts of different sizes and count each type accurately using classical image processing techniques. This system can be useful in various manufacturing and assembly processes where different types of bolts and nuts are used, and their sizes need to be identified and counted quickly and accurately.
- Object to classify
    - M5 BOLT : 5 [EA]
    - M6 BOLT : 3 [EA]
    - M5 Hexa Nut : 4 [EA]
    - M6 Hexa Nut : 4 [EA]
    - M5 Rect Nut : 5 [EA]
- Source image

2. Flow chart



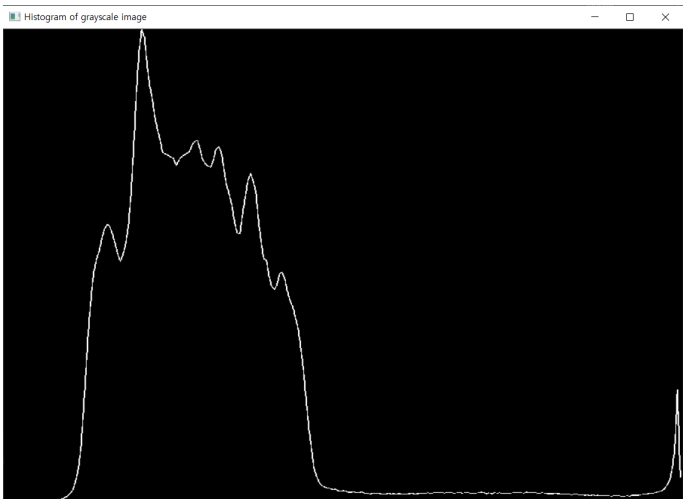
3. Processing

3.1. Preprocessing

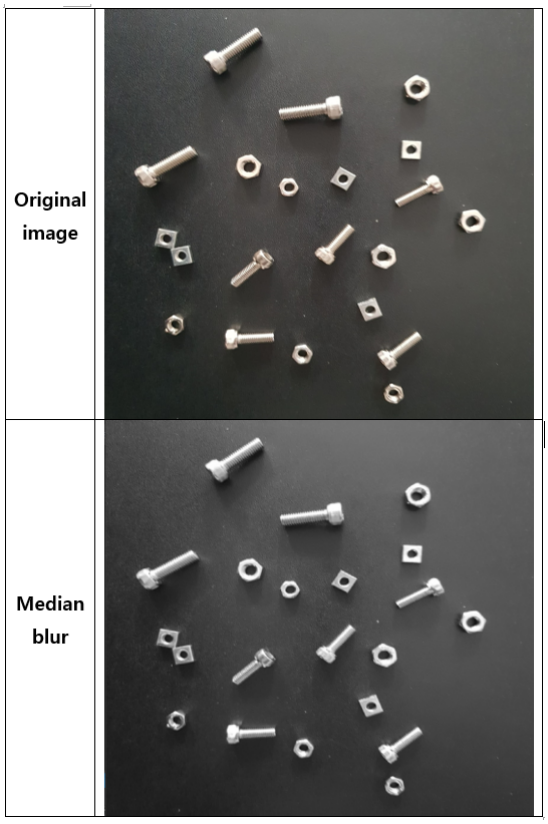
Before extracting features of each objects, it is proper to process given data in advance to reduce noise and unnecessary points which can return inappropriate results after we extract some features of source images. It is essential to identify the type of noise and select a proper filter with an appropriate kernel size, and it is also important to apply techniques such as morphology to the blurred image to avoid problems in post-processing.

3.1.1. Filter application

- Figure below shows the histogram of grayscale image.

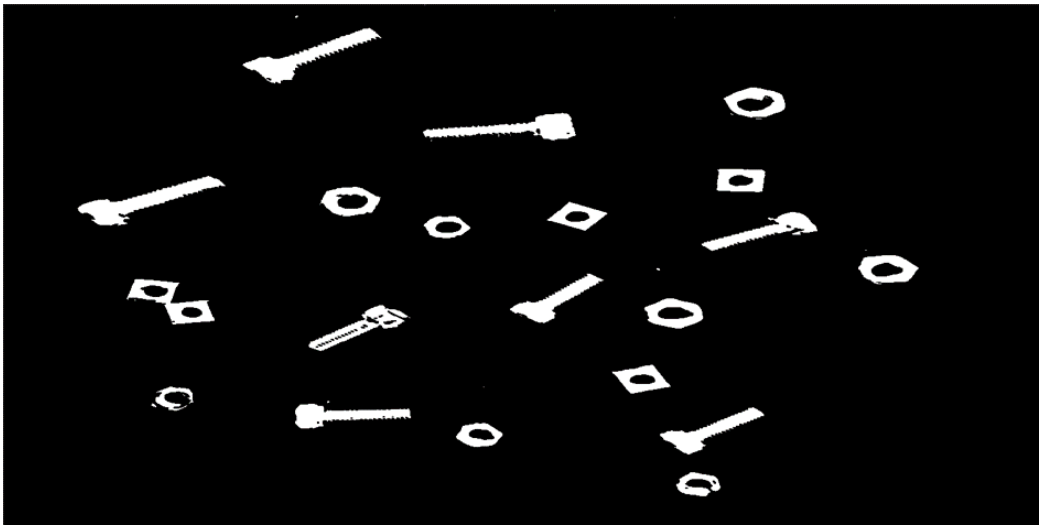


- You can confirm that salt and pepper noise is observed in the grayscale image. Generally, to remove salt and pepper noise, a median filter is applied. Accordingly, in this lab, a median filter of kernel size 5 was applied for proper preprocessing of noise. Increasing the kernel size to a proper value (**applied with 5 in this lab**) can help to smooth out the noise more effectively, resulting in a cleaner and clearer image. This is because a larger kernel can capture more neighboring pixels, allowing for a more robust estimation of the median value. The original image and the result image which is applied with median filter are as follows :



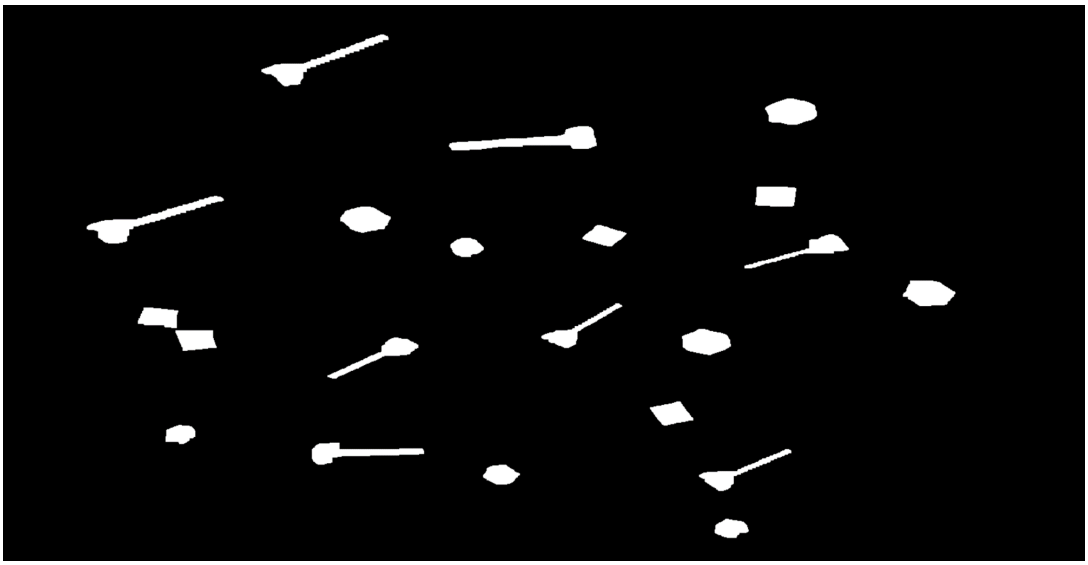
3.1.2. Threshold

In order to accurately identify objects in post-processing, it is essential to select an appropriate thresholding value to delete unnecessary parts. Accordingly, a trackbar was created to find an appropriate threshold value. Later, by applying the morphology technique, all points remaining after the threshold can be removed, so I found a value that leaves the shape of the object as much as possible, and the following value that I found is 118. Following result is as follows :



3.1.3. Morphology

In order to do accurate object segmentation and classification later, first, the work of filling the inside of the nut was done first. In the process, 13 dilation techniques were applied. When the inside of the nut was all filled, a total of 23 erode techniques were applied to remove the noise point and to separate the two nuts attached to the lower left, and the desired result was obtained. The above process is all to fill out the inside of the nut first and separating the two attached objects, and the following result is as follows.



3.2. Post processing : classification

3.2.1. Class determination

After the whole process of preprocessing for proper classification, I used the information of contour length to classify each objects. After obtaining the information on the length of each contour, visualizing it first, I checked whether the classification for each class (total of 5) worked well first. As can be seen in the figure below, it can be seen that each class is sufficiently distinguished with only length information, and the process of assigning class numbers by dividing conditions according to the information is as follows. Each class is stored with int value with macro in the code. The values that each class obtained are as follows :

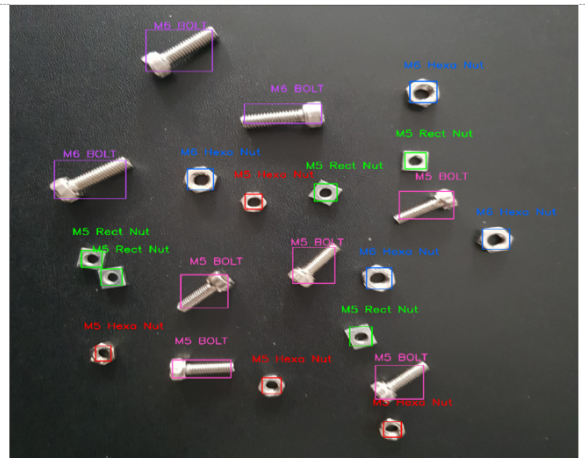
M6 Bolt	0
B5 Bolt	1
M6 Hex Nut	2
M5 Hex Nut	3
M5 Rect Nut	4

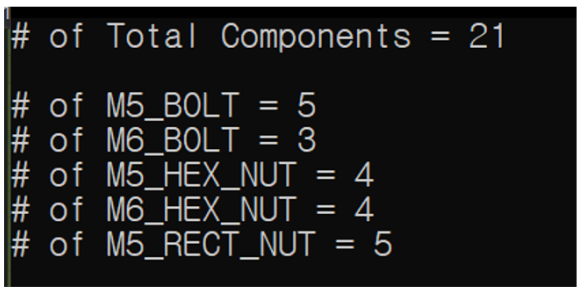
```
int detType(int len) {

    if (len > 450)                      classType = M6BOLT;
    if (len < 450 && len > 300)         classType = M5BOLT;
    if (len > 210 && len < 250)         classType = M6HEXNUT;
    if (len > 180 && len < 210)         classType = M5RECTNUT;
    if (len > 100 && len < 180)         classType = M5HEXNUT;

    return classType;
}
```

5. Result





All objects were properly classified with the method introduced above. Since I drew rectangle with the calculated contours, the shape cannot be matched perfectly as we can see in the left figure above. However, it is possible to confirm that the post-processing classification worked well with appropriate pre-processing techniques, and that each class was well classified.

6. Discussion and analysis

By applying different methods to the objects in an image, it becomes clear how important preprocessing techniques like blur, thresholding, and morphology can be. When preprocessing is done well, it makes it easier to extract and classify important features in the image. In this lab, we encountered a challenge when two nuts that were connected to each other were recognized as one contour, which made counting them difficult. However, we were able to solve this issue by using appropriate morphology techniques to separate the nuts. Furthermore, as the presence of various environmental factors or variables can produce different outcomes than those achieved through traditional image processing, it is advisable to classify any supplementary data obtained through techniques like deep learning.

7. Appendix

- Defined macro

```
#define kernelSize              5

#define threshVal               118
#define maxBinaryVal            255
#define threshType              0

#define FORLOOP(i, Final)       for(int i = 0; i < Final ; i++)

#define M6BOLT                  0
#define M5BOLT                  1
#define M6HEXNUT                2
#define M5HEXNUT                3
#define M5RECTNUT               4
```

- Variable, Function declaration

```
int classType = 0;

Mat src, dst, srcGray, dstBlur, dstThresh, dstMorph, dstContour;
Mat preProcess;

Scalar color1(255, 74, 200);        // M5_RECT_NUT : PURPLE
Scalar color2(200, 74, 255);        // M5_HEX_NUT : PINK
Scalar color3(255, 100, 0);         // M6_HEX_NUT : BLUE
Scalar color4(0, 255, 0);           // M5_BOLT : GREEN
Scalar color5(0, 0, 255);           // M6_BOLT : RED

// Initialization of # of each component
int Bolt_M5 = 0;
int Bolt_M6 = 0;
int M6_HEXA_NUT = 0;
int M5_SQUARE_NUT = 0;
int M5_HEXA_NUT = 0;
```

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
std::vector<std::vector<Point>> M6_BOLT;
std::vector<std::vector<Point>> M5_BOLT;
std::vector<std::vector<Point>> M5_HEX_NUT;
std::vector<std::vector<Point>> M6_HEX_NUT;
std::vector<std::vector<Point>> M5_RECT_NUT;


// Global variables for Morphology
int element_shape = MORPH_RECT;
int n = 3;
Mat element = getStructuringElement(element_shape, Size(n, n));


void preProcessing(Mat src);
void visualizing(int Idx,int Type);
int detType(int len);
```

- Main statement

```
void main() {

    src = imread("Lab_GrayScale_TestImage.jpg", IMREAD_COLOR);

    // Check image loading
    checkLoad(src);

    // image preprocessing
    preProcessing(src);

    //finding contour
    findContours(dstMorph, contours, hierarchy, RETR_TREE,
CHAIN_APPROX_SIMPLE);

    for (int i = 0; i < contours.size(); i++) {

        double len = arcLength(contours[i], true);

        Moments mu = moments(contours[i], false);
        Point centroid = Point(mu.m10 / mu.m00 + 30, mu.m01 / mu.m00 - 50);

        // Determine type
        classType = detType(len);

        // Drawing contours
        visualizing(i, classType);

    }
```

```
    int totalNum = Bolt_M5 + Bolt_M6 + M5_HEXA_NUT + M6_HEXA_NUT +
M5_SQUARE_NUT;

    std::cout << "# of Total Components = "<< totalNum << "\n" <<
std::endl;
    std::cout << "# of M5_BOLT = " << Bolt_M5 << std::endl;
    std::cout << "# of M6_BOLT = " << Bolt_M6 << std::endl;
    std::cout << "# of M5_HEX_NUT = " << M5_HEXA_NUT << std::endl;
    std::cout << "# of M6_HEX_NUT = " << M6_HEXA_NUT << std::endl;
    std::cout << "# of M5_RECT_NUT = " << M5_SQUARE_NUT << std::endl;


    namedWindow("Contoured image", WINDOW_FREERATIO);
    imshow("Contoured image", src);

    waitKey(0);

}
```

- Function : preprocessing

```
void preProcessing(Mat src) {

    // gray scale
    cvtColor(src, srcGray, CV_BGR2GRAY);

    Size size = srcGray.size();

    plotHist(srcGray, "Histogram of grayscale image", size.width,
size.height);

    // Median blur with kernel size 5
    medianBlur(srcGray, dstBlur, kernelSize);

    // Threshold process
    threshold(dstBlur, dstThresh, threshVal, maxBinaryVal, threshType);

    // Morphology process
    dilate(dstThresh, dstMorph, element);

    dilate(dstMorph, dstMorph, element, Point(-1,-1), 13);
    erode(dstMorph, dstMorph, element, Point(-1,-1), 23);
    dilate(dstMorph, dstMorph, element, Point(-1, -1), 2);

}
```

- Function : detType

```
int detType(int len) {

    if (len > 450)                      classType = M6BOLT;
    if (len < 450 && len > 300)         classType = M5BOLT;
    if (len > 210 && len < 250)         classType = M6HEXNUT;
    if (len > 180 && len < 210)         classType = M5RECTNUT;
    if (len > 100 && len < 180)         classType = M5HEXNUT;

    return classType;
}
```

- Function : Visualizing

```
void visualizing(int Idx, int Type) {

    if (classType == M6BOLT) {

        cv::Rect BoundingBox = cv::boundingRect(contours[Idx]);
        cv::rectangle(src, BoundingBox, color1, 2);
        Moments mu = moments(contours[Idx], false);
        Point centroid = Point(mu.m10 / mu.m00 - 50, mu.m01 / mu.m00 - 80);
        putText(src, "M6 BOLT", centroid, FONT_HERSHEY_SIMPLEX, 1, color1,
2);

        Bolt_M6++;
    }


    if (classType == M5BOLT) {

        cv::Rect BoundingBox = cv::boundingRect(contours[Idx]);
        cv::rectangle(src, BoundingBox, color2, 2);
        Moments mu = moments(contours[Idx], false);
        Point centroid = Point(mu.m10 / mu.m00 - 50, mu.m01 / mu.m00 - 80);
        putText(src, "M5 BOLT", centroid, FONT_HERSHEY_SIMPLEX, 1, color2,
2);

        Bolt_M5++;
    }


    if (classType == M6HEXNUT) {

        cv::Rect BoundingBox = cv::boundingRect(contours[Idx]);
        cv::rectangle(src, BoundingBox, color3, 2);
        Moments mu = moments(contours[Idx], false);
        Point centroid = Point(mu.m10 / mu.m00 - 50, mu.m01 / mu.m00 - 80);
        putText(src, "M6 Hexa Nut", centroid, FONT_HERSHEY_SIMPLEX, 1,
color3, 2);

        M6_HEXA_NUT++;
```

```
    }


    if (classType == M5RECTNUT) {

        cv::Rect BoundingBox = cv::boundingRect(contours[Idx]);
        cv::rectangle(src, BoundingBox, color4, 2);
        Moments mu = moments(contours[Idx], false);
        Point centroid = Point(mu.m10 / mu.m00 - 50, mu.m01 / mu.m00 - 80);
        putText(src, "M5 Rect Nut", centroid, FONT_HERSHEY_SIMPLEX, 1,
color4, 2);

        M5_SQUARE_NUT++;
    }

    if (classType == M5HEXNUT) {

        cv::Rect BoundingBox = cv::boundingRect(contours[Idx]);
        cv::rectangle(src, BoundingBox, color5, 2);
        Moments mu = moments(contours[Idx], false);
        Point centroid = Point(mu.m10 / mu.m00 - 50, mu.m01 / mu.m00 - 80);
        putText(src, "M5 Hexa Nut", centroid, FONT_HERSHEY_SIMPLEX, 1,
color5, 2);

        M5_HEXA_NUT++;
    }
}
```