**Part 1: Counting different objects in gray scale images**

แนวคิด

1. แปลงภาพ Grayscale เป็น Binary image โดยใช้ Otsu Thresholding
2. ทำ Cleaning ด้วย  Morphology operation เพื่อทำการลบจุดเล็กๆ บนภาพ
3. หา Contour เพื่อแยกวัตถุออกจากพื้นหลัง
4. สร้าง convex hull จาก Contour เพื่อให้เกิดความต่อเนื่องของเส้นขอบวัตถุ
5. หาค่า circularity จาก convex hull ที่สร้างขึ้น
6. กำหนดค่า threshold ของ circularity เพื่อแบ่งแยกประเภทของวัตถุ

#include <opencv2/opencv.hpp>

#include <iostream>

#include <math.h>

using namespace cv;

using namespace std;

int main()

{

Mat image;

image = imread("C:/Users/TETE/Desktop/OpenCVFiles/nut1.jpg", IMREAD\_GRAYSCALE);

if (image.empty()) // Check for invalid input

{

cout << "Could not open or find the image" << std::endl;

return -1;

}

namedWindow("Image", 1);

imshow("Image", image);

Mat binary;

threshold(image, binary, 0, 255, THRESH\_BINARY\_INV | THRESH\_OTSU);

namedWindow("BINARY\_INV", 1);

imshow("BINARY\_INV", binary);

Mat opened\_image;

//ทำ Cleaning ด้วย Morphology operation แบบ Closing เพื่อทำการลบจุดเล็กๆ บนภาพ

Mat SE(10,10, CV\_8U, Scalar(1));

morphologyEx(binary, opened\_image, MORPH\_CLOSE, SE);

namedWindow("opened\_image", 1);

imshow("opened\_image", opened\_image);

vector<vector<Point>> contours;

findContours(opened\_image, contours, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

Mat convexHullImg = image.clone();

vector<vector<Point>> hull(contours.size());

for (size\_t i = 0; i < contours.size(); i++)

{

//แปลง contour แต่ละอันให้เป็น convex hull

convexHull(contours[i], hull[i], true, true);

int hullCnt = hull[i].size();

Point pt0 = hull[i][hullCnt - 1];

//วาด convex hull

for (size\_t k = 0; k < hullCnt; k++)

{

Point pt1 = hull[i][k];

line(convexHullImg, pt0, pt1, Scalar(0, 255, 0), 1, LINE\_AA);

pt0 = pt1;

}

}

vector<double> len(hull.size());

vector<double> areas(hull.size());

vector<double> circularity(hull.size());

for (size\_t i = 0; i < hull.size(); i++)

{

len[i] = arcLength(hull[i], true);

areas[i] = contourArea(hull[i]);

// หา circularity ของ convex hull

circularity[i] = (4.0\*CV\_PI \*areas[i]) / (len[i] \* len[i]);

// แสดงค่า circularity ของแต่ละ component

putText(convexHullImg, to\_string(circularity[i]), contours[i][0], 2, 0.4, Scalar(0, 255, 0));

}

namedWindow("convexHull", 1);

imshow("convexHull", convexHullImg);

Mat result;

cvtColor(image, result, COLOR\_GRAY2BGR);

uint nuts = 0;

uint bolts = 0;

//กำหนดค่า circularity threshold

double circularityThreshold = 0.9;

for (size\_t i = 0; i < hull.size(); i++)

{

//ไม่ให้ component ที่มีขนาดเล็กมากเป็นวัตถุ

if (areas[i] < 100)continue;

Rect bb = boundingRect(hull[i]);

int sx = bb.tl().x; int sy = bb.tl().y;

int ex = bb.br().x - 1; int ey = bb.br().y - 1;

int cx = (sx + ex) / 2; int cy = (sy + ey) / 2;

//แสดงตำแหน่งของวัตถุ

rectangle(result, bb, circularity[i] >= circularityThreshold ? Scalar(0, 255, 0) : Scalar(0, 0, 255), 2);

drawMarker(result, Point(cx, cy), circularity[i] >= circularityThreshold ? Scalar(0, 255, 0) : Scalar(0, 0, 255));

//ถ้าค่า circularity มากกว่า threshold ให้วัตถุนั้นเป็น nut ถ้าไม่ใช่ให้เป็น bolts

if (circularity[i] >= circularityThreshold)

nuts++;

else

bolts++;

}

putText(result, "Nut = " + to\_string(nuts),Point(0,40),1,3, Scalar(0, 255, 0),3);

putText(result, "Bolt = " + to\_string(bolts), Point(0, 40+40), 1, 3, Scalar(0, 0, 255), 3);

namedWindow("Result", 1);

imshow("Result", result);

waitKey(0);

destroyAllWindows();

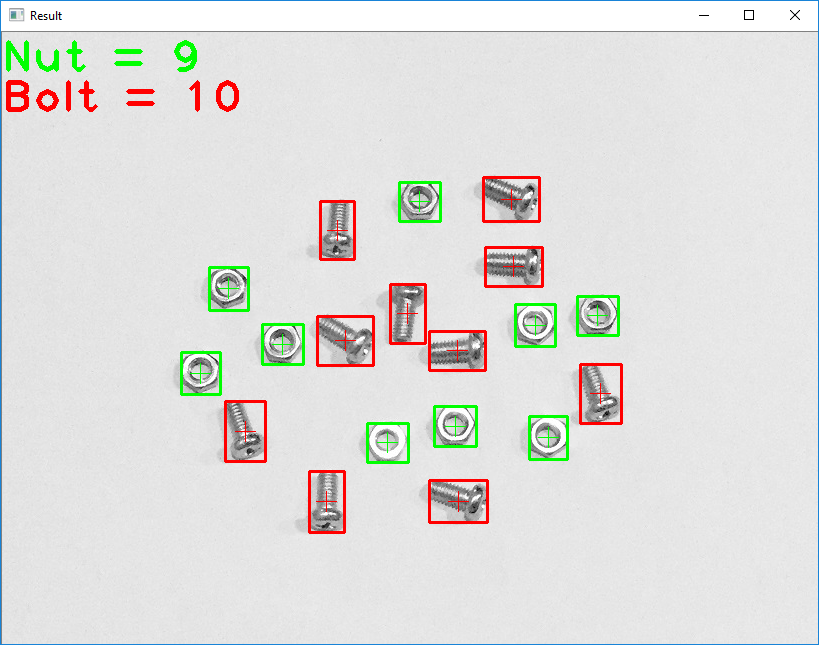
return 0;

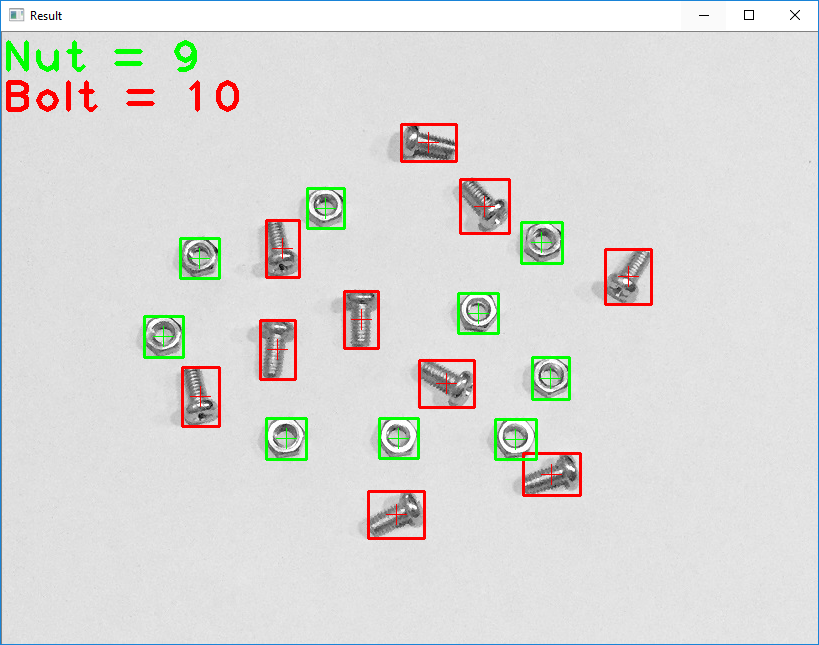
}

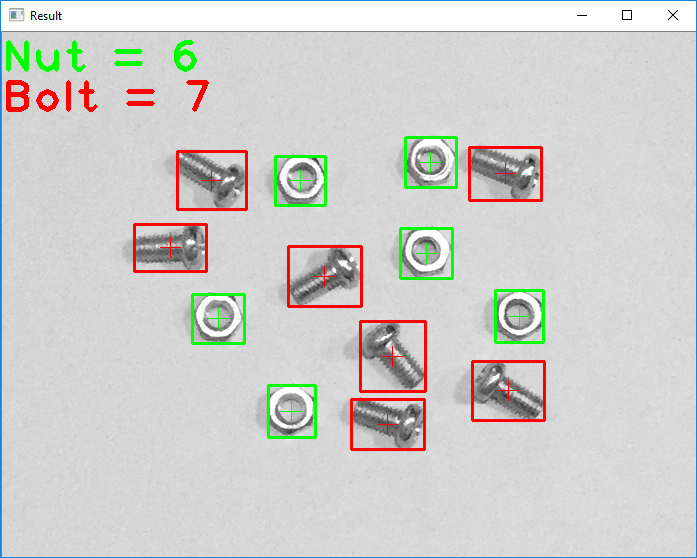
circularity

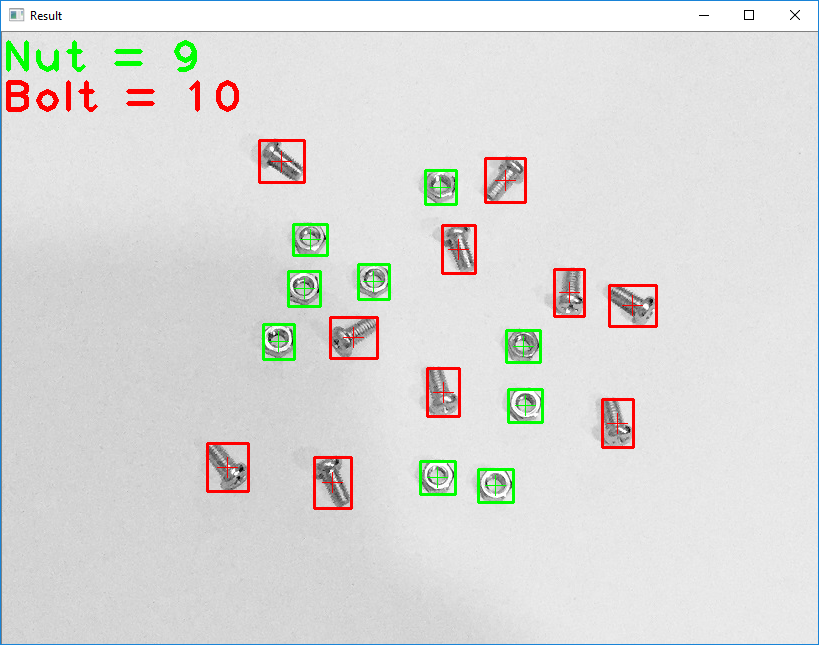


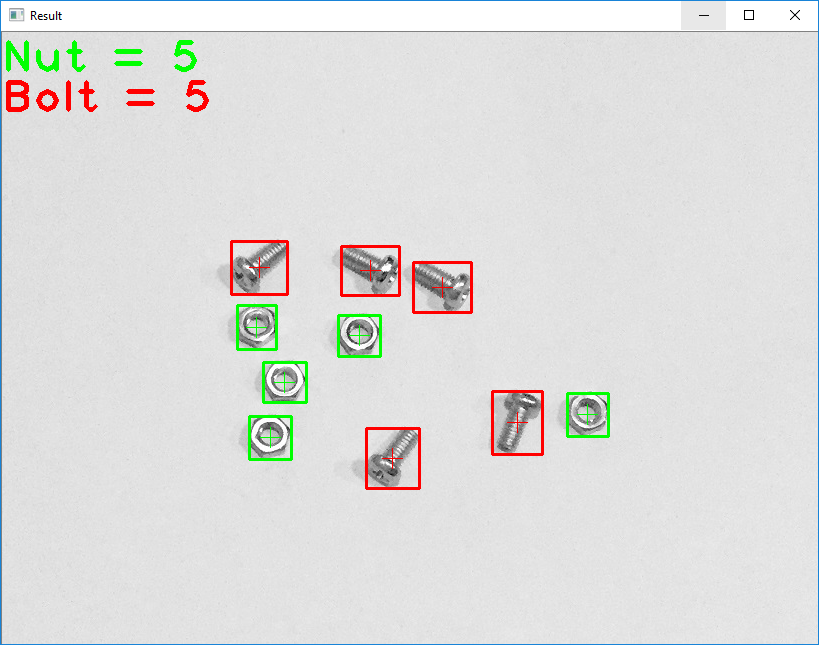
**ผลลัพธ์**











**Part 2: Counting objects with different colors in images**

**แนวคิด**

แยกวัตถุสีต่างกันโดยการใช้ค่าเฉลี่ย Hue ของวัตถุในภาพ HLS โดย map ช่วง Hue ของวัตถุในภาพเข้ากับสีต่างๆ จากนั้นใช้ค่าเฉลี่ย Luminance ของวัตถุในภาพ HLS เพื่อแยกวัตถุที่มีค่าสี Hue ใกล้เคียงกัน เช่น สีส้มกับสีน้ำตาล

**Part 3: Ball Detection**

**แนวคิด**

ใช้วิธีการแยกลูกบอลออกจากวัตถุอื่นๆ โดยใช้ขนาดพื้นที่และสีของลูกบอล โดยขั้นตอนแรกจะทำการกำหนดขนาดพื้นที่ของ component ที่จะสามารถเป็นลูกบอลได้ ซึ่งจะทำให้ได้ component เล็กๆจำนวนมากซึ่งรวม component ของลูกบอลอยู่ด้วย จากนั้นทำการแยกแยะลูกบอลจากค่าสีของ component โดยใช้ค่า Hue จากภาพ HLS ซึ่งจะทำให้ได้ component ส่วนเฉพาะที่เป็นลูกบอล

