Progress report

Name : 부준호 Bu Jun-ho
Student Number : 20140278
Group : 8th group
(advised by professor Park Young-jin)
Assigned Part : OpenCV

- First Submission

1. Work on OpenCV

① Installation of Ubuntu & Opencv

: For studying opencv sessions, I installed Ubuntu & Opencv on the notebook.

② Camera calibration

: By using ROS, I got the camera calibration parameters.

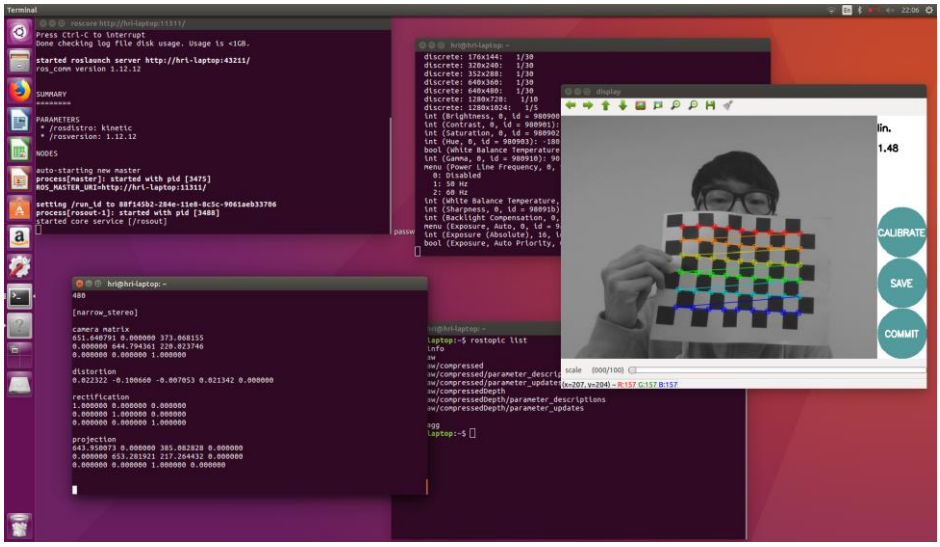


Figure 1 Calibration with ROS

```
480

[narrow_stereo]

camera matrix
651.640791 0.000000 373.068155
0.000000 644.794361 220.023746
0.000000 0.000000 1.000000

distortion
0.022322 -0.100660 -0.007053 0.021342 0.000000

rectification
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000

projection
643.950073 0.000000 385.082828 0.000000
0.000000 653.281921 217.264432 0.000000
0.000000 0.000000 1.000000 0.000000
```

Figure 2 Calibration Parameter

③ Basic opencv

: In opencv part, we will use QtCreator in which we can deal with the image so called image processing. So the first of all, we learned the basic of opencv.

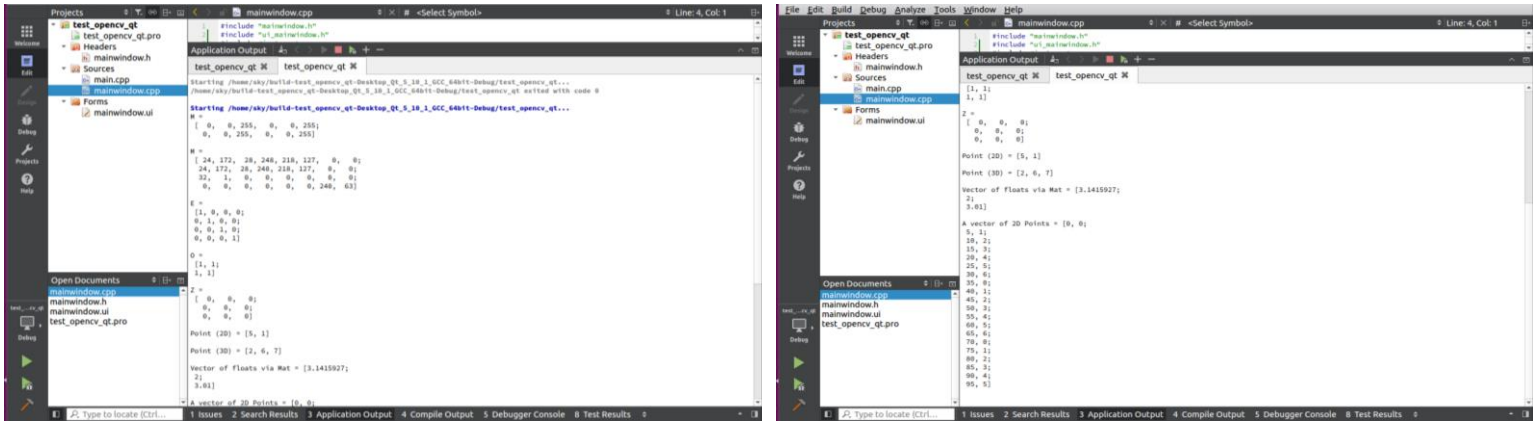


Figure 3 Basic Opencv

④ Image Processing

: By using Qtcreator, we learned about the image processing method and I applied various methods by myself.

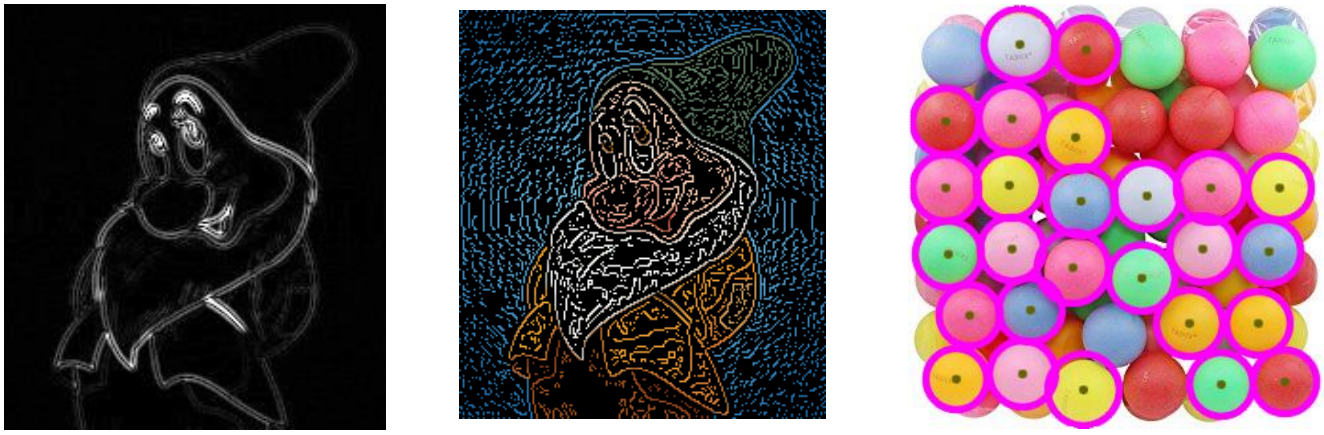


Figure 4 Image processing methods

- Second Submission

1. Execution ball detection code

In various situations, I executed the ball detection code in Opencv by using Qtcreator. That is, changing in brightness, angle of the webcam, the lowest detectable radius, the HSV range and so on.

We are able to know that red ball is detected with orange ball and blue ball is with green ball. Because we should succeed in case for detecting only red and blue balls, it's not big problem, we think.

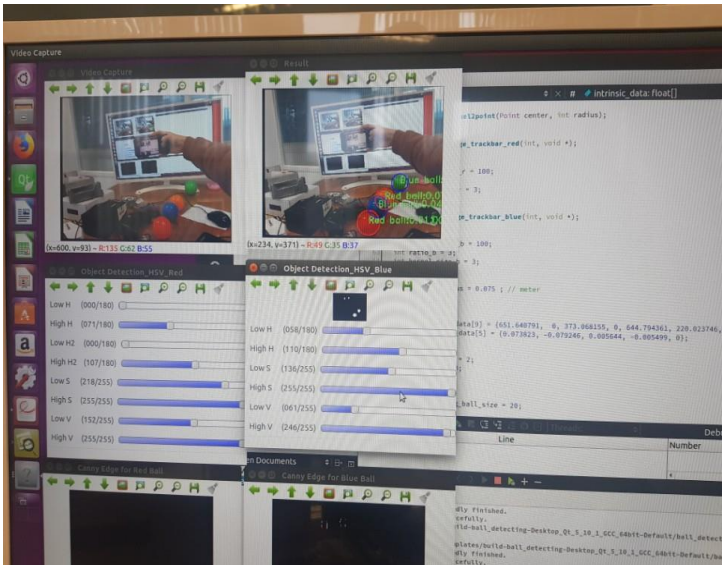


Figure 5 Ball detecting

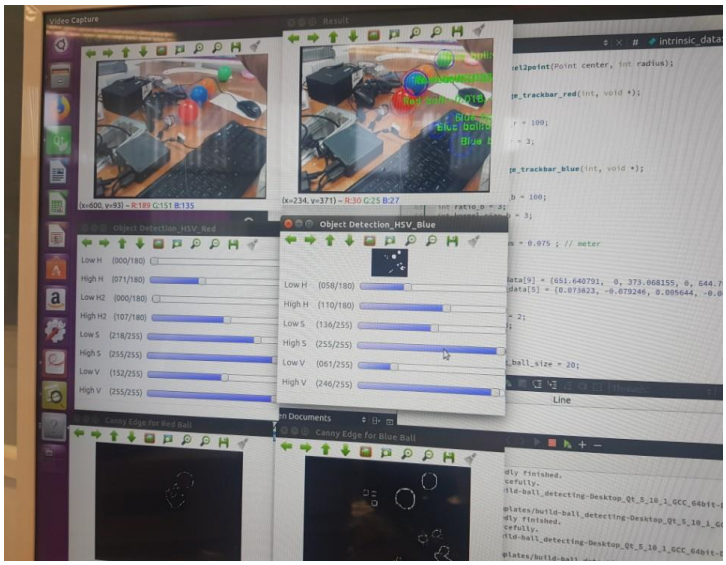


Figure 6 Suitable HSV range

I found the suitable HSV range for detecting. But, it's very uncomfortable for external environment such as brightness and the distance between webcam and ball. In the founded HSV range, the maximum detectable distance between webcam and balls is 1~1.5m. So, we should find the other methods for detecting, like pasting papers on webcam, using rplidar for going around the balls etc.

Discuss with ROS for integrating and observed the ROS code for detecting ball, but the answer did not come out.

2. Analyzing and searching the ball detection code

For more perfect ball detect, I searched the ball detection code using Opencv in Youtube, but there are no cases detecting the ball far away, about 4~5m from the webcam because our robot will start from the balls with that distance.

For more completeness, I analyzed the detecting code given by teaching assistant. From this process, we can change the image processing method and discover method to send the data to ROS.

```
//erode&dilate를 위한 morphOps 함수 불러옴
void morphOps(Mat &thresh);

//pixel 값을 position 값으로 바꾸는 함수 불러옴
vector<float> pixel2point(Point center, int radius);

//red image processing을 위한 함수(canny edge method) 불러옴
void on_canny_edge_trackbar_red(int, void *);

//canny edge를 실행시키기 위해 필요한 변수 값 지정
int lowThreshold_r = 100;
int ratio_r = 3;
int kernel_size_r = 3;

//blue image processing을 위한 함수(canny edge method) 불러옴
void on_canny_edge_trackbar_blue(int, void *);

//canny edge를 실행시키기 위해 필요한 변수 값 지정
int lowThreshold_b = 100;
int ratio_b = 3;
int kernel_size_b = 3;

//ball의 최소 detecting 가능 값을 정의
float fball_radius = 0.062 ; // meter

//camera calibration을 해주기 위해 각각의 웹캠 값을 넣어줌
Mat distCoeffs;
float intrinsic_data[9] = {596.497705,  0, 318.023172, 0, 565.010517, 271.337191, 0, 0, 1};
float distortion_data[5] = {0.073823, -0.079246, 0.005644, -0.005499, 0};

//글자 크기 설정
double fontScale = 2;
int thickness = 3;
String text ;

//최소 tracking ball size를 지정
int iMin_tracking_ball_size = 20;

//image processing한 결과를 출력하는 main 함수 실행
int main()
{
        //각각의 변수를 불러옴
    Mat frame, bgr_frame, hsv_frame,
        hsv_frame_red, hsv_frame_red1, hsv_frame_red2, hsv_frame_blue,
        hsv_frame_red_blur, hsv_frame_blue_blur,
        hsv_frame_red_canny, hsv_frame_blue_canny, result;

        //calibration parameter를 위한 matrix 생성
    Mat calibrated_frame;
    Mat intrinsic = Mat(3,3, CV_32FC1);
    Mat distCoeffs;
        //위에서 지정한 값을 calibration 값으로 지정
    intrinsic = Mat(3, 3, CV_32F, intrinsic_data);
    distCoeffs = Mat(1, 5, CV_32F, distortion_data);

        //빨간색, 파란색의 테두리의 끝점 값을 나타내는 변수 정의
    vector<Vec4i> hierarchy_r;
    vector<Vec4i> hierarchy_b;

        //빨간색, 파란색의 테두리 값을 저장
        vector<vector<Point> > contours_r;
    vector<vector<Point> > contours_b;

        //0번 카메라로 찍어서 출력
    VideoCapture cap(0);
```

Figure 7 Analysis of Detecting Code

```
        findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPLE );
        //use moments method to find our filtered object
        double refArea = 0;
        bool objectFound = false;
        if (hierarchy.size() > 0) {
                int numObjects = hierarchy.size();
                //if number of objects greater than MAX_NUM_OBJECTS we have a noisy filter
                if(numObjects<MAX_NUM_OBJECTS){
                        for (int index = 0; index >= 0; index = hierarchy[index][0]) {

                                Moments moment = moments((cv::Mat)contours[index]);
                                double area = moment.m00;

                //if the area is less than 20 px by 20px then it is probably just noise
                //if the area is the same as the 3/2 of the image size, probably just a bad filter
                //we only want the object with the largest area so we safe a reference area each
                                //iteration and compare it to the area in the next iteration.
                                if(area>MIN_OBJECT_AREA){

                                        Object object;

                                        object.setXPos(moment.m10/area);
                                        object.setYPos(moment.m01/area);
                                        object.setType(theObject.getType());
                                        object.setColor(theObject.getColor());

                                        objects.push_back(object);

                                        objectFound = true;

                                }else objectFound = false;
                        }
                        //let user know you found an object
                        if(objectFound ==true){
                                //draw object location on screen
                                drawObject(objects,cameraFeed,temp,contours,hierarchy);}

                }else putText(cameraFeed,"TOO MUCH NOISE! ADJUST FILTER",Point(0,50),1,2,Scalar(0,0,255),2);
        }
}

int main(int argc, char* argv[])
{
        //if we would like to calibrate our filter values, set to true.
        bool calibrationMode = true;

        //Matrix to store each frame of the webcam feed
        Mat cameraFeed;
        Mat threshold;
        Mat HSV;

        if(calibrationMode){
                //create slider bars for HSV filtering
                createTrackbars();
        }
        //video capture object to acquire webcam feed
        VideoCapture capture;
        //open capture object at location zero (default location for webcam)
        capture.open(0);
        //set height and width of capture frame
        capture.set(CV_CAP_PROP_FRAME_WIDTH,FRAME_WIDTH);
        capture.set(CV_CAP_PROP_FRAME_HEIGHT,FRAME_HEIGHT);
        //start an infinite loop where webcam feed is copied to cameraFeed matrix
        //all of our operations will be performed within this loop
        waitKey(1000);
        while(1){
                //store image to matrix
                capture.read(cameraFeed);
```

Figure 8 Other detection code

● Third Submission

1. Soldering

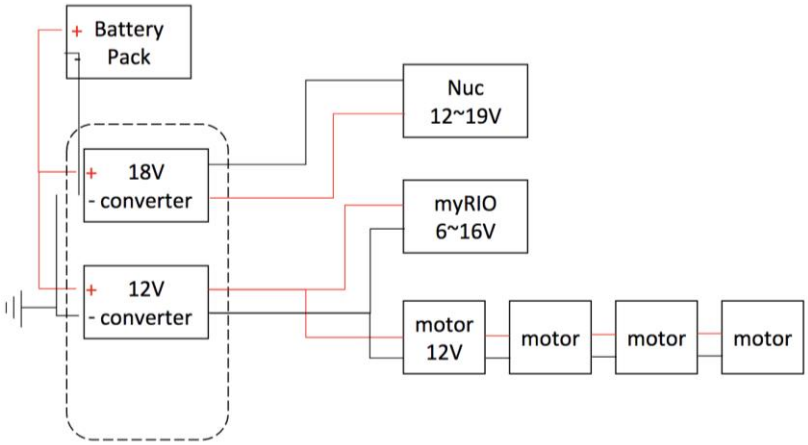


Figure 9 Converter Circuit

For the above circuit, we should have had the time to solder. But, we didn't buy the converter to connect between NUC and battery pack, so in our group, I and the other person soldered only one converter to use myRIO and SMPS like the under photos.

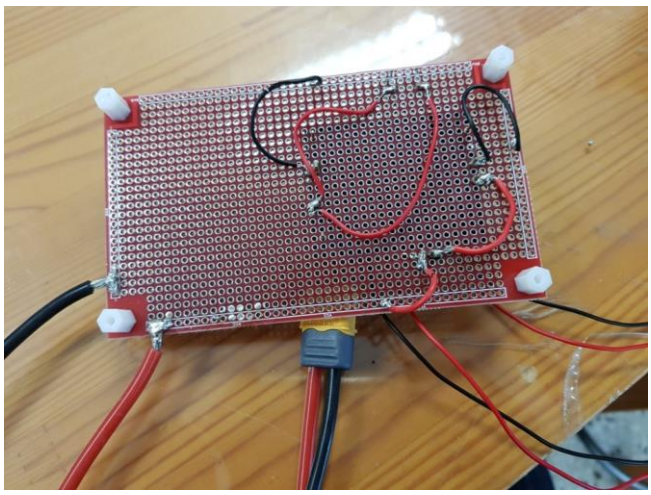


Figure 10 The back side of the panel

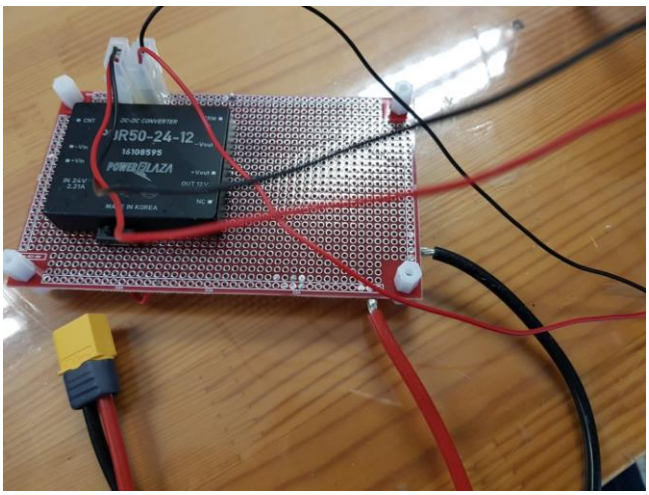


Figure 11 The front side of the panel

2. Opencv

To go to final stage in project, there are many things to do. First, I changed the input value in the ball detect code such as tracking ball size which is minimum

2pixels to percept red or blue. I found that the webcam can detect the ball 4m away from the webcam. But if in this case, the webcam can't detect the ball as the one. So, I should verify the code mixing the other various image processing methods.

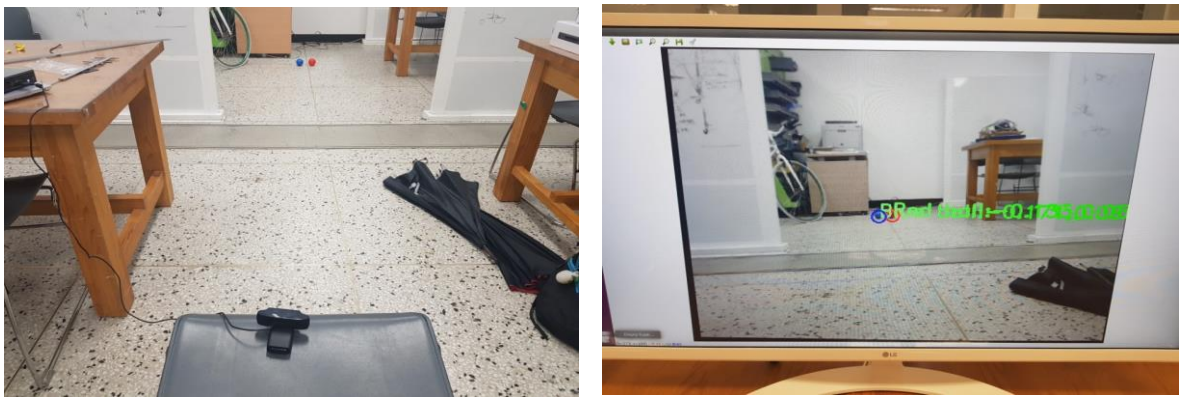


Figure 12 Ball detecting the ball 4m away from the webcam

Second, so I met the Opencv teaching assistant for asking for advice about the stream of code. I will verify the code to detect in various situations.

Third, we integrated the ROS and Opencv parts. We will add the position code in the ball detect code so that the robot will recognize the position of the balls and then communicate with LabView part.

Fourth, we did the RPlidar for mapping. We concluded that the slam method is not appropriate in mapping. Before using RPlidar, we discussed about path generation and made a rough planning. In the next Monday, we will make a final decision about path generation because the information about map will be noticed at that day.

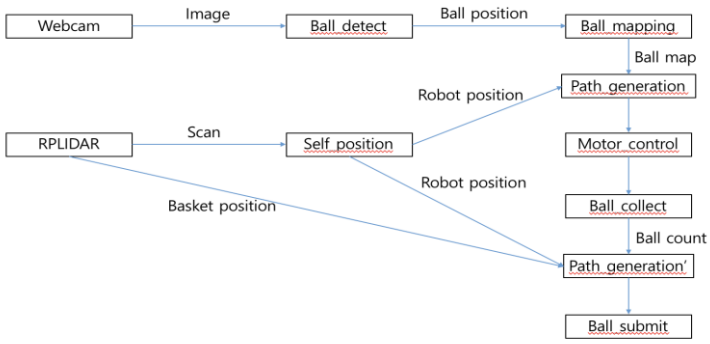


Figure 13 rqt graph

● Fourth Submission

1. Second Presentation

For the second presentation, we should prepare the video clips showing the whole operation. In that operator, there should be ball detection, motor driving through communication between ROS and LabView. Before we take a video, we check whether the connection line with converter is realizable and the planning for online order.

In the assembly, the position of the camera is very important. So, I derived the minimum height of the camera through webcam angle, the detection distance, and the demo size etc. (webcam angle = 78 degree, detection distance = about 4m)

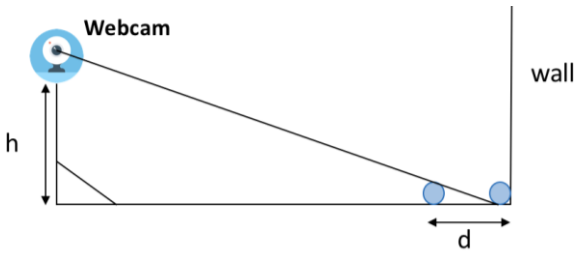
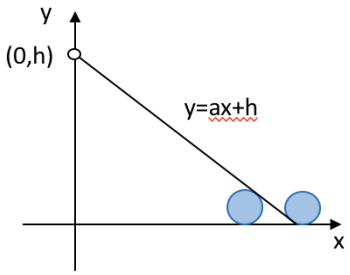


Figure 14 The side view of worst case

The above figure represents the worst case when our mobile platform is detecting. In this case, webcam can distinguish the balls if the tangent line of each ball is the same. So, we can derive using formula expressing the distance between the point and the center of circle. The procedure of proof is the same as below.



Circles' equations
: $(x-x_1)^2+(y-r)^2=r^2$,
$(x-x_2)^2+(y-r)^2=r^2$

Tangent line equation
: $y-ax-h=0$

$$\Rightarrow \quad \frac{|ax_1-r+h|}{\sqrt{a^2+1}}=\frac{|ax_2-r+h|}{\sqrt{a^2+1}}=r \quad \Rightarrow \quad h=\frac{r(x_1+x_2)}{\sqrt{(x_2-x_1)^2-4r^2}}+r$$

$$x_1=202cm, x_2=259cm, r=3.75cm \qquad \therefore h=34.35cm$$

Figure 15 Proof of the minimum webcam height

2. OpenCV code modification

For the upgrading detect code, I spent my time. First of all, I thought that it is important to not only detect well but also send the meaningful data to ROS. So, I modified the code to distinguish the circles in a ball because our demo ball's reflexibility is high and if the ball is low distance from webcam, then the webcam detects a ball into lots of circles. For this modification, I used for, if, and while statement, and the result window is below. As we see in the right window, webcam detects the 3 circles in a ball, but we can send to 1 circles data to ROS.

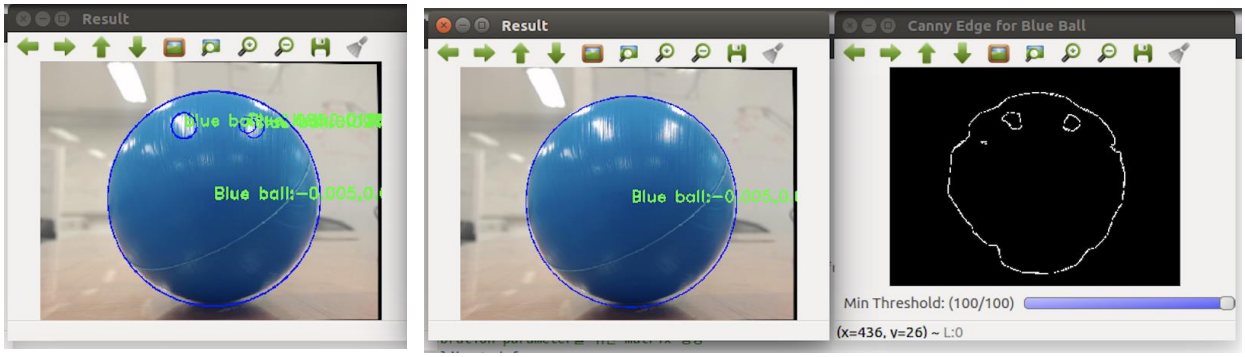


Figure 16 Detection Result Window before & after modifying

And then, I did experiment for the demo that check whether the webcam detects the three blue balls according to distance. About in 4.5m distance, detection is little bit uncomfortable, and from about 4m to close distance, it's comfortable. So, I changed the code for increasing the detecting distance for red balls, green balls as well.

Figure 17 Detection with change to distance

● Fifth Submission

1. Modification red ball detect code

Our robot should avoid red balls when it goes to the blue balls. However, the basic code of ball detecting has some problem when detecting red balls. It is strongly related to the condition of light and distance. So, although we only need to detect red balls about 1m far from our robot, it is very annoying to adjust the value of detect according to various demo condition in every time. So, I should modify the code only for detecting red balls more better, and I found the suitable method. That is transformation 'RGB' to 'YCbCr' color space, which differs from 'HSV' color space we used.

In HSV color space, H means Hue, S saturation, V value. V value represents the brightness of the image, and the problem is the point that our balls reflect some lights and the webcam doesn't detect the red ball in image with HSV image processing.

In contrast with HSV, Y in YCbCr is the value of luminance meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries. In easily, the luminance value is regarded as the intensity of the specular surface. So we make the webcam detect the lighting surface of red balls. The other values, Cb and Cr have the same role with Hue in HSV. In conclusion, the YCbCr color space doesn't have the saturation value, and has the one luminance value and two hue values.

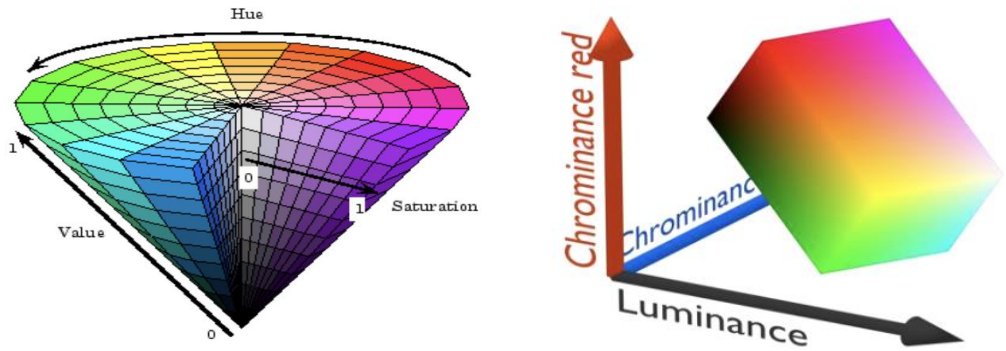


Figure 18 'HSV' & 'YCbCr' color space

As a result of demo, I can check that YCbCr color space is better than HSV in detecting red balls. After that, I found all values in Y, Cb, and Cr for final demo.

Figure 19 Light condition for demo

2. Demo Rehearsal

Because we don't have enough time to final demo, I should have found the problem in my part and modified code. First, our group found the appropriate light circumstances in real demo place. During the left time, we should check that in every situation the robot will detect all balls in image from webcam.

In vibration part, we don't have to control the vibration because that is much smaller than the amount of affecting the robot's vision. In control part, our robot should avoid the red ball exactly in wanted position that is derived from the data of ROS part. However, in the vision part the detecting balls has a little bit noise. The ball detection code is the optimal code for detecting balls and sending the datum to ROS, so I should use another way to eliminate noises.

The idea is that after storing the position datum of previous loop and comparing datum in present loop with the previous loop, we add weight to each datum. For this method, I will use the array grammar. The method is not completed yet, so I will modify code at least in 2 days for perfect demo rehearsal.