Image Processing with OpenCV

Practice 1 : Image Processing

1. X_Y.cpp

1-1. 코드설명

```
int main(int argc, char ** argv) {
    string command; // menu 선택
    gMatImage = imread("usecase diagram.png", 1); //0은 gray scale,1은 color scale
    gMatFlipImage = cv::Mat::zeros(gMatImage.size(), gMatImage.type());

    cv::namedWindow("Image");
    cv::imshow("Image", gMatImage);

    cv::namedWindow("FlipImage");

    while (1)
    {
        cin >> command;
        if (command == "x")
        {
            cv::setMouseCallback("Image", reflect_X, 0);
            cv::waitKey(0); //이미지를 띄울시간을 정해줌
        }
        else if (command == "y")
        {
            cv::setMouseCallback("Image", reflect_Y, 0);
            cv::waitKey(0); //이미지를 띄울시간을 정해줌
        }
    }

    return 0;
}
```

main 함수에서는 command line으로 x또는 y의 input이 들어올때까지 기다리고 input 값에 따라 mouse callback 함수가 호출된다.

```
else if (event == CV_EVENT_LBUTTONUP)
{
    drawing_box = false;
    //원본에서 사각형 바운더리 공간만 바꾸기
    gMatImage.copyTo(gMatFlipImage);
    for (int i = left_upper.y; i < right_down.y; i++)
    {
        for (int j = left_upper.x; j < right_down.x; j++)
        {
            gMatFlipImage.at<cv::Vec3b>(i, j) = gMatImage.at<cv::Vec3b>((right_down.y - 1) - i+left_upper.y, j);
        }
    }
    cv::imshow("FlipImage", gMatFlipImage);
    gMatFlipImage.copyTo(gMatImage);
}
```

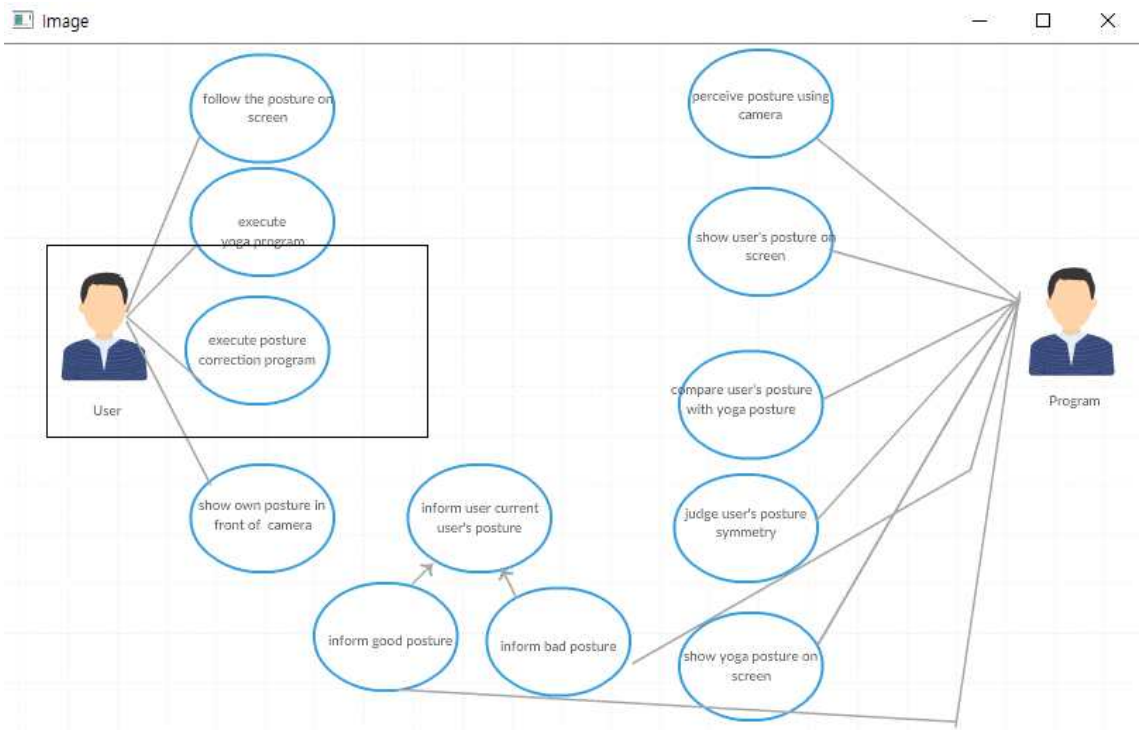
위의 코드는 reflect_X 함수에서 상하반전을 시키는 코드이다. 먼저 원본이미지를 결과이미지에 복사하고, 사용자가 임의로 지정한 사각형의 영역안에서 원본이미지를 참고하여 x축 대칭인 점들을 mapping 시킨다.

```
else if (event == CV_EVENT_LBUTTONUP)
{
    drawing_box = false;
    gMatImage.copyTo(gMatFlipImage);
    for (int i = left_upper.x; i < right_down.x; i++)
    {
        for (int j = left_upper.y; j < right_down.y; j++)
        {
            gMatFlipImage.at<cv::Vec3b>(j, i) = gMatImage.at<cv::Vec3b>(j, (right_down.x - 1) - i+left_upper.x);
        }
    }

    cv::imshow("FlipImage", gMatFlipImage);
    gMatFlipImage.copyTo(gMatImage);
}
```

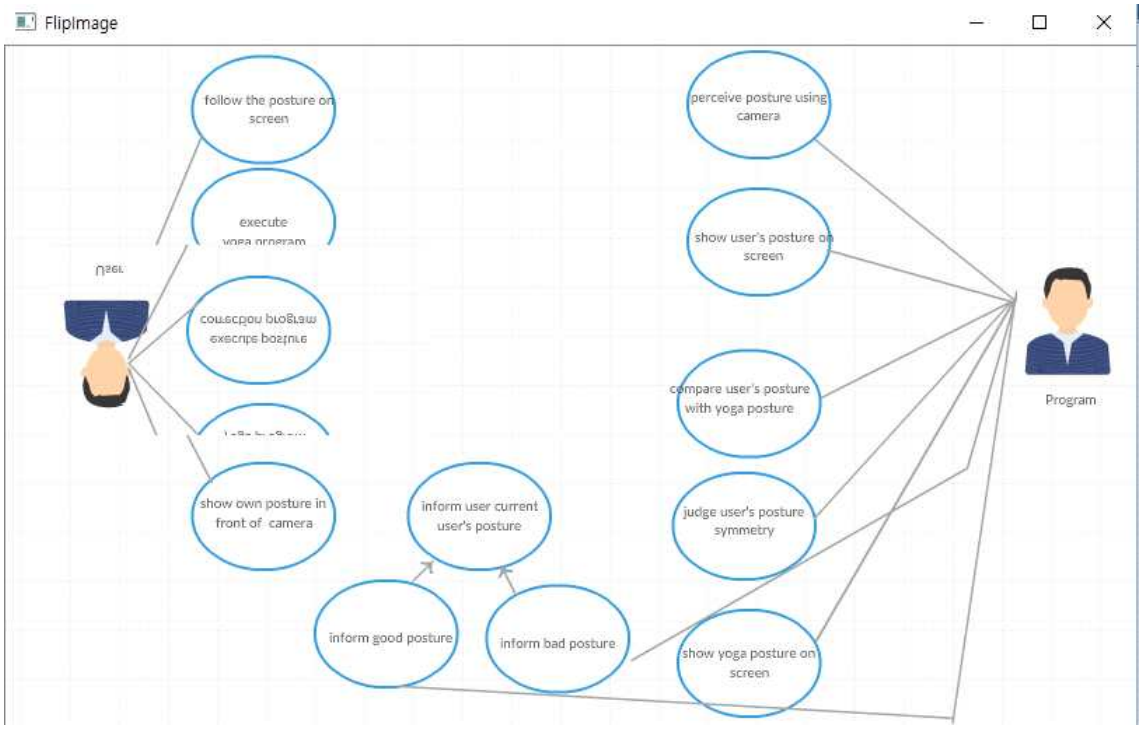
위의 코드는 reflect_Y 함수에서 좌우반전을 시키는 코드이다. 먼저 워본이미지를 결과이미지에 복사하고, 사용자가 임의로 지정한 사각형의 영역안에서 원본이미지를 참고하여 Y축 대칭인 점들을 mapping 시킨다.

1-2. 실행결과

<X축원본>



<X축 반전 결과>


FlipImage
follow the posture on screen
execute yoga program
execute posture correction program
show own posture in front of camera
perceive posture using camera
show user's posture on screen
compare user's posture with yoga posture
judge user's posture symmetry
show yoga posture on screen
inform user current user's posture
inform good posture
inform bad posture
Program

<Y축 원본>

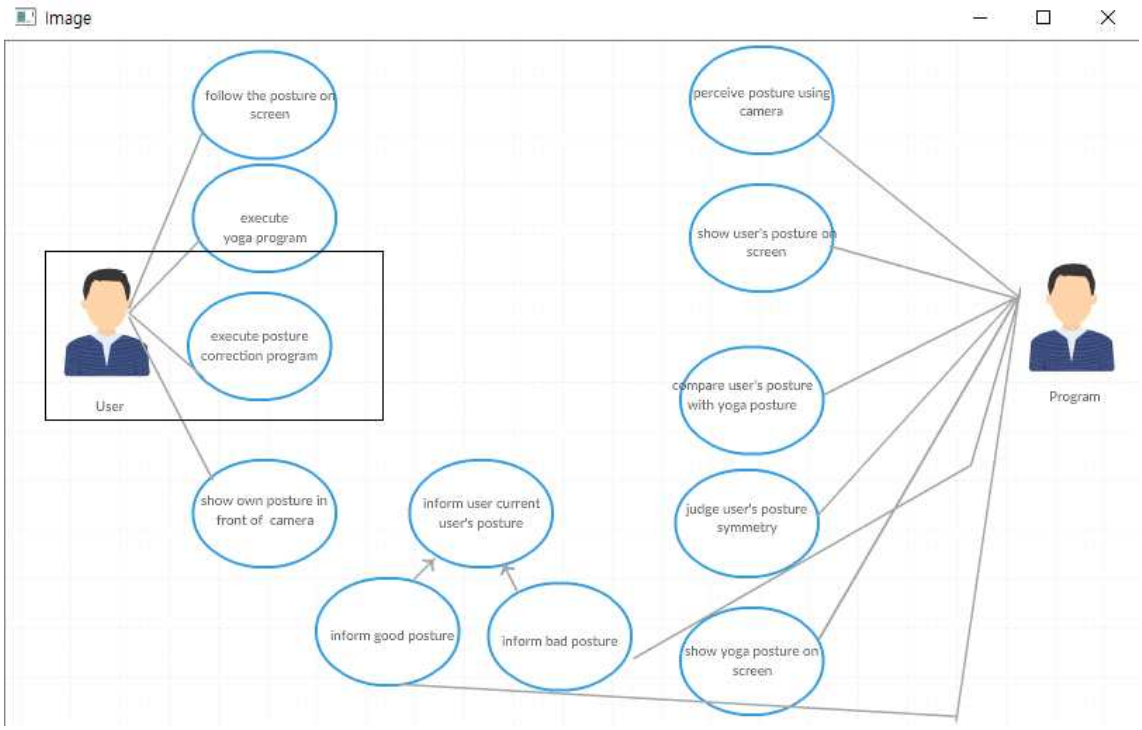
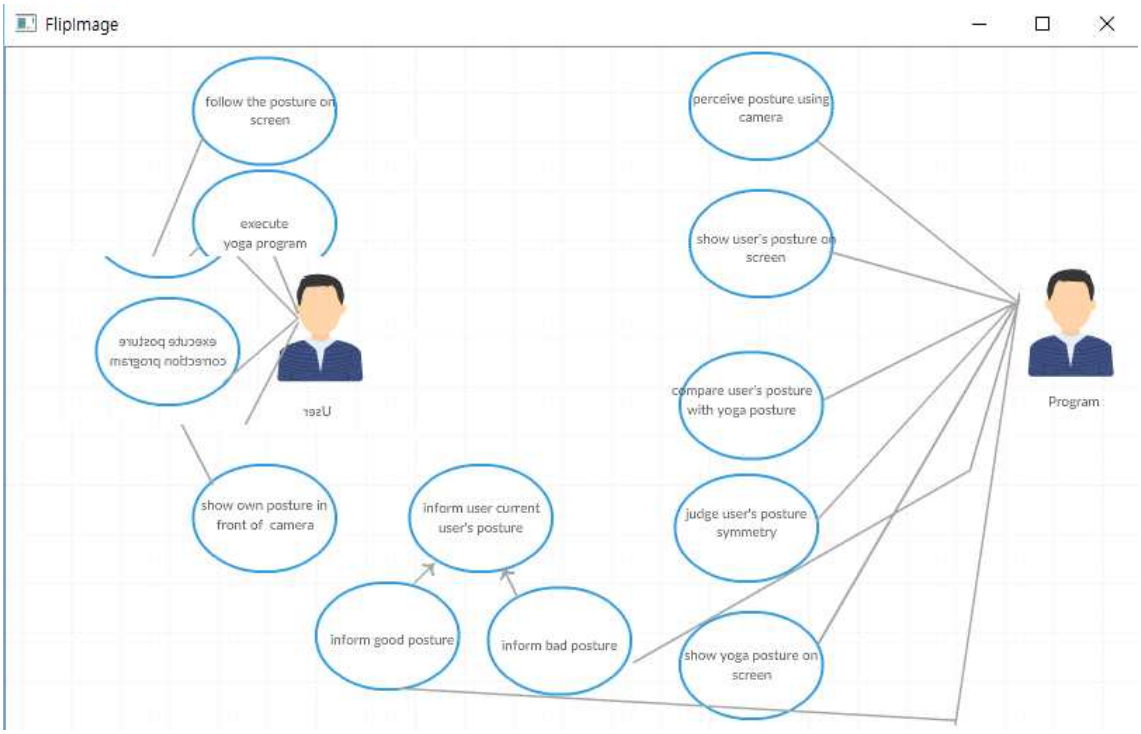


Image
follow the posture on screen
execute yoga program
execute posture correction program
User
show own posture in front of camera
perceive posture using camera
show user's posture on screen
compare user's posture with yoga posture
judge user's posture symmetry
show yoga posture on screen
inform user current user's posture
inform good posture
inform bad posture
Program

<Y축 반전 결과>



2. Scale.cpp

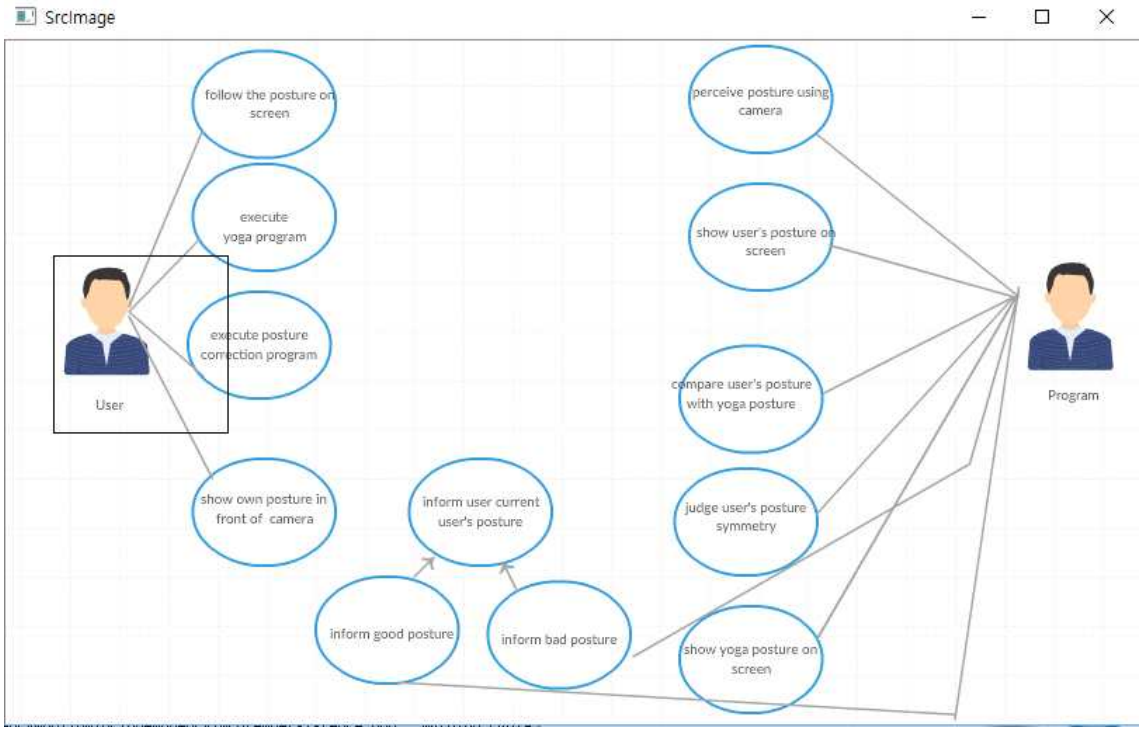
2-1. 코드설명

```
cv::Mat temp_img(src,cv::Range(left_upper.y, right_down.y), cv::Range(left_upper.x, right_down.x));//확대할 이미지

cv::resize(temp_img, scaled_img, cv::Size(temp_img.cols*1.5, temp_img.rows*1.5), 0, 0, cv::INTER_LINEAR);//1.5배 확대

center.x = (left_upper.x + right_down.x)/2;
center.y = (left_upper.y + right_down.y)/2;
new_lu.x = center.x - scaled_img.cols/2;
new_rd.x = center.x + scaled_img.cols / 2;
new_lu.y = center.y - scaled_img.rows / 2;
new_rd.y = center.y + scaled_img.rows/2;

cout << x << " " << y << endl;
cout << new_lu.x << " " << new_lu.y << endl;
cout << new_rd.x << " " << new_rd.y << endl;

cout << scaled_img.rows << " " << scaled_img.cols << endl;
src.copyTo(dst);
imshow("Scaled Image", dst);
for (int i = new_lu.x; i < new_rd.x; i++)
{
    for (int j = new_lu.y; j < new_rd.y; j++)
    {
        /*cout << j - new_lu.y << " " << i - new_lu.x << endl;*/
        dst.at<cv::Vec3b>(j, i) = scaled_img.at<cv::Vec3b>(j-new_lu.y, i-new_lu.x);
    }
}

imshow("Scaled Image" ,dst);
```

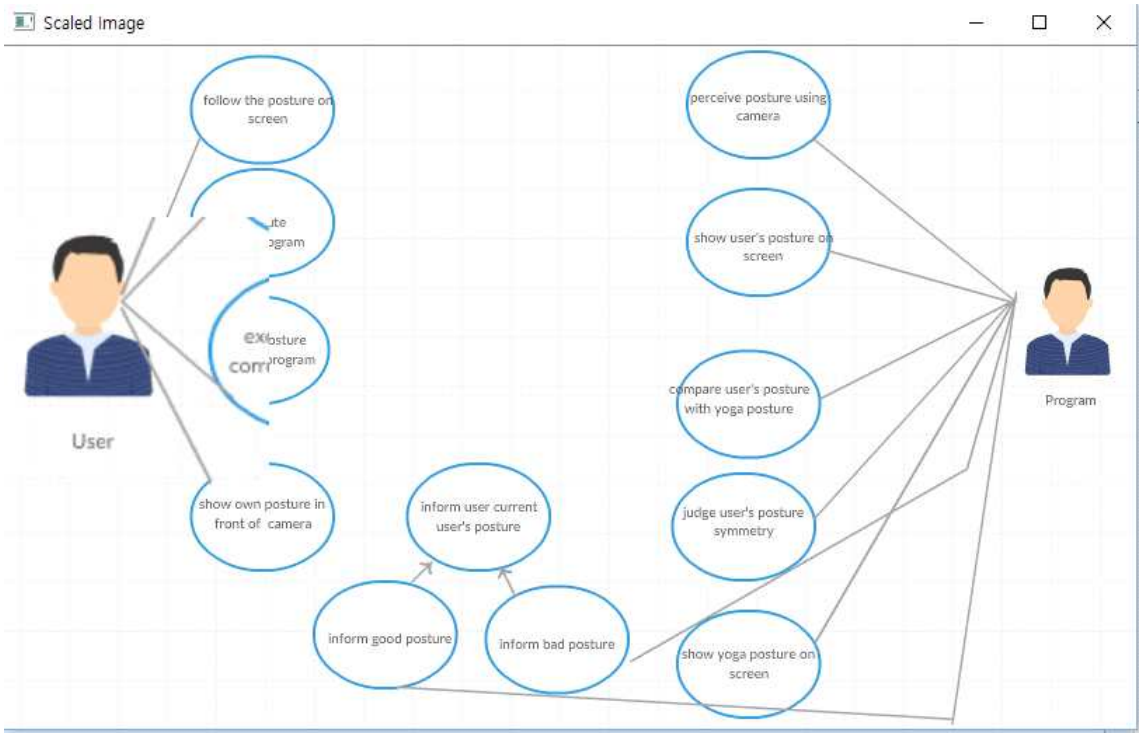
위 코드는 scale함수에서 1.5배 확대시키는 부분이다. 먼저 사용자가 지정한 사각형 영역을 temp_img에 저장한다. 그리고 resize함수를 호출하여 1.5배 확대하고 그 이미지를 원본이미지에 복사한다.

1-2. 실행결과

<1.5X 확대 원본>



<1.5X 확대 결과>



3. Rotate.cpp

3-1. 코드설명

```
cv::Mat temp_img(src, cv::Range(left_upper.y, right_down.y), cv::Range(left_upper.x, right_down.x));//회전할 이미지
temp_img.copyTo(flip_img);
for (int i = 0; i < temp_img.rows; i++)
{
    for (int j = 0; j < temp_img.cols; j++)
    {
        flip_img.at<cv::Vec3b>(i, j) = temp_img.at<cv::Vec3b>((temp_img.rows - 1) - i, j);
    }
}
Mat matRotation = cv::getRotationMatrix2D(Point(temp_img.cols/2, temp_img.rows/2),90, 1); //90도 만큼회전
cv::transpose(flip_img, rotated_img);
```

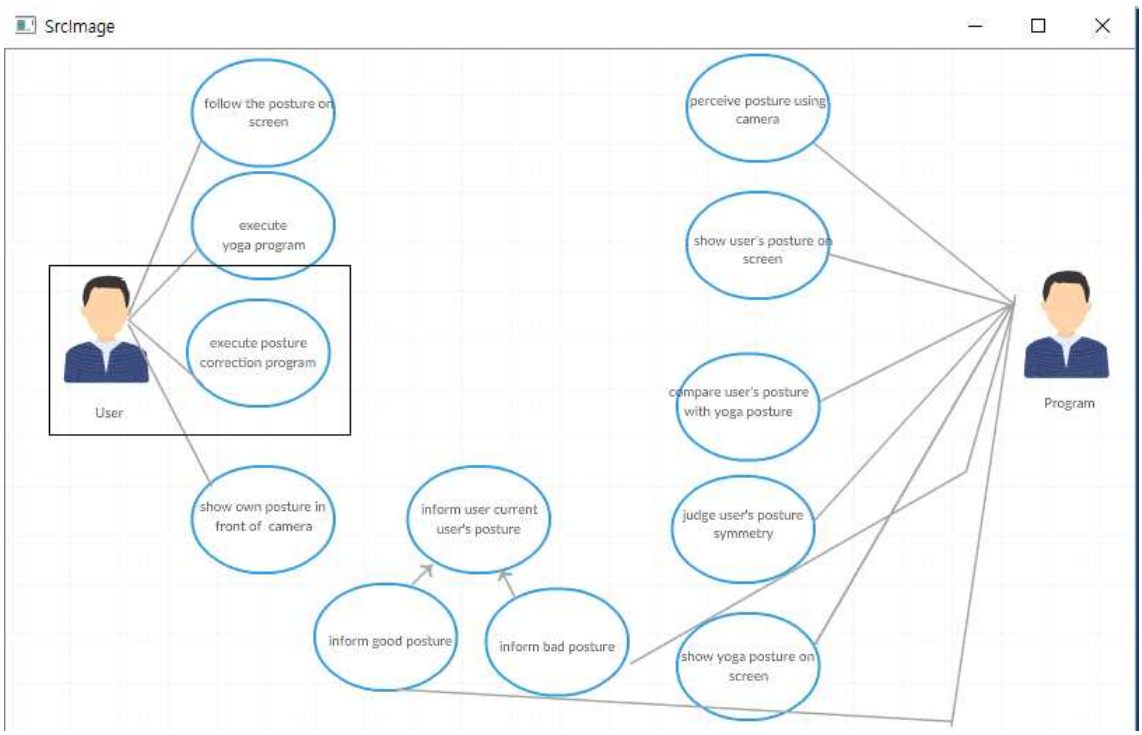
rotate함수에서는 temp_img에 사각형 이미지를 넣고 for문을 돌면서 먼저 좌우반전을 시킨다.

```
cv::transpose(flip_img, rotated_img);

imshow("Rotated Image", flip_img);

center.x = (left_upper.x + right_down.x) / 2;
center.y = (left_upper.y + right_down.y) / 2;
new_lu.x = center.x - rotated_img.cols / 2;
new_rd.x = center.x + rotated_img.cols / 2;
new_lu.y = center.y - rotated_img.rows / 2;
new_rd.y = center.y + rotated_img.rows / 2;

src.copyTo(dst);
for (int i = new_lu.x; i < new_rd.x; i++)
{
    for (int j = new_lu.y; j < new_rd.y; j++)
    {
        //cout << j - new_lu.y << " " << i - new_lu.x << endl;
        dst.at<cv::Vec3b>(j, i) = rotated_img.at<cv::Vec3b>(j - new_lu.y, i - new_lu.x);
    }
}
```

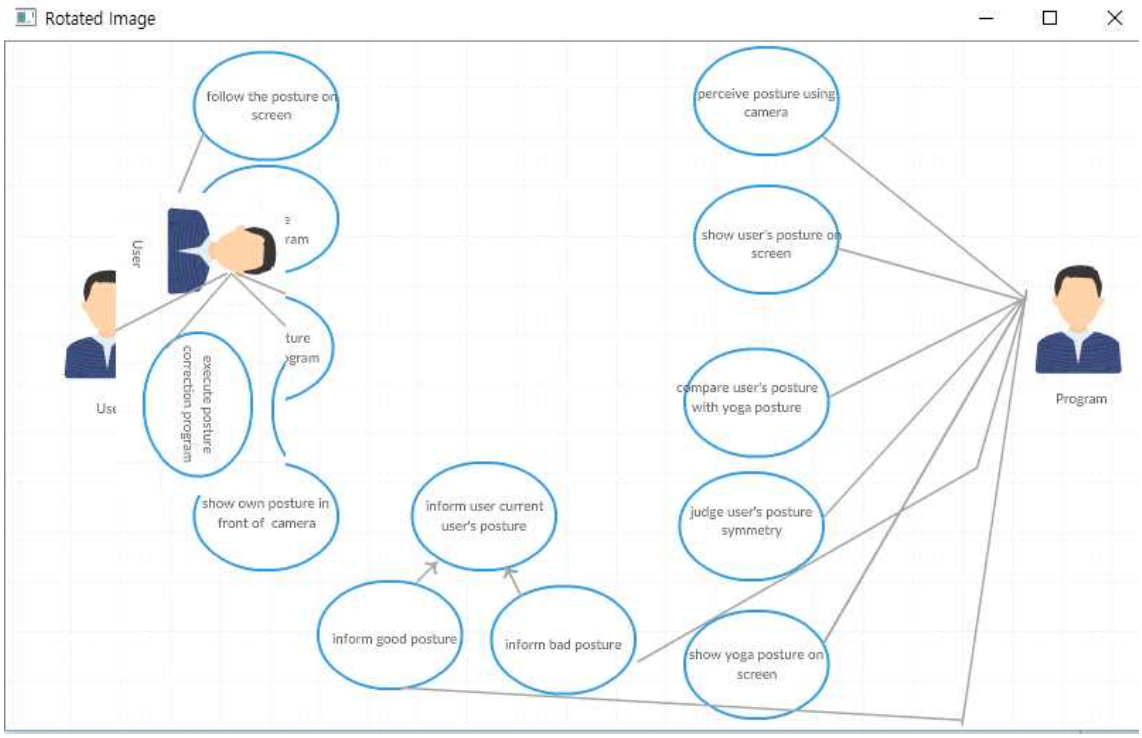
좌우반전시킨 이미지를 transpose 하면 -90도 만큼 회전한 이미지가 rotated_img에 저장된다. 이제 마지막으로 rotated_img를 원본이미지의 복사본인 dst이미지에 복사한다.

3-2. 실행결과

<90도 회전 원본>



<90도 회전 결과>



Practice 2 : Global Thresholding

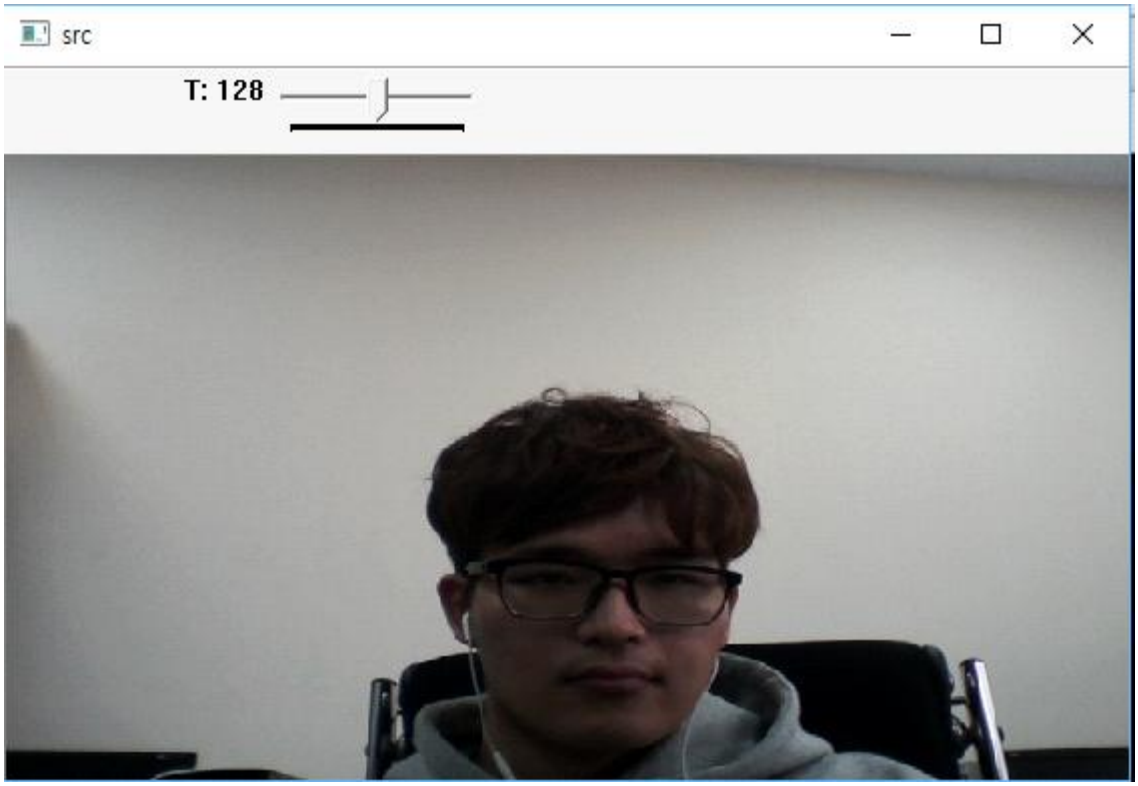
2-1. 코드설명

```
for (;;)
{
    cv::Mat frame;
    cap >> frame; // get a new frame from camera
    cv::cvtColor(frame, g_YCrCb, CV_RGB2YCrCb);
    cv::split(g_YCrCb, channels);

    imshow("src", frame );
    //imshow("RGB", g_RGB);

    /*threshold(channels[1], channels[1], 77, 150, 3);
    threshold(channels[2], channels[2], 133, 173, 3);*/
    imshow("Y", channels[0]);
    cv::inRange(channels[1], cv::Scalar(77), cv::Scalar(150), channels[1]);
    cv::inRange(channels[2], cv::Scalar(133), cv::Scalar(173), channels[2]);
    imshow("Cr", channels[1]);
    imshow("Cb", channels[2]);
    cv::bitwise_and(channels[1], channels[2], g_Gray);
    imshow("Gray", g_Gray);
    cv::cvtColor(g_Gray, g_Bgr, CV_GRAY2BGR); //COLOR_GRAY2BGRA로 하면 안됨
    cv::bitwise_and(g_Bgr, frame, result);
    imshow("result", result);
    if (waitKey(30) >= 0) break;
}

return 0;
```

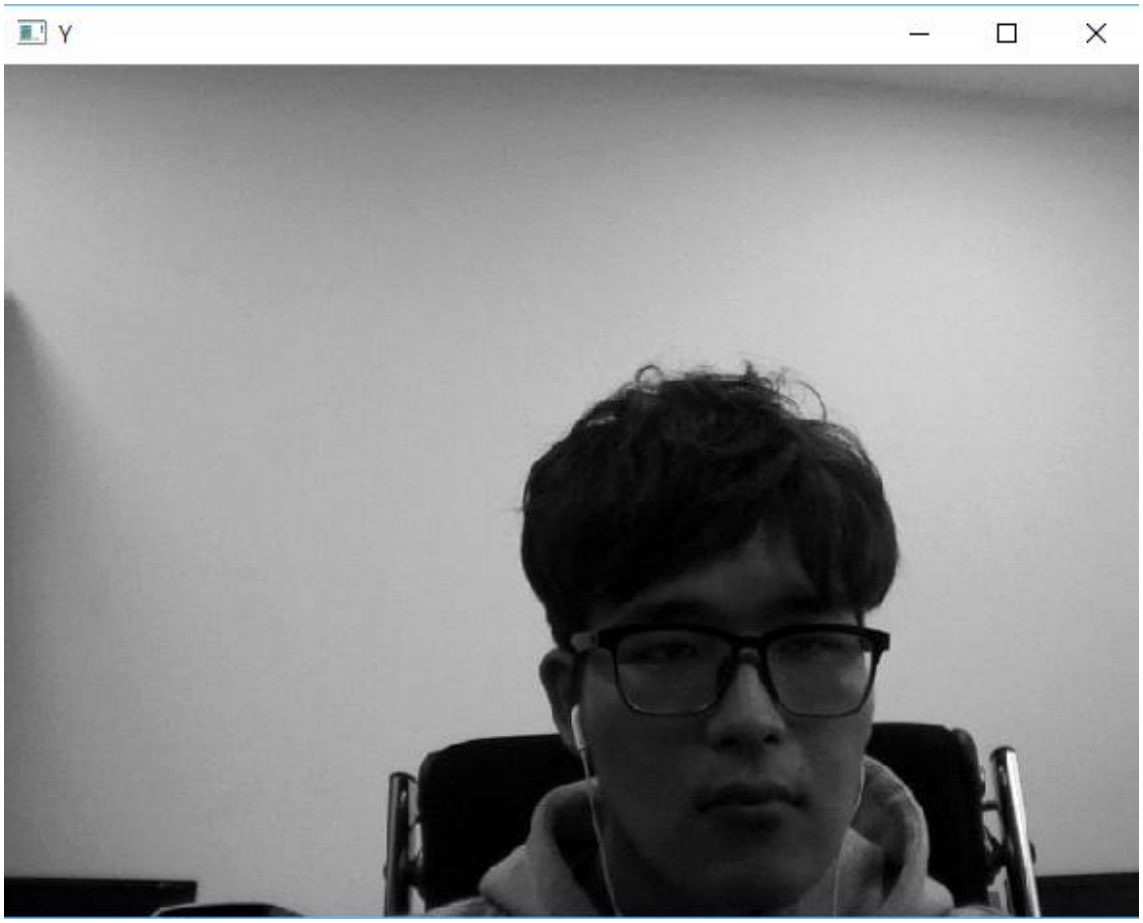
위와같이 비디오를 RGB->YCrCb로 변환하고 Cr과 Cb channel에 대하여 threshold를 한뒤 서로 bitwise를 하였다. 그리고 그 결과를 다시 BGR로 변환하고 원본 이미지와 bitwise하여 살색을 분리해내었다.

2-2. 실행화면
다음은 위 코드를 실행한 결과이다.

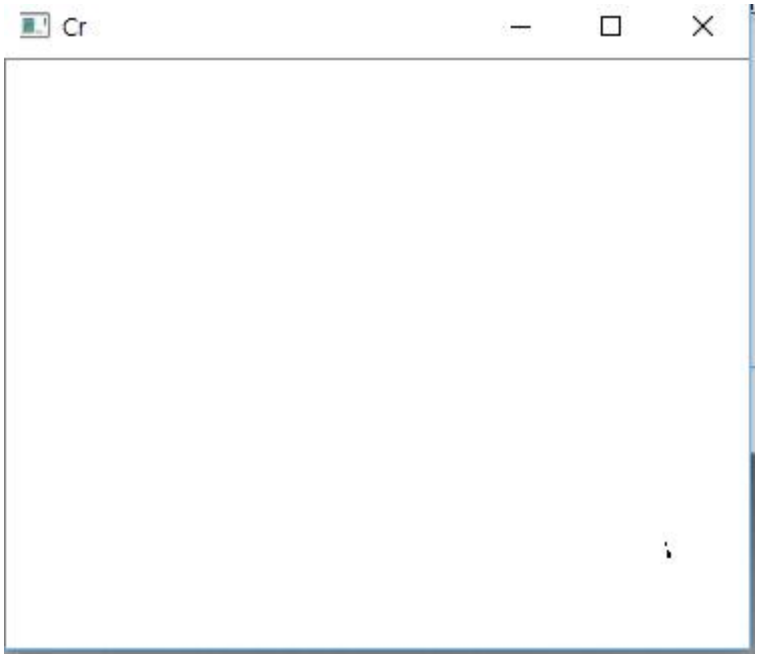
<원본영상 캡처>



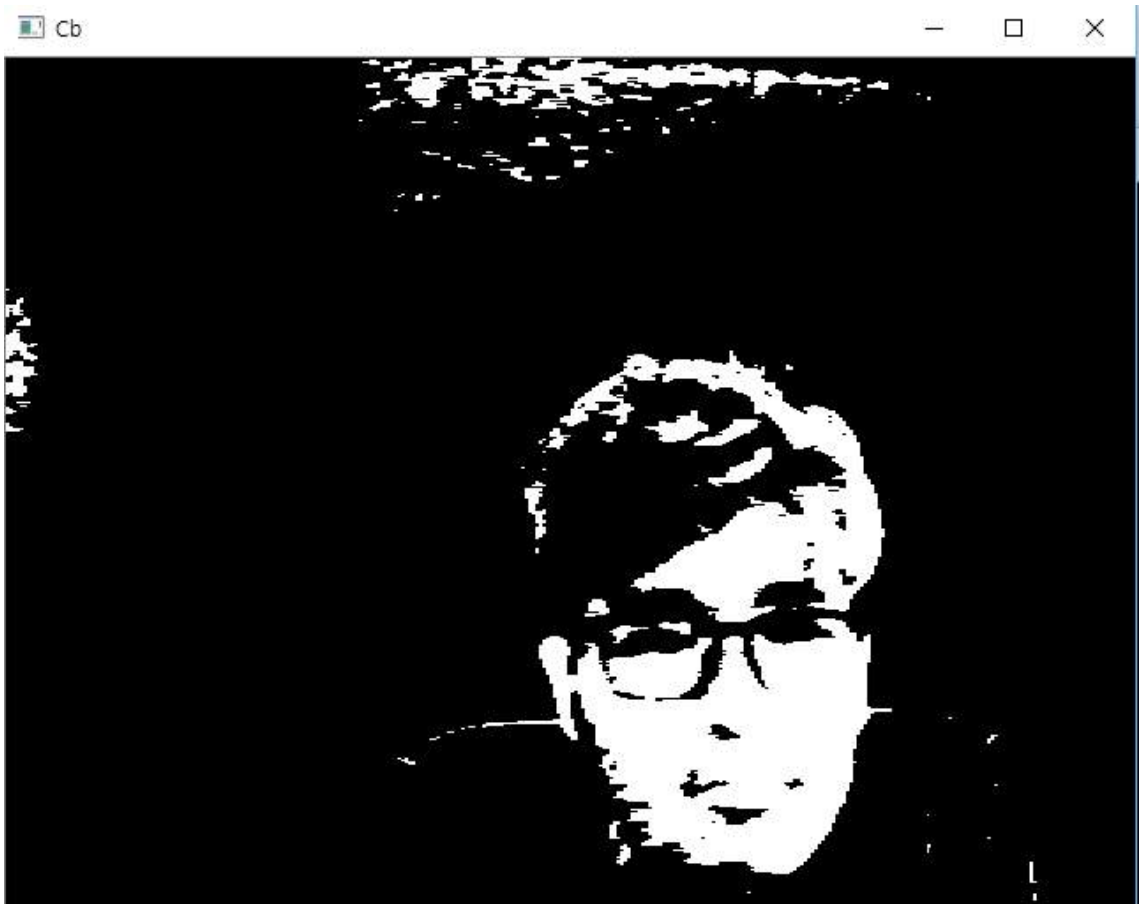
<Y>



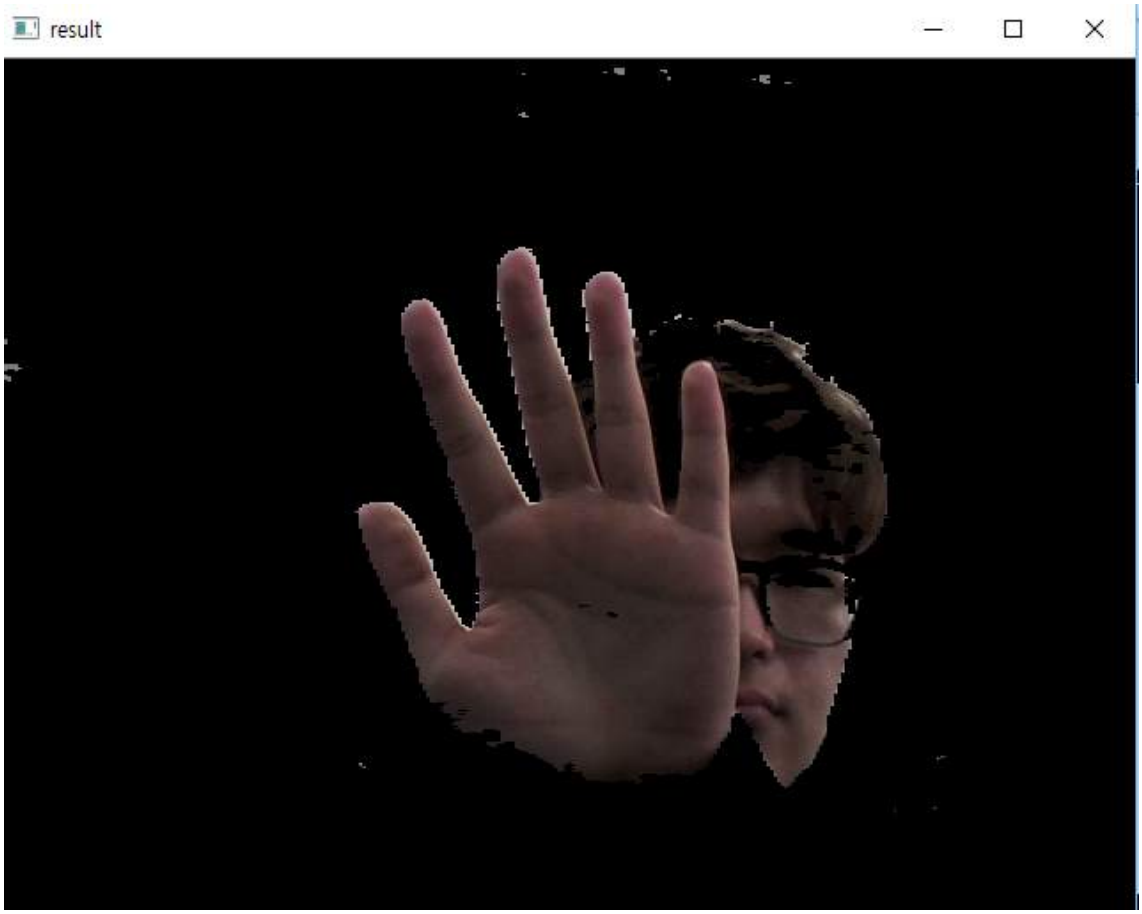
<Cr>



<Cb>



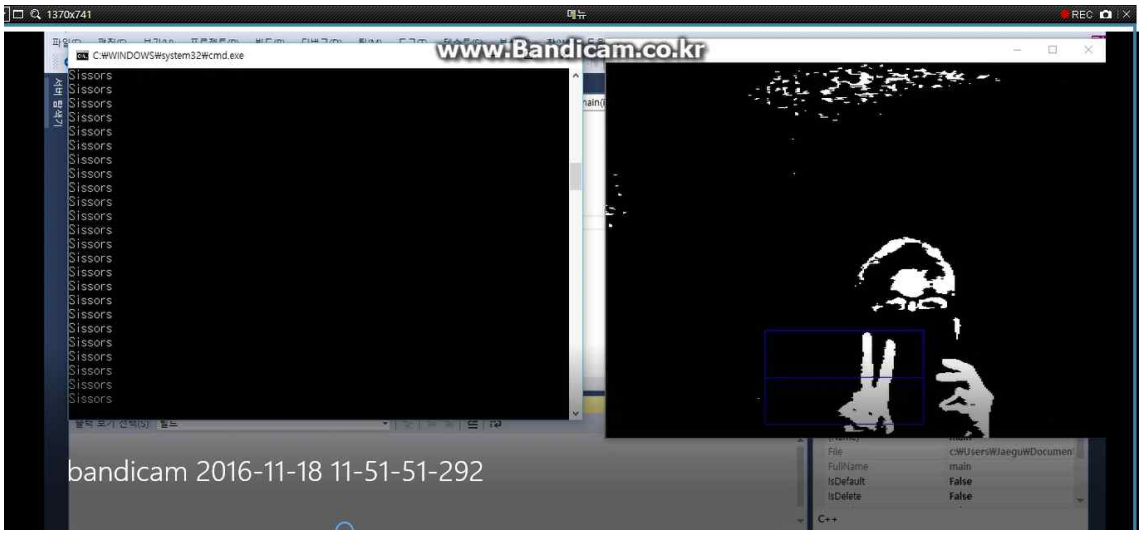
<Global Thresholding 결과화면>



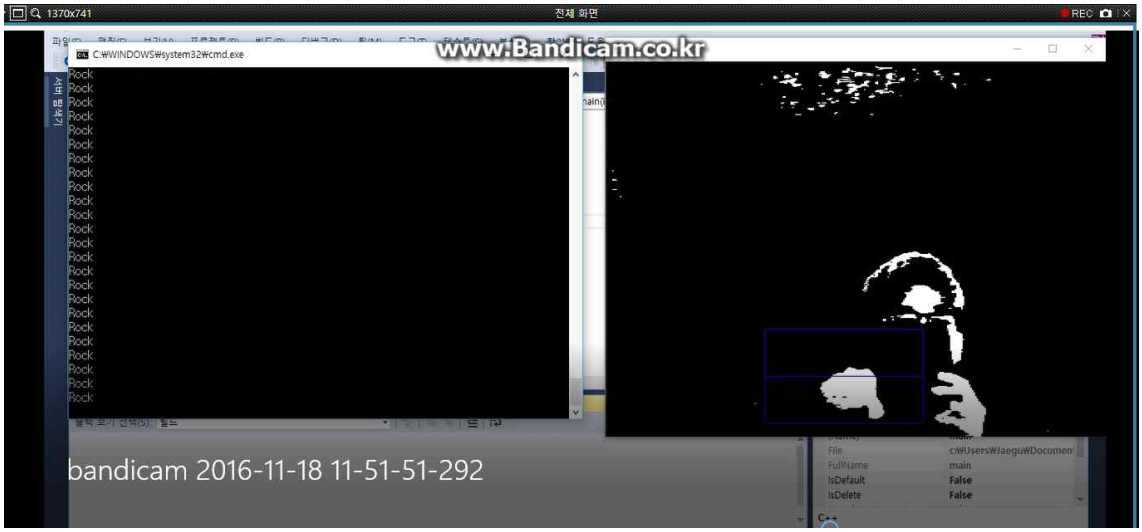
Practice 3 - Hand Tracking

시연영상 : https://youtu.be/cNubU2EixdE

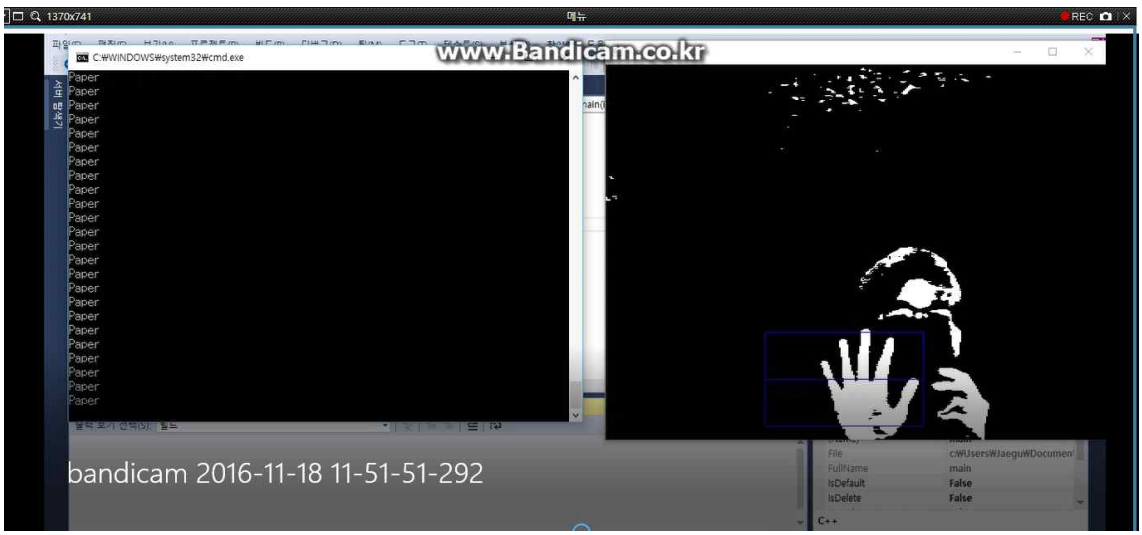
<가위>



<바위>



<보>



* 가위 바위 보 인식 방법

main idea : **Width 방향을 검사하여 손가락의 개수를 검사**

```
cv::rectangle(temp, cv::Point(203, 342), cv::Point(406, 462), cv::Scalar(255, 0, 0));
cv::line(temp, cv::Point(203,(342+462)/2), cv::Point(406,(342 + 462) / 2), cv::Scalar(255, 0, 0));

imshow("result",temp);
//imshow("result", g_Bgr);

int line_y = (342 + 462) / 2;

for (int x = 203; x<406; x++)
{
    // get pixel
    Vec3b color = g_Bgr.at<Vec3b>(Point(x, line_y));
    Vec3b color2 = g_Bgr.at<Vec3b>(Point(x + 1, line_y));
    if (color.val[0] == color2.val[0] && color.val[1] == color2.val[1] && color.val[2] == color2.val[2]) //색깔변화가 없을경우
    {

    }
    else //색깔변화 있을경우
    {
        change_cnt++;
    }
}

//cout << "change_cnt : " << change_cnt << endl;
if (change_cnt == 2)
    cout << "Rock" << endl;
else if (change_cnt == 4)
    cout << "Sissors" << endl;
else if (change_cnt == 10)
    cout << "Paper" << endl;
else
    cout << "Unknown" << endl;
```

1) 위의 코드에서처럼 화면의 하단부에 직사각형을 그리고 직사각형을 가로로 2등분하는 선을 긋는다.
2) 원본이미지(직사각형을 그리기 전 이미지)에서 2등분하는 선에 해당하는 픽셀을 하나씩 보면서 색깔 변화를 감지한다. 색깔변화가 있을 경우에 change_cnt값을 1씩 증가시킨다.
3) '가위'의 경우 색깔변화가 총 4번 나타나므로 change_cnt의 값은 4가된다.
'바위'의 경우 색깔변화가 총 2번 나타나므로 change_cnt의 값은 2가된다.
'보'의 경우 색깔변화가 총 10번 나타나므로 change_cnt의 값은 10이 된다.
4) 위의 change_cnt 값에 따라서 가위, 바위, 보를 구분한다.