MIN-VIS-2016

ASSIGNMENT 5 – GROUP4

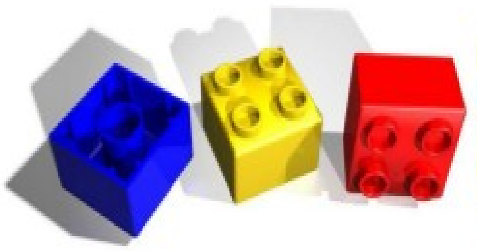
Group Members:

- Sandio Fongang R
- Armaan Rustami

Assignment 5 MINES-VIS

This Assignment will be graded. So hand in your work on time.

You must hand in a well-documented source code. (Clearly commented explanation of the functions between the C++ code)
What you need for this assignment: videostreams with the colored objects. See Canvas.



Task 1 Fast Object Tracking
Pick your preferred color object. Develop a Fast Object Tracking application using the OpenCV. Give a demonstration of your application which is tracking the colored object that you have chosen. The centroid of your selected object should be marked with a X and the coordinates should be displayed also on screen. You might use "SingleColors.mp4"

Task 2:

Choose a non-circle colored object, see "MeasuringAngle.mp4" and determine the orientation θ of that particular object.
Read paragraph 3.3.2 Orientation of the document "Simple Image Analysis by Moments.pdf" for more information. Give a demo of your app.

Task 3 Tracking and classifying multiple colored objects.

Modify your application in such a way that you can now track each objects (the collection of the colored object chosen by you.) and classify them. Each object must be marked with a box or circle. Use "MultipleColors.mp4"

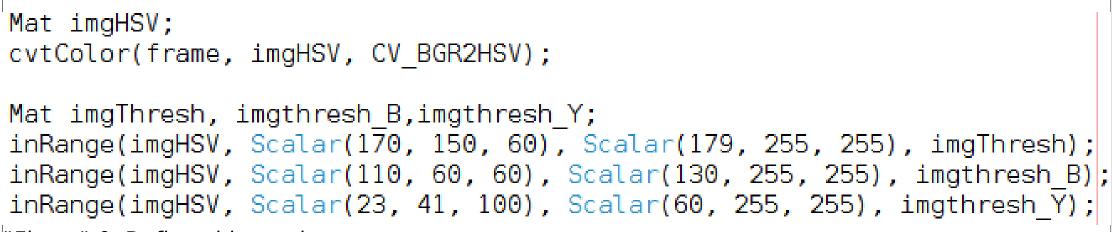
Discuss your results!! (for extra credits)
A demo presentation of your applications is mandatory!
Assignment must be handed in before Tuesday October the xxth 2016 with a Demo presentation!

2. Solution

a. Fast Object Tracking:

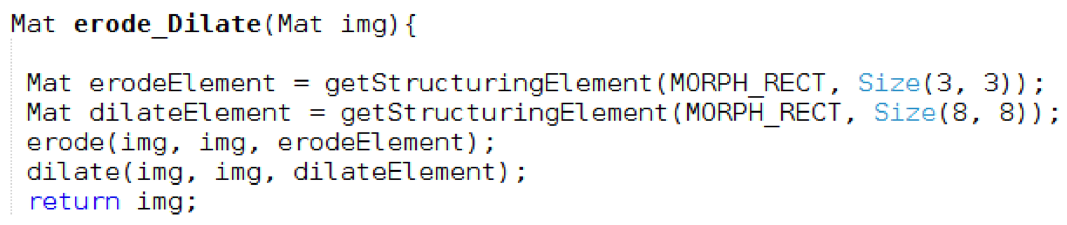
In order to track coloured Object and mark the Centre of Selected Object by marking With “X” , here are some Steps we took:

- Convert camera frame colour from BGR to HSV to get the actual colour . *Figure1*
- Detect an object based on the range of pixel value using OpenCV inRange() .*Figure1*
- Dilates white space(making it larger) and Erode int White space (making it smaller or non-existent) using OpenCV functions.*Figure2*



```
Mat imgHSV;
cvtColor(frame, imgHSV, CV_BGR2HSV);

Mat imgThresh, imgthresh_B,imgthresh_Y;
inRange(imgHSV, Scalar(170, 150, 60), Scalar(179, 255, 255), imgThresh);
inRange(imgHSV, Scalar(110, 60, 60), Scalar(130, 255, 255), imgthresh_B);
inRange(imgHSV, Scalar(23, 41, 100), Scalar(60, 255, 255), imgthresh_Y);
```

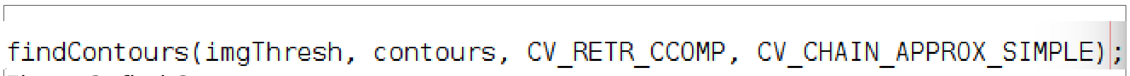
"Figure" 0: Define object color range



```
Mat erode_Dilate(Mat img){

 Mat erodeElement = getStructuringElement(MORPH_RECT, Size(3, 3));
 Mat dilateElement = getStructuringElement(MORPH_RECT, Size(8, 8));
 erode(img, img, erodeElement);
 dilate(img, img, dilateElement);
 return img;
```

Figure 2: Dilate And Erode

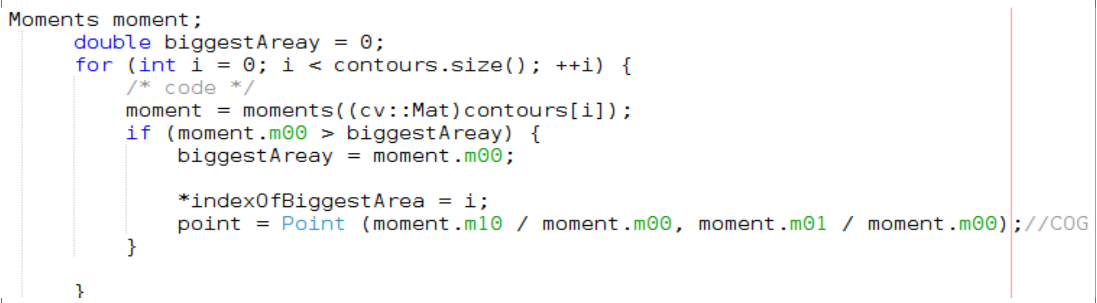
- Afterward, we apply findContour function on the output of erode and dilate ,which has the object of inrange (between specific pixel value range).*Figure3*



```
findContours(imgThresh, contours, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
```

Figure 3: find Contours

- After doing the mentioned above steps , we find the Find the biggest area of object using Moments *(Figure 4)* to Circle the object and its Centre of Gravity (COG) to centroid the object by marking with “X” label. *(Figure 5)*



```
Moments moment;
    double biggestAreay = 0;
    for (int i = 0; i < contours.size(); ++i) {
        /* code */
        moment = moments((cv::Mat)contours[i]);
        if (moment.m00 > biggestAreay) {
            biggestAreay = moment.m00;

            *indexOfBiggestArea = i;
            point = Point (moment.m10 / moment.m00, moment.m01 / moment.m00);//COG
        }

    }
```

Figure 4: find the biggest Area

```
putText(drawing,"X",COG,1,1.2,Scalar(0,255,255),2,8,false);
```

Figure 5: Label centre of Object

- Then we apply circling the object but before that we find the enclosing circle to the contour and then circle based in radius of contour .Figure 6

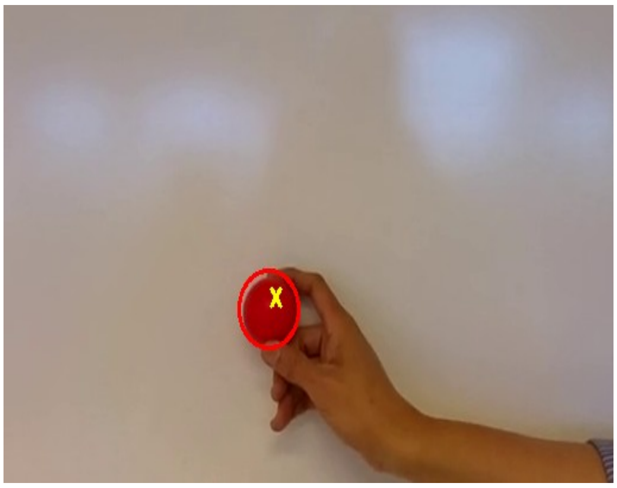
```
void Circle_Color(int indexOfBiggestArea ,vector<vector<Point> > contours,vector<float>radius,vector<Point2f>center,
        vector<vector<Point> > contours_poly, Scalar color){


            approxPolyDP(Mat(contours[indexOfBiggestArea]), contours_poly[indexOfBiggestArea], 3, true);
        //  boundRect[indexOfBiggestArea] = boundingRect(Mat(contours_poly[indexOfBiggestArea]));
          minEnclosingCircle((Mat) contours_poly[indexOfBiggestArea], center[indexOfBiggestArea], radius[indexOfBiggestArea]);
        //  drawContours(drawing, contours, indexOfBiggestArea, Scalar(0, 0, 255), 2, 8, hierarchy, 0, Point());
            circle(drawing, center[indexOfBiggestArea], (int) radius[indexOfBiggestArea], color, 2, 8, 0);
            //rectangle(drawing , boundRect[indexOfBiggestArea].tl(), boundRect[indexOfBiggestArea].br(), Scalar(0,0,255), 2, 8, 0 );
               // pointTheCenter(COG);
               getOrientation( contours[indexOfBiggestArea]);
}
```

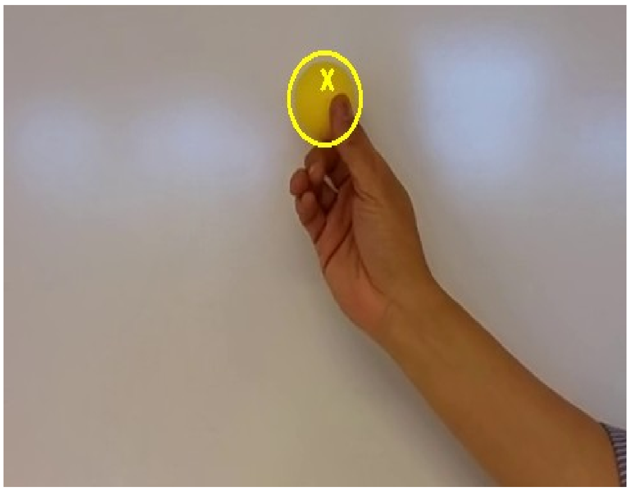
Figure 6: Circle the Contour

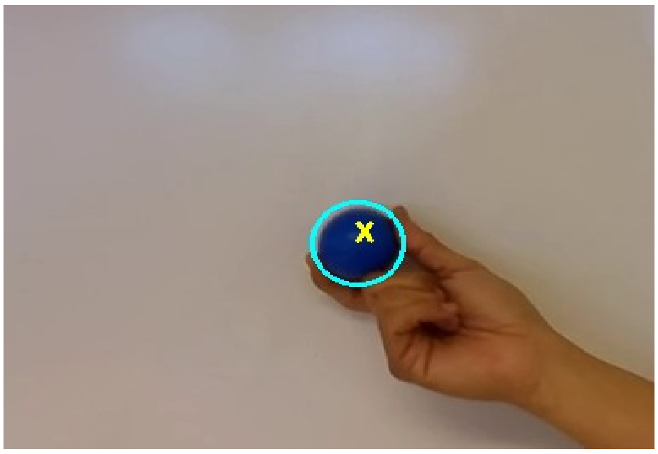
```
imshow("Output", drawing);
if (waitKey(30) >= 0) break;
```

Figure : Show the output

Output:



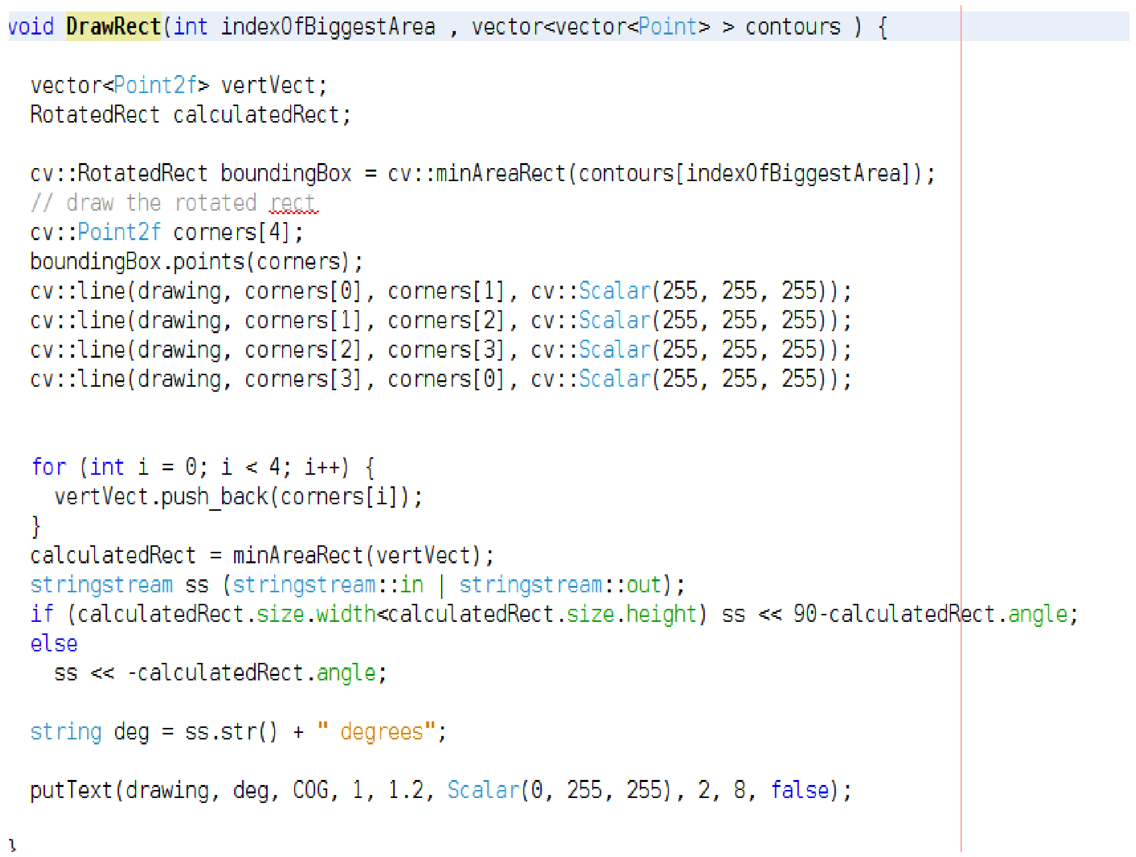




b. Measuring Angle:

The goal of this assignment is to determine the orientation of object ,in this Task we OpenCV builtin “RotatedRect” function , which its rotating with the object Rotation.

We the use mostly the same methods as Task1 ,and since its asked to determine the orientation of the object ,in that case we use the 'RotatedRect' angle property which gives us angle of the object.



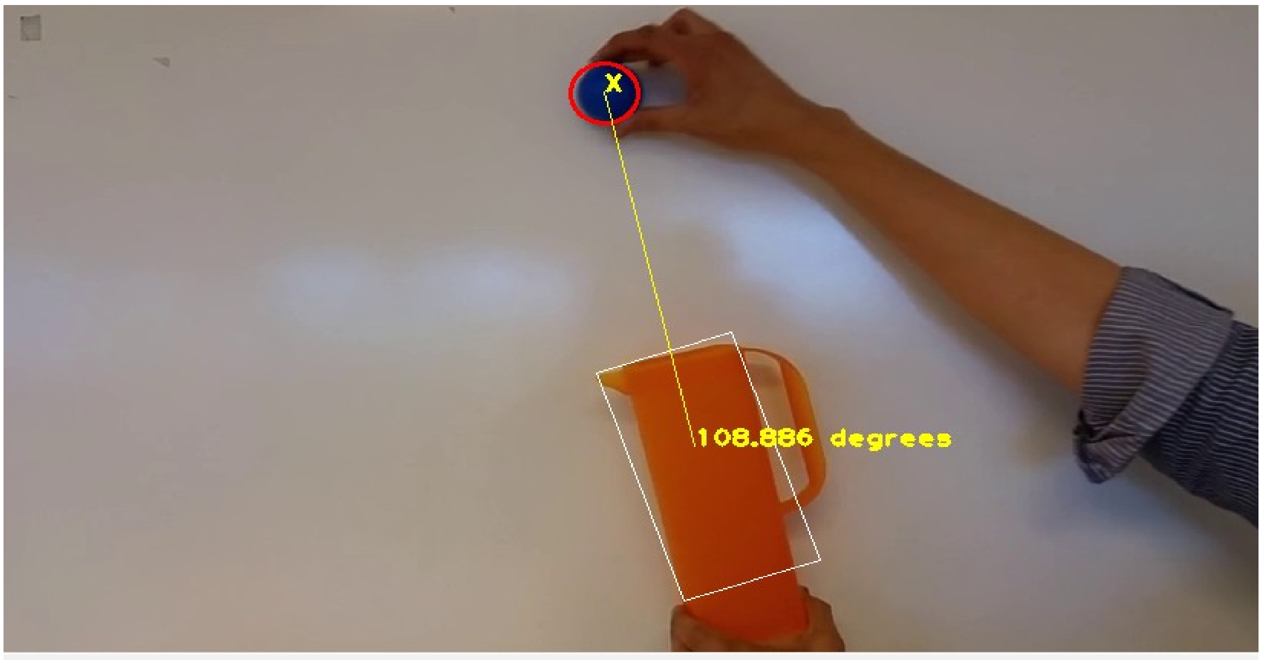
```
void DrawRect(int indexOfBiggestArea , vector<vector<Point> > contours ) {

  vector<Point2f> vertVect;
  RotatedRect calculatedRect;

  cv::RotatedRect boundingBox = cv::minAreaRect(contours[indexOfBiggestArea]);
  // draw the rotated rect
  cv::Point2f corners[4];
  boundingBox.points(corners);
  cv::line(drawing, corners[0], corners[1], cv::Scalar(255, 255, 255));
  cv::line(drawing, corners[1], corners[2], cv::Scalar(255, 255, 255));
  cv::line(drawing, corners[2], corners[3], cv::Scalar(255, 255, 255));
  cv::line(drawing, corners[3], corners[0], cv::Scalar(255, 255, 255));


  for (int i = 0; i < 4; i++) {
    vertVect.push_back(corners[i]);
  }
  calculatedRect = minAreaRect(vertVect);
  stringstream ss (stringstream::in | stringstream::out);
  if (calculatedRect.size.width<calculatedRect.size.height) ss << 90-calculatedRect.angle;
  else
    ss << -calculatedRect.angle;

  string deg = ss.str() + " degrees";

  putText(drawing, deg, COG, 1, 1.2, Scalar(0, 255, 255), 2, 8, false);

}
```

"Figure" 8: Drawing Rotated Rectangle and Degree

Output:

146.31 degrees

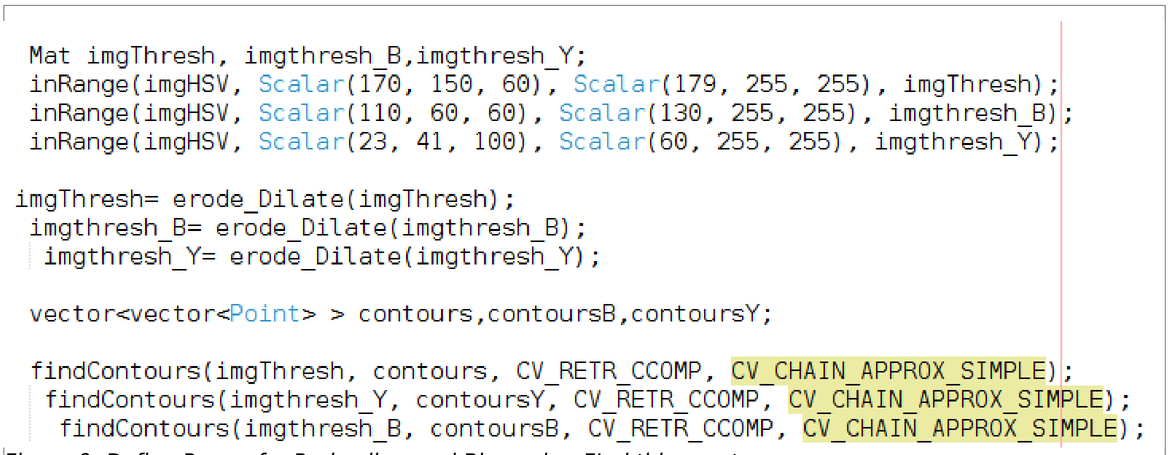
180 degrees


X
108.886 degrees

c. Tracking and classifying multiple coloured objects:

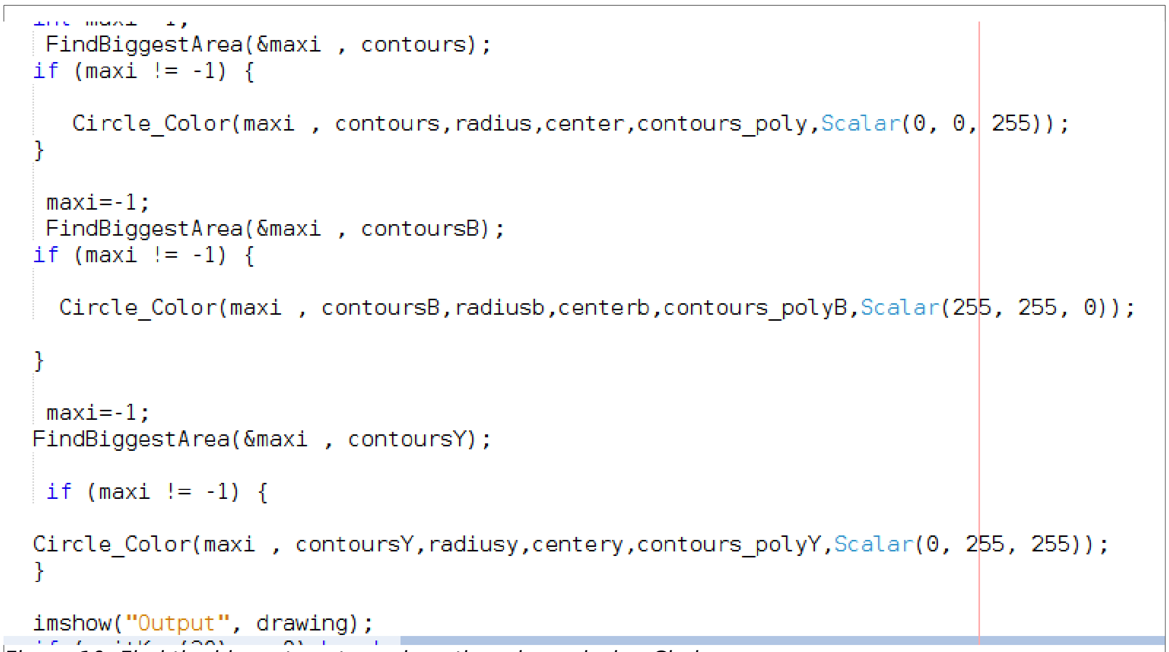
this task follows “Task a (**Fast Object Tracking**)” ,in which we define the ranges for multiple colours and find its contour .

The same as “Task a” we the find the biggest contour area and draw the minimum enclosing circle to the contour .



```
 Mat imgThresh, imgthresh_B,imgthresh_Y;
 inRange(imgHSV, Scalar(170, 150, 60), Scalar(179, 255, 255), imgThresh);
 inRange(imgHSV, Scalar(110, 60, 60), Scalar(130, 255, 255), imgthresh_B);
 inRange(imgHSV, Scalar(23, 41, 100), Scalar(60, 255, 255), imgthresh_Y);

imgThresh= erode_Dilate(imgThresh);
 imgthresh_B= erode_Dilate(imgthresh_B);
  imgthresh_Y= erode_Dilate(imgthresh_Y);

 vector<vector<Point> > contours,contoursB,contoursY;

 findContours(imgThresh, contours, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
  findContours(imgthresh_Y, contoursY, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
   findContours(imgthresh_B, contoursB, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
```

Figure 9: Define Range for Red,yellow and Blue color .FInd thier contours



```
  FindBiggestArea(&maxi , contours);
 if (maxi != -1) {

    Circle_Color(maxi , contours,radius,center,contours_poly,Scalar(0, 0, 255));
 }

  maxi=-1;
  FindBiggestArea(&maxi , contoursB);
 if (maxi != -1) {

   Circle_Color(maxi , contoursB,radiusb,centerb,contours_polyB,Scalar(255, 255, 0));

 }

  maxi=-1;
 FindBiggestArea(&maxi , contoursY);

  if (maxi != -1) {

 Circle_Color(maxi , contoursY,radiusy,centery,contours_polyY,Scalar(0, 255, 255));
 }

 imshow("Output", drawing);
```

Figure 10: Find the biggest contour ,draw the min-enclosing Circle

Output :