Feature Extractions 2

Date: 20-09-2018

Author(s): Rafał Grasman, Bas Woudstra

Table of content

Task 1: Fast Object Tracking

To track the colored balls the image is firstly converted to HSV format

```
cvtColor(prosessing, result, COLOR_BGR2HSV);
```

Then the image is scanned three times. For each times its checked in parts of the image fall within the color range. This is where the colored balls are filtered from the background. Then the image is blurred to smooth the edges and imperfections and the circles are detected with the HoughCircles function.

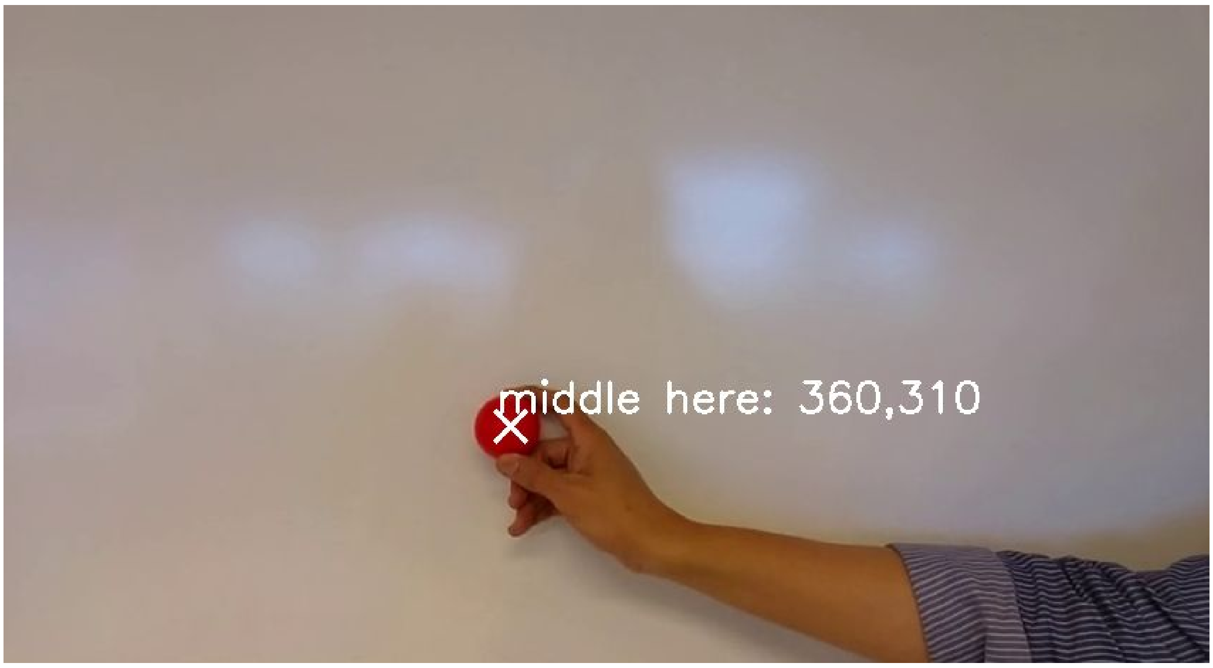
```
inRange(prosessing,Scalar(range[i][0],range[i][1],range[i][2])
        ,Scalar(range[i][3],range[i][4],range[i][5]),color);

Mat blurred;
GaussianBlur(color, blurred, Size(9,9), 2,2);
std::vector<Vec3f> circles;
HoughCircles(blurred, circles, CV_HOUGH_GRADIENT, 1,50, 210,12,10,50);
```

When the circles are detected the points are retrieved from the vector and drawn on the displayed frame.

```
for (int j = 0; j < circles.size(); j++)
{
    Point center(cvRound(circles[j][0]), cvRound(circles[j][1]));
    int radius = cvRound(circles[j][2]);
    std::string text = std::string("middle here: ") + std::to_string(center.x)+std::string(",") + std::to_string(center.y);
    putText(result, text, Point(center.x-10 ,center.y-10), FONT_HERSHEY_SIMPLEX, 1, Scalar(255,255,255), 2);
    //X marks the spot
    line(result, Point(center.x-10 ,center.y+10), Point(center.x+10, center.y-10), Scalar(255,255,255), 3);
    line(result, Point(center.x+10 ,center.y+10), Point(center.x-10, center.y-10), Scalar(255,255,255), 3);
}
```

Result

The program will track the three balls and display the center location of the detected ball



Task 2: Measuring Angle

Firstly the image is converted to HSV and the orange color is differentiated from the background.

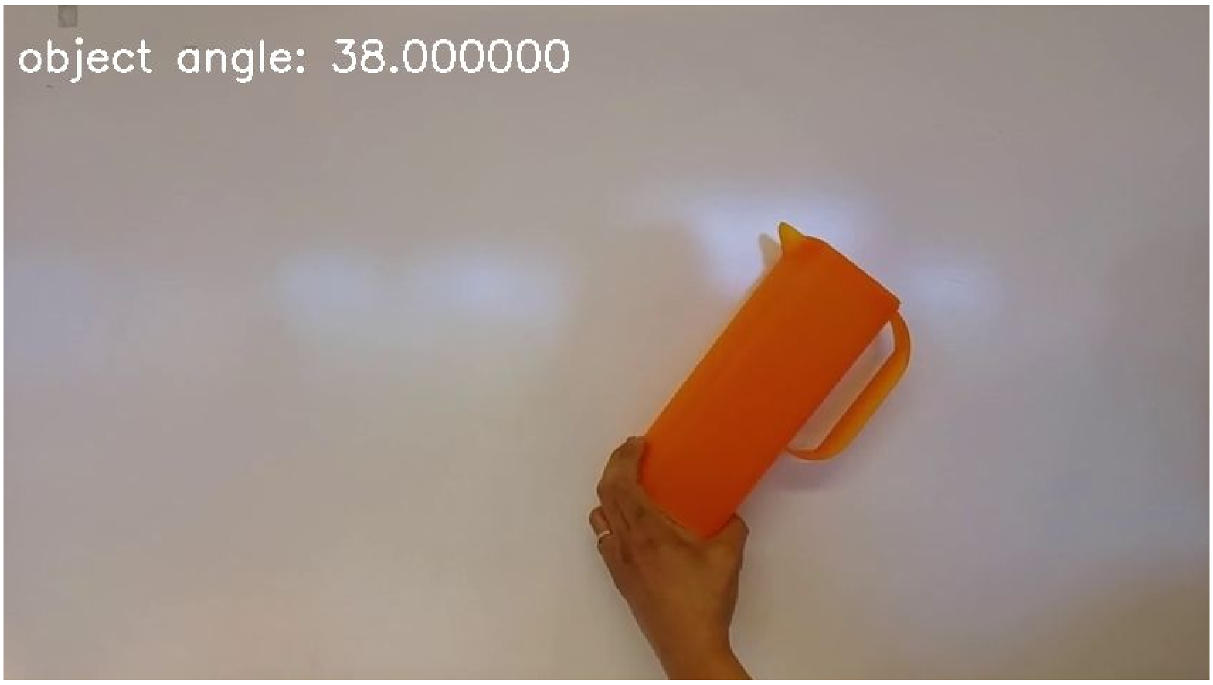
```
cvtColor(prosessing, result, COLOR_BGR2HSV);

prosessing = result.clone();
result = frame.clone();
inRange(prosessing,Scalar(5,230,125)
        ,Scalar(20,255,255),result);
```

Then the image is blurred and canny edge detection is applied.
With HoughLines we detect the long vertical edges of the object.

```
prosessing = result.clone();
GaussianBlur(prosessing, result, Size(9,9), 2,2);

prosessing = result.clone();
Canny(prosessing, result, 100,100);

prosessing = result.clone();
std::vector<Vec2f> lines;
HoughLines( result, lines, 1, CV_PI/180, 80 );
```

The angle of these edges are then averaged and displayed.

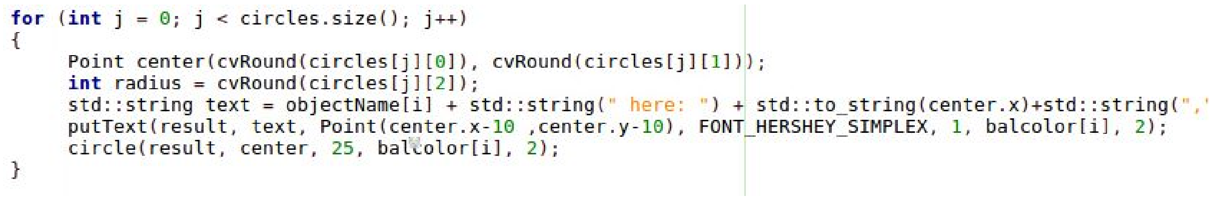
Result
The angle is displayed in the top left corner. Because the angle is retrieved from the HougLines function from 90 to 180 degrees if leaning to the left and 0 to 90 degrees is leaning to the right.



```
object angle: 38.000000
```

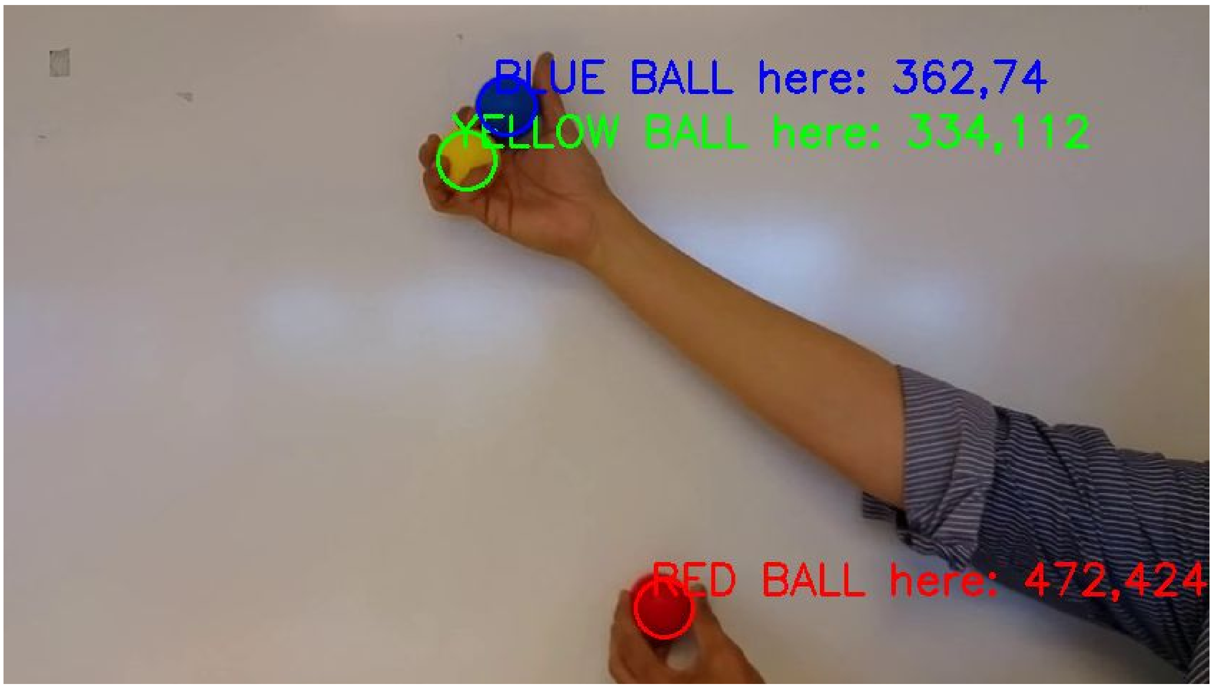
Task 3: Tracking and classifying multiple colored objects.

This application is almost the same as task one, however a slight difference to the displayed image has been implemented to comply with task criteria.
A circle around the ball is drawn and information about the ball is displayed in the same color as the bal



```
for (int j = 0; j < circles.size(); j++)
{
      Point center(cvRound(circles[j][0]), cvRound(circles[j][1]));
      int radius = cvRound(circles[j][2]);
      std::string text = objectName[i] + std::string(" here: ") + std::to_string(center.x)+std::string(","
      putText(result, text, Point(center.x-10 ,center.y-10), FONT_HERSHEY_SIMPLEX, 1, balcolor[i], 2);
      circle(result, center, 25, balcolor[i], 2);
}
```

Result



Task 4: Hand and Finger detector.

For the hand detection a find contour function was used to get the contour of the hand. This function requires a binary image. The binary image cannot be created with threshold only. The function only works with the second part, although it should work with the original part, the reason for this bug is unknown.

```
////not working part
//GaussianBlur( frame, prosessing, Size(11, 11),2, 2 );
//threshold(prosessing, threshold_output,180,255,THRESH_BINARY_INV);

// working part
cvtColor(frame, HSV, COLOR_BGR2HSV);
GaussianBlur(HSV, HSV, Size(11, 11), 2, 2);
inRange(HSV, Scalar(0,0,0), Scalar(255,255,191), HSV);
threshold( HSV, threshold_output, 100, 255, THRESH_BINARY );
```

After that workaround the contours can be detected

```
std::vector<std::vector<cv::Point>> contours;
std::vector<Vec4i> hierarchy;
findContours(threshold_output, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE );
```

Next up the defects are calculated for every contour using the convex hull and convexity defects functions.

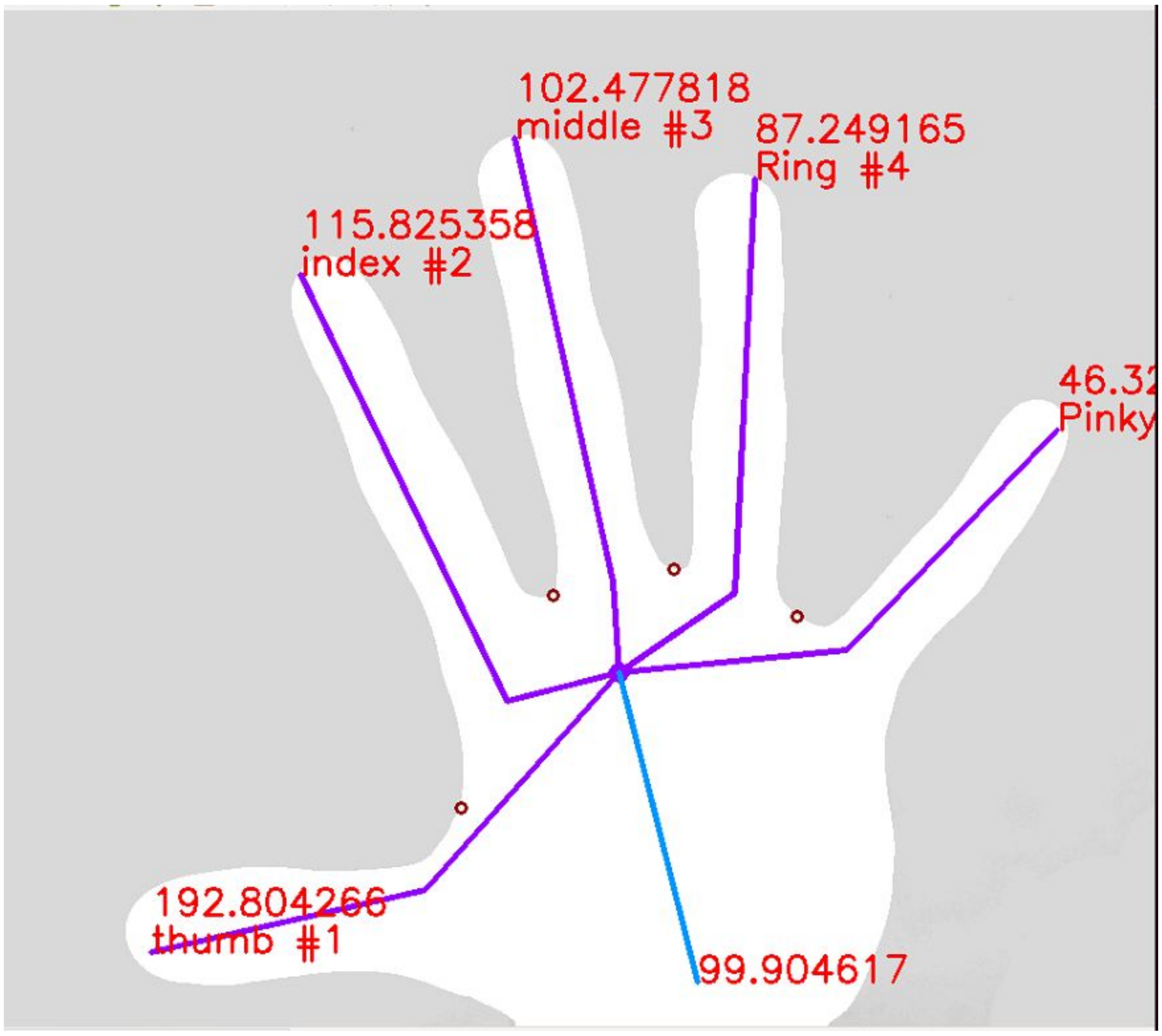
```
std::vector<std::vector<int>> hullI(contours.size());
std::vector<std::vector<Vec4i>> defects(contours.size());
for (int i = 0; i < contours.size(); i++)
{
    convexHull( contours[i], hullI[i], false);

    convexityDefects(contours[i], hullI[i], defects[i]);
    std::cout<<defects[i].size()<<std::endl;
}
```

The contours are also used to calculate the mayor axis and center of mass.

```
mu[i] = moments( contours[i], false );

centerOfMass = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 );
circle(result,centerOfMass,5,Scalar(255,0,150),5);
mayorAxis = 180+(0.5*atan2((2 * mu[i].mu11), (mu[i].mu20 - mu[i].mu02))) * (180 / M_PI);
```

The information about the defects and moments are then filtered and put true an algorithm. This algorithm will draw the fingers, angles and points needed to display.

Result
The fingers with index and degree of rotation are shown. The center of gravity is where the finger lines join with the wrist line. THe mayor axis is shown at the bottom of the hand.


102.477818
middle #3
87.249165
Ring #4
115.825358
index #2
46.32
Pinky
192.804266
thumb #1
99.904617