MIN-VIS-2016

ASSIGNMENT 7 – GROUP4

Group Members:

- Sandio Fongang R
- Armaan Rustami

Assignment week 7 Hand and Finger Detection

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

This work can be done individually or maximum two students.

The purpose of this assignment is as follows: Develop a Hand and Finger detector. On Canvas you will find the file "hands.jpg"

Task:
Calculate using "hands.jpg" also depicted in figure 1, the Centre of Gravity the angle of the major axis of the hand, the tip points, fold points, determine the fingertips using the depth and angle as discussed in the PowerPoint presentation and number each finger starting with thumb as no #1. Show your code of your calculations or algorithms with comments!

Please note when using ConvexHull() to use the correct argument CV_CLOCKWISE or CV_COUNTERCLOCKWISE (clockwise = TRUE or FALSE) .

Figure-1Hands

Extra credit: !!
You might use a pre-recorded video stream of your own hand and apply the detector on that video. You might also wear a coloured hand gloves for better detection if you want to use life video.

You are free to discuss with other students, but take care you hand in your own original material. Don't forget to include your name, maximal 2 students per assignments.

Assignments must be handed in before November the 7th 2016 on a **pdf document** with a well-documented C++ sources and comments on your results. A demo presentation of your applications is mandatory!

2. Solution

a. **Hand And Finger Detection:**

In order to detect hand and finger tips , here are some Steps we took:

- Convert Hand frame colour from BGR to Gray.
- Apply threshold () .*Figure1*

```
Mat thresh;cv::threshold(frame,thresh,165,200,THRESH_BINARY);
```

Figure 1: Threshold

- After Threshold we apply to convert the threshold frame to RBG and then toHSV.*Figure2*

```
Mat hsv;
cvtColor(thresh,thresh,CV_GRAY2RGB);
cvtColor(thresh,thresh,CV_RGB2HSV);
```

Figure 2: Convert To HSV

- Then remove noise we apply erode and Dilate filters

```
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 3: Dilate And Erode

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

```
findContours(bw.clone(), contours_hull, hierarchy, CV_RETR_TREE , true, Point(0, 0) );
```

Figure 4: 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 “COG” label. *(Figure 5)*

```
int findBiggestContour(vector<vector<Point> > contours)
{
  int indexOfBiggestContour = -1;
  int sizeOfBiggestContour = 0;

  for (int i = 0; i < contours.size(); i++){
    if(contours[i].size() > sizeOfBiggestContour){
      sizeOfBiggestContour = contours[i].size();
      indexOfBiggestContour = i;
    }
  }
  return indexOfBiggestContour;
}
```

Figure 5 find the biggest Area

- • Then we apply convexHull in Clockwise and detect convexity , after that we loop through the convexityDefects and filter defects by its depth and name the fingers . .Figure 6

```
vector<vector<Point> > contours_poly(contours_hull.size());
vector<Rect> boundRect(contours_hull.size());

for (int i = 0; i < contours_hull.size(); i++) {
    convexHull(Mat(contours_hull[i]), hull[i], true);
    convexHull(Mat(contours_hull[i]), hullsI[i], true);
    convexityDefects(Mat(contours_hull[i]), hullsI[i], defects[i]);
    cout << defects.size() << endl;
    moment = moments((cv::Mat)contours_hull[i]);
    if (moment.m00 > biggestAreay) {
        biggestAreay = moment.m00;
        COG = Point(moment.m10 / moment.m00, moment.m01 / moment.m00); //COG
    }
    cout << defects[i].size() << endl;
    for (int j = 0; j < defects[i].size(); ++j) {
        const Vec4i& v = defects[i][j];
        float depth = v[3] / 256;
        if (depth > 30) //  filter defects by depth
        {
            int startidx = v[0];
            Point ptStart(contours_hull[i][startidx]);
            int endidx = v[1];
            Point ptEnd(contours_hull[i][endidx]);
            int faridx = v[2];
            Point ptFar(contours_hull[i][faridx]);
            // line(frame, ptStart, ptEnd, Scalar(0, 255, 0), 1);
            //line(frame, ptStart, ptFar, Scalar(0, 255, 0), 1);
            line(original, ptEnd, ptFar, Scalar(0, 255, 0), 2);
            circle(original, COG, 4, Scalar(0, 255, 0), 2);
            circle(original, ptFar, 4, Scalar(0, 255, 0), 2);
            circle(original, ptEnd, 4, Scalar(0, 255, 0), 2);
            putText(original, label[counter], ptEnd, 1, 1.2, Scalar(0, 0, 255), 2, 8, false);

            counter++;
```

Figure 1: ConvexHull ,ConvexityDefects and Name fingers

Output: