Cairo University

Faculty of Engineering

Computer Engineering Department

Image Processing

Final Project

Team #09

**Team Members**

Gehad Mohsen

Reham Abdellatif

Reem Mohamed Ali

Sara Mohamed

18-12-2017

**Game logic and assumption:**

-A real time game with 2 levels written in python using openCV.

-The user can play it by opening his webcam.

-Game sequence:

1) User is given the option to choose between 2 games.

He selects the game by swapping his hand right to choose the first game or left to choose the second game.

Five seconds are given to the user to be ready for moving his then we start taking the data.

2) If he moved his hand right the first game is loaded.

* The second game generates random numbers.
* displays 2 numbers of them on the screen.
* ask the user to raise his hand with the true answer by enetring 2 digits with his hand(the most digit then the least digit).
* between taking the 2 digits the user is given 8 seconds to be ready then we take the answer.
* if he answered true , score will increase by 1 else the lives will be decreased by one.
* when the lives are 0 the game terminates and a game over screen appears.

3) If he moved his hand right the second game is loaded.

* An image contains random different shapes appears.
* the user is asked to press any key to start playing.
* the game detects the types of shapes in the image and count their number.
* we ask the user about the number of each shape.
* if it is equivalent to the number calculated by our program then we print true else we print false.
* we give the user 8 seconds to be ready before taking the answer.
* the second game terminates when the user answer the 8 questions

(How many squares are in the image?)

(How many rectangles are in the image?)

(How many triangles are in the image?)

(How many circles are in the image?)

(How many rhombuses are in the image?)

(How many hexagons are in the image?)

(How many pentagons are in the image?)

(How many ellipses are in the image?)

.Note that at any point you can terminate the game by pressing ‘c’.

**Hand swapping detection part:**

* Threshold the image using inRange function.
* Capture the frame.
* Change the color space to YCR\_CB.
* Threshold based on the upper and lower bound using inRange function.
* Dilate the image to make white blobs larger.
* Use findContour function to get the contours in our image.
* Getting the maximum contouring area. We assume that the user will put his hand close to the camera so that the larger area with skin color will be his hand.
* Then we take many frames when the user start moving his hand
* We use boundingRect to get some parameters of our contours, like its starting x,y points.
* For every frame we check the change in x value of the boundary rectangle.
* We calculate the difference in x between each frame and the previous so that we know if he moved left or right!

-In this part we faced a problem detecting the skin color. We searched for global skin hsv values to consider but they returned very bad values for example if we were setting in a room with wall colors like the image shown below, the maximum contour area will be the wall not your hand!

This problem was solved when we changed the color space to YCR\_CB and we got much better results.

**Detect Shapes From Image(part 1):**

* First we get edges from image and fill every  
  Shape’s edge with white color
* Then divide image to images every image contain one shape by using measure.label
* Then read every image and get from it interested points  
  We try to use harris function to get corners but we get wrong results
* So we use findContours that return interested points in image
* Then we use minAreaRect that take points and paint rectangle contain shape. This function return centers of rectangle and It’s width and height
* We can get area of the shape by sum the white pixels in every image
* Then we can get ratio=((height\*width)/number of white pixels)
* If ratio value near to 1 that mean this shape is rectangle or square
* So we can get ratio2= (height/width) if ratio2 value is near to one so it’s square else it’s rectangle
* If ratio value isn’t near to one so we apply hough transform

**Detect Shapes From Image(part 2):**

In this part we used the following algorithms in the final code:

1.Hough transform

2.DSU

First we transform the RGB image into a binary image, and to separate each shape in the original image so that we can process it, we thought of using 8-flood-fill algorithm and by filling each shape in the binary image with certain color that is increased by 1 when entering a new shape, so after filling each shape with specific color and having a list of these colors according to the number of shapes, we could separate each shape by thresholding about the color of that shape, but this idea failed due to the very large number of recursive calls needed for one image.

After that we used measure.label to separate shapes/ divide the big image into shapes, each shape is in his own image.

After using findContours and having “ratio” variable we can detect that if “ratio” is less than 0.9 or greater than 1.2 this means that the shape we have now is neither a square nor rectangle so it is one of the other six shapes.

To detect the shapes we thought of using Hough Transform algorithm,then get the edged image with the help of canny algorithm and to detect some data about the edges produced by Canny we used the function “HoughLines” which returns an array of pairs of rho and theta (each rho of an edge has its own rho), each edge is given an index, but when using this function we faced many problems as:

* The multidimensional array returned by “HoughLines” was difficult to get the values or the size of this array, but later after trying many times we got the required values from the returned array.
* When we have a circle or an ellipse, “HoughLines” returns no edges, hence the size of the returned array is NONE and that have made us sometimes have some errors when trying to access NONE, later we fixed this problem
* Another problem is that when the threshold passed to function “HoughLines” increases this made “HoughLines” returns less indices then couldn’t know the number of edges and couldn’t know the shape type, and when threshold decreases, “HoughLines” returns more edges than we do have, but this problem to be fixed we found due to tracing many times that the edges exceeding the number of original edges have rho and theta very close to original edges, so using DSU we could detect what are the edges that are same as other edge and make it child to the original image, In this way we could have the real number of edges in the given input image.
* By tracing we found that the perfect threshold to be passed to “HoughLines” was equal to 70. This value was perfect for detecting the shapes, but more than that value will remove some original edges and below that value more extra edges are returned. But we still have some extra edges that are approximated to the original corresponding edge using DSU to get the real edges number .
* Also “HoughLines” when the pentagon, hexagon and rhombus are very small, they are treated as circle so the value returned from “HoughLines” was NONE although this NONE is returned only for the circle and ellipse.

After we got the array from the “HoughLines” function we test it to know its type:

-If the the array is NONE then we detected a circle or ellipse, to differentiate between them, the ellipse’s height and width of the conour containing it is not 1 then it means that we have ellipse, else we have Circle

-If the array is not NONE then we test the size the array or the number of original edges, if we have 6 edges then we detected a hexagon, when having 5 original edges then we have a pentagon, when having 4 edges(and the “ratio” variable is not between 0.9 and 2.1) then we have a rhombus, and at last we have triangle.

To detect the number of each shape in the given input image we put the results in a dictionary that have each shape has its corresponding number in the image.

This dictionary is used to know later in the game that if the player counts the number of each shape in the given image right or wrong.

**Workload**

|  |  |
| --- | --- |
| Reem Mohamed Ali | Shapes detection-part2(Hough Transform, Canny edge detection) and integrating the shapes detection parts |
| Reham Abdellatif | Shapes detection-part1(Harris,minarearect, contours) |
| Gehad Mohsen | Fingers detection then numbers detection |
| Sara Mohamed | Hand swapping detection, upgrading the game to level 2 to have also the questions that ask about the number of shapes in a selected image and integrating all the parts of the project |