Computer Vision Assignment № 3

Metehan Seyran, 150170903, Istanbul Technical University 21/12/2020

Part 1 - Sobel Filter

In this part it is required to implement an edge detection algorithm using Sobel Filters. Convolving 3x3 filters inside the screenshot takes couple minutes because the convolutions are written from scratch, which might be inefficient. The result image will contain the edges from all shapes page. In the implementation, before feeding image to the convolution operation, the image passed through Gaussian Blur function which is used to get rid of the noise, and make image smoother.

Part 2 - Canny Edge Detector

In this part, I have used the Canny Edge Detector function from OpenCV. The parameters for low high values are selected as 30 and 200. As a result, the edge map of the image is returned.

Part 3 - Minimum Eigenvalue Corner Detector

In this part, it is required to implement the minimum eigenvalue corner detector algorithm from scratch. The algorithm traverses through the image provided, and at every pixel it calculates the sum of the specified window-sized area. This will give us the matrix G, which we need to find eigenvalues in order to make a prediction whether the point is a corner or not. At every point, the eigenvalues are calculated for matrix G, and finally if the lowest eigenvalue is bigger than the threshold, 25, then that point will assigned to the corner list. For the last part, I mark the corner point with red dot, and the function will return the image with the red dots located at the corners.
Since the implementation is written from scratch, it will require couple minutes to run.

Part 4 - Dance Game

In this part, it is required to play a game. In order to gain points, we must classify the passing shapes correctly and press specific buttons for each shape before it reaches the reddish box.
The algorithm firstly takes a screenshot of the game, then crop the approximately middle part of the bottom part where shapes are passing. Secondly, *findContours* function will find the cropped image's contours. Finally the function *approxPolyDP* will approximate the contours into a polygonal shape returning list of approximated corners. By counting the corners, we can

predict the shape and return the correct button to press. If the number of corners does not match with the existing shapes, then nothing happens.

Compilation Details

All of the 4 parts are located inside 1 python file. During compilation, extra argument is required to enter, which is the part number (1-4). The example compilation command should look like this;

```
> python assignment3.py --part 4
```

The code will run forever without a stopping criteria, that is why, user should kill the running process in manually.