Solution and Answers on Homework 2

Solution Explanation:

In this Homework we were tasked to implement a script to detect a colored ball, estimate its 3D position, and draw a 3D cube over the ball.

To start we were provided with a script that extracts the input.mov as a movie and loops through it to extract and process each frame.

We were tasked to use the MagicWand object and its process_frame() method to process each frame. The MagicWand object already has methods and their input/returns predefined, so we just had to fill them in with code and use the methods in the process_frame() method.

process_frame() itself takes a frame and returns all detected circles and their corresponding 3D Coordinates.

In process_frame() we firstly use the detect ball method. Which takes in the frame, grayscales it applies a gaussian blur to it and then uses the cv2.HoughCircles() function to detect any balls in the provided image. The return of HoughCircles value is then modified to return the desired list of (x, y, radius) tuples.

After that we check if process_frame() found any circles and if so loop through the circles.

For each circle found we then calculate the X, Y, Z Coordinate using the calculate_ball_position() which takes the (x, y, radius) of the circle and uses the projection equations as discussed in class to compute X, Y, Z. The needed focal length and real-world radius of the ball were provided in the predefined script. (which reads in a config file containing those values)

After we have the X, Y and Z coordinates we use the draw_ball() method which was already prefilled with the necessary functionality to draw a circle around the ball as well as writing the Z coordinate in cm on the ball.

After that we call the draw_bounding_cube() method to draw a cube around the ball.

For which we first must calculate each of the 8 possible corners of the ball and then call the draw_line() method which takes two 3D points and draws a line in-between them.

The draw_line() method is called a total of 12 times to account for the 12 edges of a cube.

How the corners 3D coordinates are calculated as well as how the corners are combined to draw the edges is best observed in the script.

The draw_line() method then uses the project() method to project the 3D coordinates of the corners to 2D image coordinates using the projection equations.

After that draw_line() takes the two corners and their coordinates and draws a line using cv2.line().

Once that one frame has been processed, meaning one or more balls have been detected, the distance, the bounding circle as well as the bounding cube are drawn into the frame.

After that we return all found 3D circles and their 3D coordinates to tracker.py.

In tracker.py we then process the found balls. Depending on the number of balls detected we have two cases. One for only one ball detected which is used to later plot a 3D trajectory of the ball. And one for two balls detected which is used to later calculate the distance between two balls on a wand.

As script does not differentiate between a wand with 1 ball and wall with 2 balls, we must keep that in mind while using the script on a input.mov video.

In case of our 1 ball, we just collect all single circles in a list and use matplotlib to plot its 3D trajectory.

In case of our 2 balls on a wand we calculate the distance of the two balls using the 3D distance formula and collecting the distances in a list to mean all the distances and get a mean distance in the end.

Note the comments in the code and the code itself for further clarification and the corresponding return types.

Qualitative Evaluation Questions:

- Q: Does the ball detector seem accurate? When does it fail to detect the ball and why?
    - A: It seems as that the ball detector is fairly accurate. It detected the single ball in blue.mov 92% of the time for example. After observing the images where the ball wasn't detected the image seemed kind of blurry, so that may be a reason why the ball detector of OpenCV fails.

- Q: Is it possible to correctly rotate the 3D box according to the ball's orientation? Why or why not? If not, how could we physically modify the magic wand so that we could calculate the correct rotation of the box?
    - A: With the movies provided it is currently not possible to rotate the 3D box according to the ball's orientation because the ball only consists of one color. As a digital image is only a 2D projection of the 3D world, the depth parameter gets lost. Because of this any rotation of the ball will look like the circle (the 2D ball) does not change. It may be modified by drawing reference shapes on the ball (i.e., dots on the ball) itself or maybe add another ball or any easy shape that can be detected on one side of the magic wand to keep track of the rotation of the ball.