Title: Determining the distance between two points and the size of objects in an image

Authors:
Kevin Zhang 101148146
Ning Hu 101151560
Satsang Adhikari 101145635

Abstract:
For this project, we came up with a program that takes in as input an image and calculates the real-life distance between the objects within that image. The program was built using the Python OpenCV library. We completed several milestones for the project. First, we started off with detecting all the objects in the image that were circular in shape. Then, we detected all the objects in the image that were rectangular in shape. Afterwards, the program was able to outline boxes around each object. Then, the program was able to compute the size of each object. Finally, we were able to compute the distance between the objects in the image. For each of the above milestones, we considered various edge cases. The final result is a program which was tested meticulously to determine the real-life distance between two objects as well as the size of objects in an image.

Introduction:
This project is inspired by an argument we had about the height of objects while in a public space together. We argued about the height of a building and even placed monetary bets, however we were unable to determine its real height (i.e. the distance from the bottom to the top of the building) without any sufficient measuring devices. Essentially, the project followed the thought of “I wish we could use an app that uses the camera to determine the distance”. There were several challenges we faced when working on the project. Firstly, determining whether repeating patterns in the background of the image were counted as noise or actual objects to find the size / distance of. Secondly, what would happen if the object we are identifying was very tiny, if there were shadows in the image or if the object was partially out of frame. Furthermore, we had to consider the case where objects were stacked on top of each other or adjacent to each other in the image and decide whether or not they are considered separate objects. Of course, there was also the challenge of classifying objects as either circles or rectangles, for size purposes. Also, we need to keep in mind that each image improves the results with its own set of parameter values for the circles inside the cv2.houghCircles() and so different values yield different results.

Background:

Finding-distance-between-objects-in-an-image-using-OpenCV/distance_between.py at master · VigneswaranB97/Finding-distance-between-objects-in-an-image-using-OpenCV (github.com)

Our project source code was inspired from the above GitHub repository. It is also a program for finding distance between objects in an image using Python OpenCV. At a high level, the following are the steps our inspiration project takes: it takes in an image to read, performs edge detection, finds contours, draws outlines of the box around the object, computes midpoints, and calculates the distance after drawing a line between the midpoints.

We were inspired by the approach to use for our project while also having some original features of our own. Firstly, we have a functionality that detects all circular objects as well as all rectangular objects in the image. Another major difference is that the inspiration image calculates the distance from the left-most object to every other object in the image. But we calculate the distance between all objects ordered from left to right.

Methodology:

For the approach in which we conducted our experiment, we initially analyzed different factors and variables that could influence the outcome. By using a variety of different images, we tested each of them to evaluate the results from the experimental images and how our code detected objects within the image. Our data set is composed of images with a varying set of different sizes, quality, scale, depth, focus, number of objects, etc. This approach allowed us to assess the accuracy of our object detection and the distance measured between them and make sure our program works under various conditions.

As arguments to the program, we first take the path of the image file and the width of the left-most object to be detected in the image as a reference. The left-most object in the image is used as a reference for the size and distance calculations in the entire image.

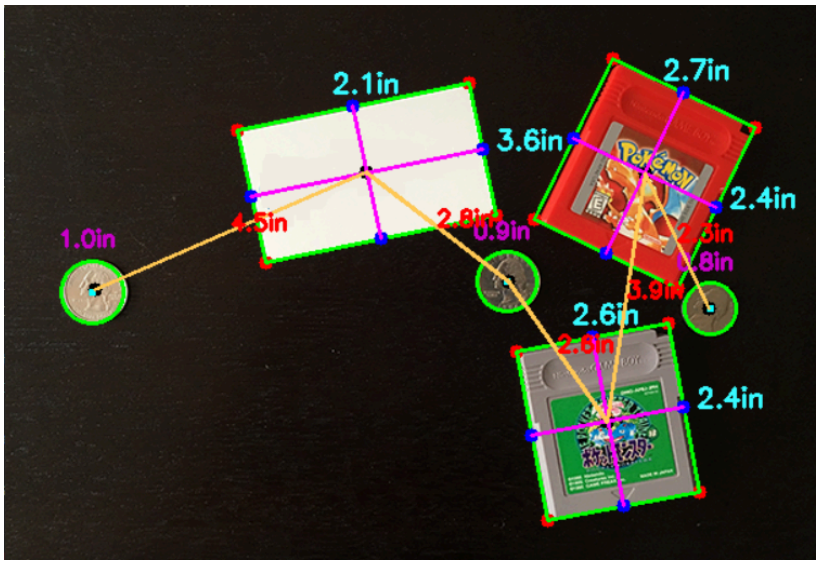
First, we convert the image to grayscale since all we need is one dimension of intensity to detect edges / objects. We then blur the image so that small details in the image will be considered as noise rather than as objects. We then use the Hough method of finding circles in the image, where circles will then be eliminated from the rectangle detection later on. For the rectangle detection we use Canny edge detection followed by the findContours function to get rectangle-like objects.

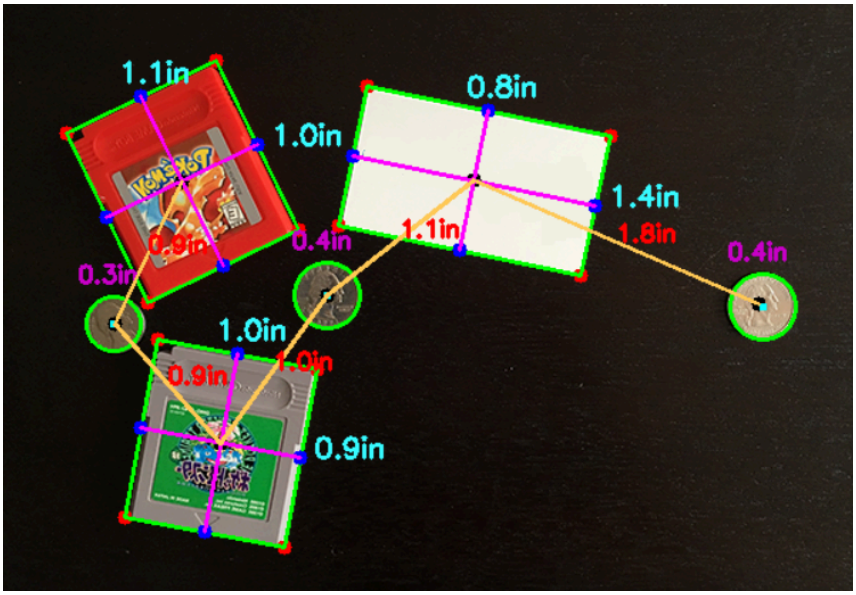
To eliminate circle-like objects from the contour detection, we compare the center of the Hough detected circles with the midpoint of the bottom left and top right points of detected rectangles, and if they are very close we use the object as a circle rather than as a rectangle.

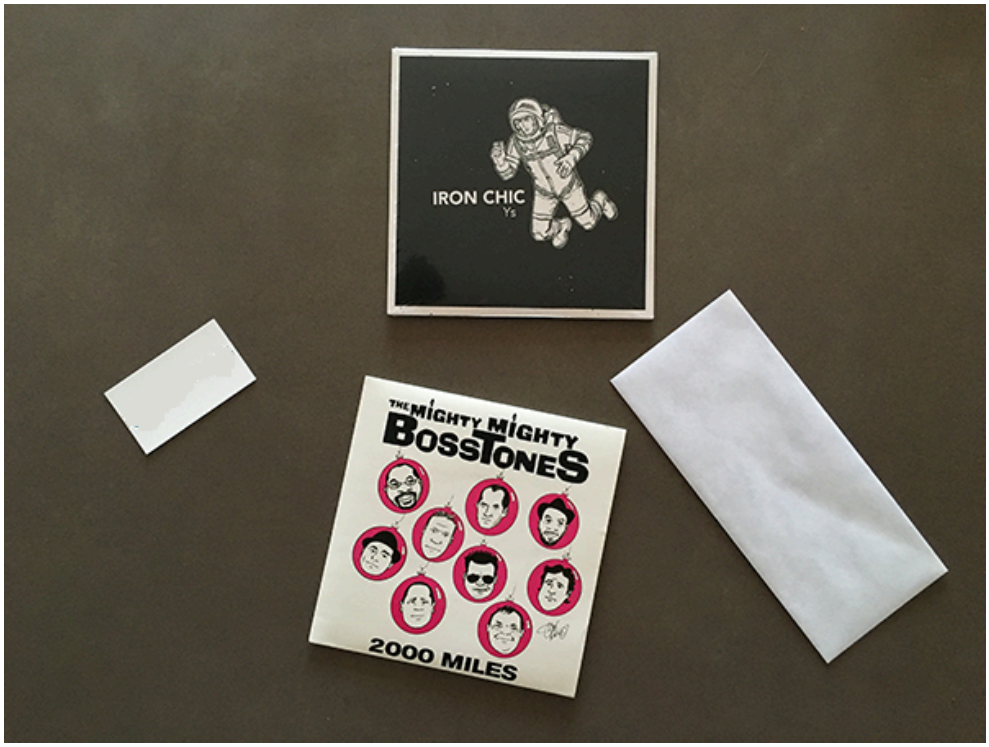
To find all distances and lengths we calculate euclidean distance in pixels divided by pixels per metric (pixels per meter, centimeter, etc.) of the left-most object to get the actual distance length between the two points.

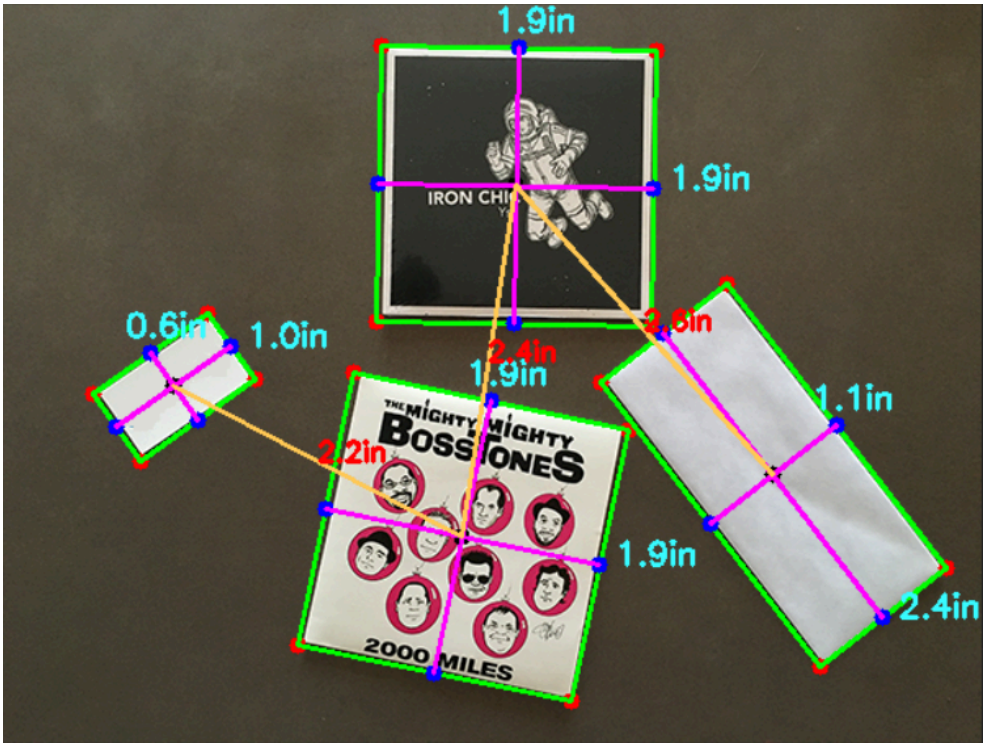
Results:

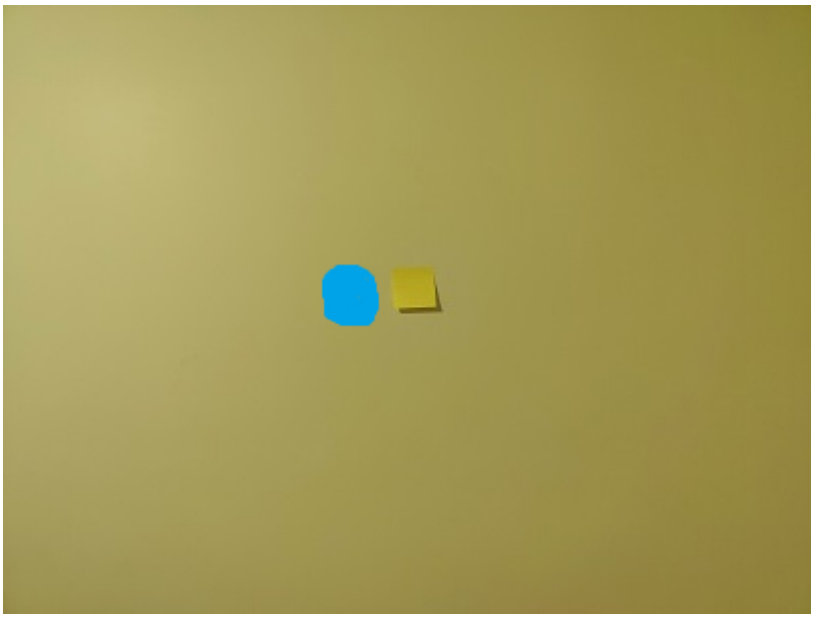
The numbers colored in purple above the circle objects represent its diameter. The numbers colored in turquoise around the rectangular objects represent its length and width.The numbers colored in red represent the distance between the two corresponding objects.

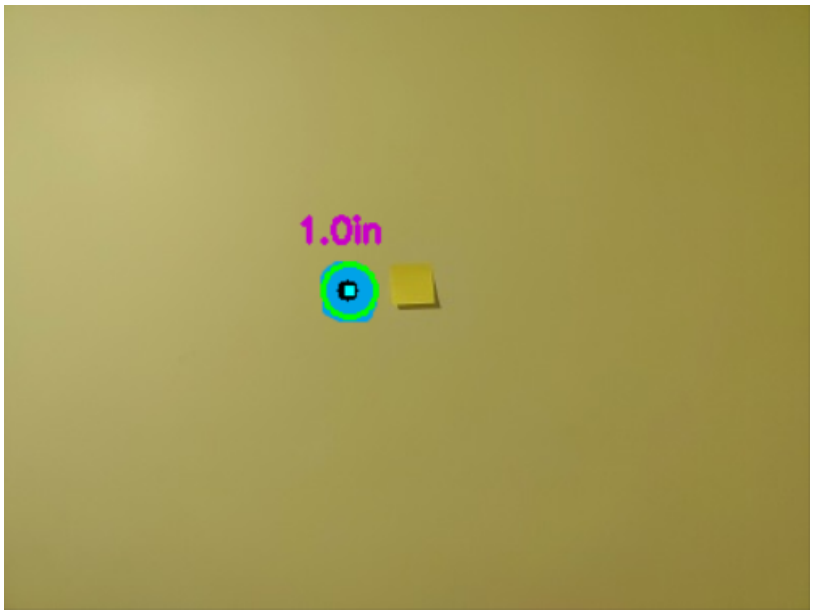


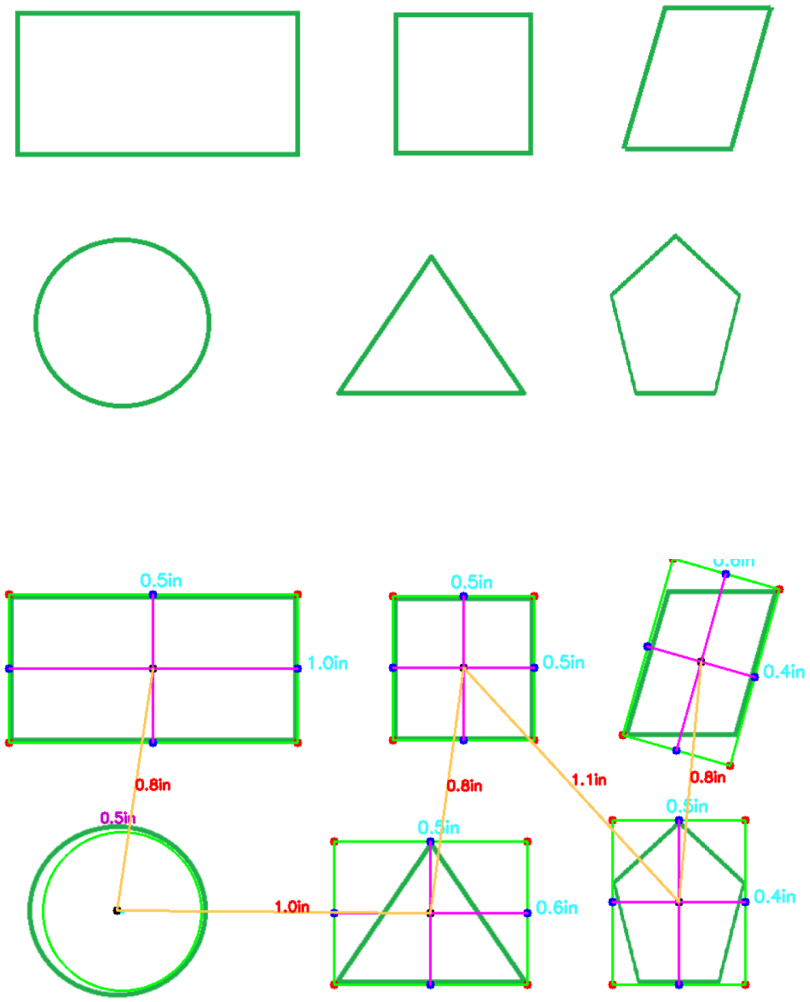

1.1in
1.0in
0.8in
1.4in
1.1in
1.8in
0.4in
0.4in
0.3in
1.0in
0.9in
1.0in
0.9in


IRON CHIC
THE MIGHTY MIGHTY BOSSTONES
2000 MILES


1.9in
1.9in
0.6in
1.0in
2.4in
1.9in
2.6in
1.1in
2.2in
1.9in
2.4in
2000 MILES




1.0in



Conclusion:

With all the results gathered from our program, we are able to correctly detect circular and rectangular objects and compute their sizes and measure the distances between them for the most part. This application showcases the practical implications for industries like robotics, surveillance and autonomous driving. However, the project is not without its limitation. It relies on needing a specific object on the left-most side to be detected in the image as a reference and the accuracy value can be influenced by external factors such as lighting and image quality.

For future developments, integrating the detection of 3d objects could significantly improve the programs capabilities. This advancement would address the current limitations of needing specific types of images and would broaden the range of detectable objects and improve accuracy. This would also include more complex environments and backgrounds of images, allowing for better understanding of how objects relate to each other in the same space.