Table of Contents

1.0 Objectives

The proposed project is the building of an image recognition system built in Python. The main goal of this system is to identify the objects in a given scenario as much as possible. The system will define a scene with as much plausible information as possible, such as the amount of object in the scene, the shape of the objects, the size of the objects, etc. The system accomplishes this by preprocessing the given digital images with functions from the OpenCV library, such as filtering, drawing contours, thresholding, and more. Image preprocessing provides an easier way for the computer to understand the information inside the image and semi-accurately identify the objects in the images. The system will then present the information it obtained from the digital image to the user through outlining and labelling the objects with indices and any other extra information.

2.0 Problem Statement

The proposed system is an automated scene description with image processing. The system is able to analyze the predefined image and perform image processing that considers the lighting conditions, visual revolution, and other elements that will improve a noisy image. Its primary purpose is to recognize various objects in the image.

As humans can only process an image in the form of brief glimpses, determining characteristics of everyday objects with an autonomous system by processing images can show a deep understanding of the image. Take a car picture as an example, humans only recognize the car type, colour, and accessories. However, the computer will break the car picture into many small pixels and check the car paintwork, body panels, and every detail that the car appearance shows.

When simply reading an image, the computer itself does not know much about the information of the image, such as the amount of object, the shape of the object inside the image, etc. With image processing techniques, the computer can improve its image understanding by converting the raw data into colour-corrected images.

Its application includes digital cameras, film production, and face recognition techniques. The camera in our phone has the function of recognizing captured objects with the help of the image processing technique. For example, focusing on a cake with the camera will recognize the picture as food and implement a relevant filter to beautify the image. Film production requires a high image quality to show a more apparent environment scene to provide a good user experience. The face recognition technique implements image processing to outline the face shape, describe the skin tone, and identify facial features such as eyes, eyebrows, ears, mouth, and nose.

3.0 Literature Review

Image processing has been a popular field of research over the past decade, one topic in particular is object recognition. Object recognition is considered as one of the most difficult areas due to the fact that images and videos have jumbled backgrounds, occlusions, variations in lighting, weathers and others. In 2018, research was done to illustrate a new approach to object recognition when there are different environmental situations (Singha & Bhowmik, 2018). In this research, the proposed system is a patch feature-based method over similarity concept of image block system. The system operates by splitting an image to individual blocks, then within each of these blocks, a Representative Score is calculated based on the features. The Representative Score is derived from merging the scalar value of Akinity and Liability; these values help calculate which point on the picture should be detected as an object. With the given RP, the researchers were able to classify the objects from its surroundings given the environmental conditions. Singha and Bhowmik (2018) then tested the system with their own collected video clips, which were taken with a NIKON Camera. This research provided a great contribution to the field of object recognition, by distinguishing classifiable objects from its background in addition to weather effects such as fog, rain, and dust. Future research could implement this method in various other scenarios where video quality may not be clear enough, or the environment is not suitable for object detections.

Recent advances in object recognition have also allowed for more applications such as in the realm of image captioning, though often relying on machine learning algorithms for identification of the objects themselves. This process is typically achieved through the system identifying the likelihood of each object type according to its shape, arrangement of contours, sides, and edges. Caption generation is then achieved by comparison of the image features with labelled sample images from the database. A solution was proposed where localized segmentation of parts of the image is conducted to separate them into layers for identification of the overall image. The individual layers are then looked at for context, with semantically meaningful regions (*Fu et al., 2017*). However, an argument in segmentation robustness arises when the images in question are not of decent quality. Specifically, multiple factors such as illumination, artifacts, occlusion, and background may interfere with the ability to recognize a particular object. In conjunction, a recognition algorithm based on sparsity and collaborative representation was

proposed, which managed increased detection rates with minorly occluded images, achieving a 99.1% recognition rate (*Deng D, 2020*). Additionally, a proposed use of deep learning classification algorithms in the realm of image classification of weather conditions has been seen to usurp traditional feature-based detection algorithms achieving only 71.4%, and by a significant margin, (*Kang et al., 2019*).

4.0 Algorithm description and justification

4.1 Thresholding

On startup of the program, it will first prompt the user to key in the color of the image background. Depending on the color of the background specified by the user, the program will select the threshold type and value accordingly via an if-else control statement. After performing many tests and adjustments to the parameters for the thresholding function, we have concluded the following values that generate the best result for the threshold function:

Background color	Threshold Type	Threshold Value
White	THRESH_BINARY_INV	240
Black	THRESH_BINARY	75
Non-specific	THRESH_BINARY_INV	100

Table 1: Threshold values

Thresholding can only accept grayscale images, therefore gray scaling will be applied to the input image alongside blurring and morphing to clean up the image and reduce noise. After that, the program uses the cv2.findContours operation to identify the edges of the objects segmented from the background.

4.2 Find contours to identify objects

After pre-processing and finding the contours of the image, the program calculates the area within the connected contour. The program evaluates the area, and we set 500 pixels to be the benchmark for which the program considers the connected contours to be an object. The program draws the contours identified onto the original image to visualize the identified object.

4.3 Counting the number of objects

After obtaining the corresponding area of each connected contour, the program appends them into an array. Using the len() operation on the array, the program can count the number of objects.

4.4 Getting the largest and smallest object

By obtaining the area of the objects, the program can compare between the areas and identify the largest and smallest object within the image using the max() and min() operations. The ".index" can be used to get the id of the respective object.

4.5 Identify shape of object

The program can also identify the shape of the objects identified from the connected contours. Using the cv2.arcLength() function, the program can get the perimeter of the contour. A second parameter is set to “TRUE”, specifying that the contour must be connected, indicating an object. The cv2.approxPolyDP() function approximates the curve drawn by the contour to an accuracy specified by the second parameter “e”. Again here “TRUE” is set for the third parameter to indicate the contour must be connected. The coordinates of the approximated curve is stored in an array. We can get the length of the array to obtain the number of corners of the object. Using the corners of the object, we can run through an “if-else” control statement to identify the shape. E.g. if there are 4 corners, it will be a rectangle, if there are 3 corners, it will be a triangle. For objects above 10 corners, the algorithm considers it to be a circle.

4.6 Visualization

For each object identified by getting the contours of the image, the program displays the if of the object, the number of corners for each object, and the shape of the object. The contours are also drawn onto the original image, and the shape of the object is labelled onto each identified object.

4.7 Color of object

In another program, we identify the colors of objects by converting the input image into an HSV image. We can use this converted image to identify the color of objects within the image through a spectrum of colors. For each color in the color spectrum, the program masks out the color from the image. After masking the colors from the image, the program finds the contours of objects and draws them onto the original image to identify the objects with the corresponding color. The program also labels the color onto the identified object.

5.0 Test plan and results

To test our program and algorithm, we have obtained two images from the internet containing simple objects. We have also used 6 images we took ourselves with real life objects that we can find around us. Our program can detect objects in images as well as identifying colors.

5.1 Object detection

16 shapes image

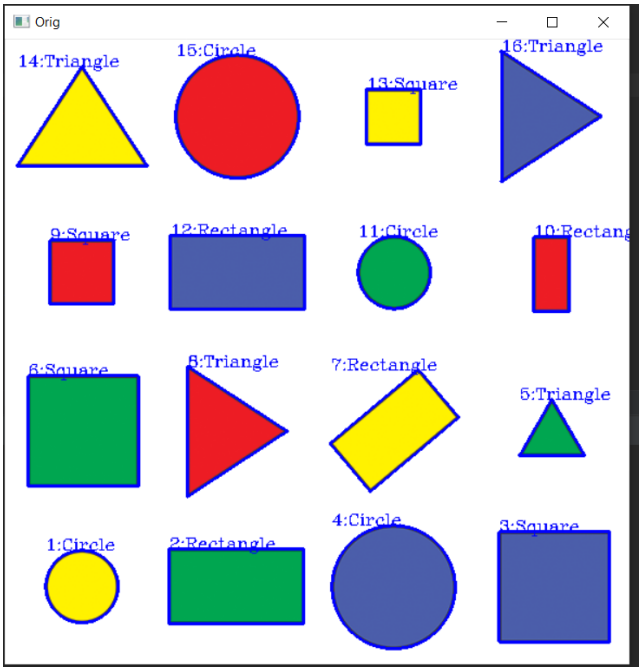

Figure 1: 16 shapes image object detection result

```
The number of objects in the scene is:
16

Largest object in the scene is id:  3 Smallest object in the scene is id:  5
```

Figure 2: 16 shapes console result

The first picture contains a white background with 16 simple shapes. The algorithm works very well for this image as it has objects that have a single color, with well-defined edges and high contrast compared to the background. The algorithm was able to identify all 16 objects and label all shapes correctly.

Fruits Image

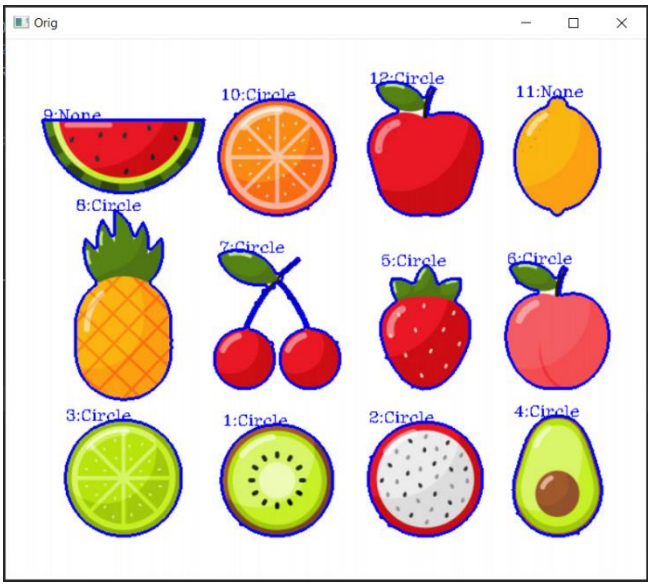

Figure 3: fruits image object detection result

```
The number of objects in the scene is:
12

Largest object in the scene is id:  8 Smallest object in the scene is id:  11
```

Figure 4: fruits console result

The second sample image contains a white background with 12 fruits, the algorithm was able to identify all 12 objects. This image was also another easier picture for the algorithm to achieve a high accuracy as the objects have high contrast compared to the background. However, the fruits are all in different shapes, therefore it was difficult to identify the exact shape.

Test image 1

Figure 5: Test image 1 object detection result

```
The number of objects in the scene is:
3

Largest object in the scene is id:  1 Smallest object in the scene is id:  3
```

Figure 6: Test image 1 console result

The image above contains five objects. The algorithm was only able to identify three objects out of five. For the object with id 2, the contour drawn was not accurate due to the object having two sections with different contrasts. The part with very little contrast with the background was not contoured, causing the object identified to be an irregular shape. There is also shadow around object id 1, causing the contour drawn to be irregular and not matching the object's true edges.

Test image 2

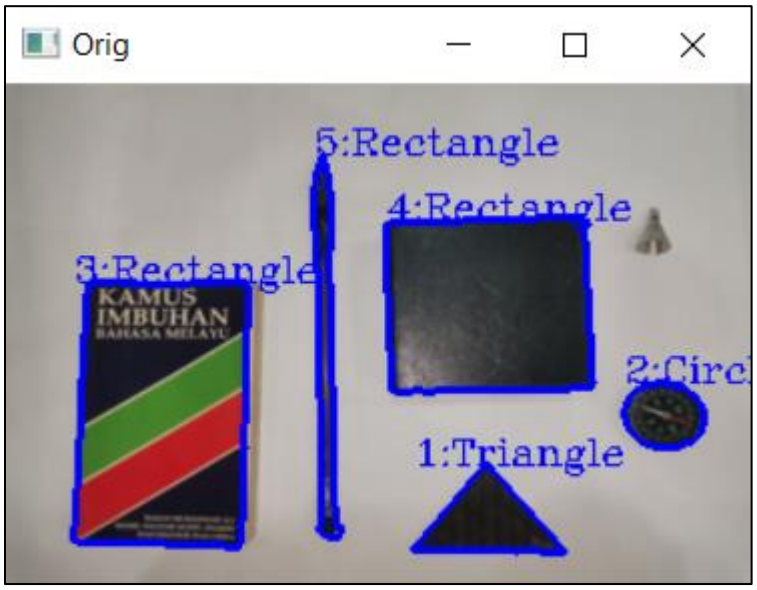

Figure 7: Test image 2 object detection result

```
The number of objects in the scene is:
5

Largest object in the scene is id:  3 Smallest object in the scene is id:  2
```

Figure 8: Test image 2 console result

There are six objects in test image 2. However, the system can only recognize five objects and label them from 1 to 5. It may be the reason of its small size and results in an unidentifiable object.

In this scene, the system identifies object ID 3 as the largest object and object ID 2 as the smallest object. If the top right-hand object is detected, it will replace the object ID 2 and become the smallest object for this set of image input.

Object ID 1 is detected for having only three corners and is identified as a triangle, which is similar to the human's sense of vision. From our view, object ID 2 is a circle shape, and the system correctly labels it as a "circle." The system recognizes objects ID 3,4 and 6 as rectangle shapes. However, from the human aspect, only object ID 3 tallies with the rectangle requirement. Object ID 4 should be a square shape, while object ID 5 should be recognized as "none" as it possesses strange shapes. Overall, for this test image, the system reaches a 50% success rate.

Test image 3

Figure 9: Test image 3 object detection result

```
The number of objects in the scene is:
8

Largest object in the scene is id:  8 Smallest object in the scene is id:  2
```

Figure 10: Test image 3 console result

In this test image, three objects, namely a clock, ruler, and a tissue box, was captured on top of a multi-patterned mouse pad. Upon processing with our code, immediately there is visible difficulty in detecting the contours of the objects.

Upon processing, the system detected 8 different objects in the scene, Object ID 8 being the largest object, and Object ID 2 being the smallest object. The objective here was to detect the three objects, which the system failed to do so in this case. A solution is discussed in the critical comments and analysis section.

Test image 4

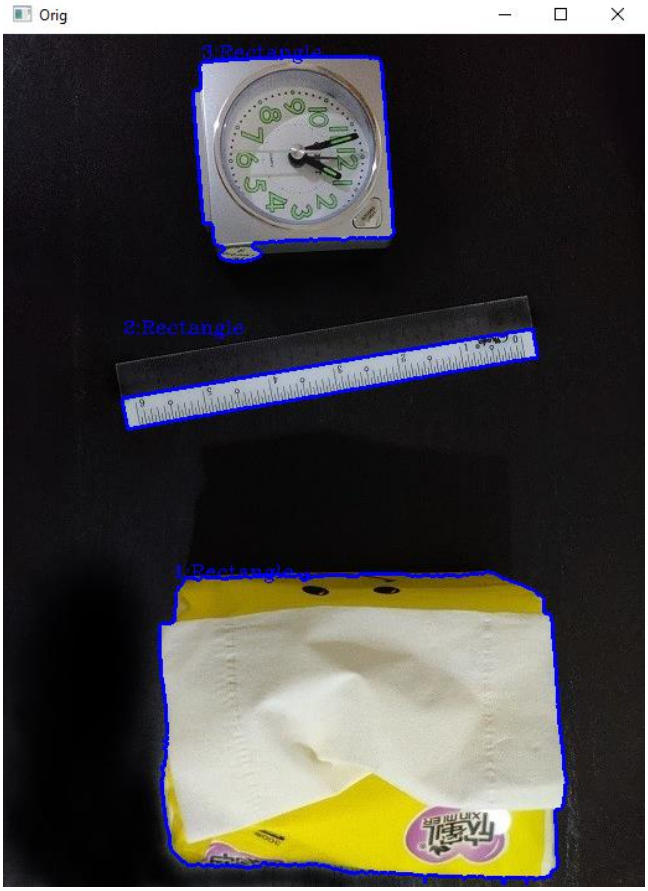

Figure 11: Test image 4 object detection result

```
The number of objects in the scene is:
3

Largest object in the scene is id:  1 Smallest object in the scene is id:  2
```

Figure 12: Test image 4 console result

Testing again with the clock, ruler, and tissue box, but this time on a black table. The system was able to detect all three objects almost flawlessly. The only problem here was the ruler was only detected in half, as the transparent half was not detected. Everything else is working accordingly.

Test image 5

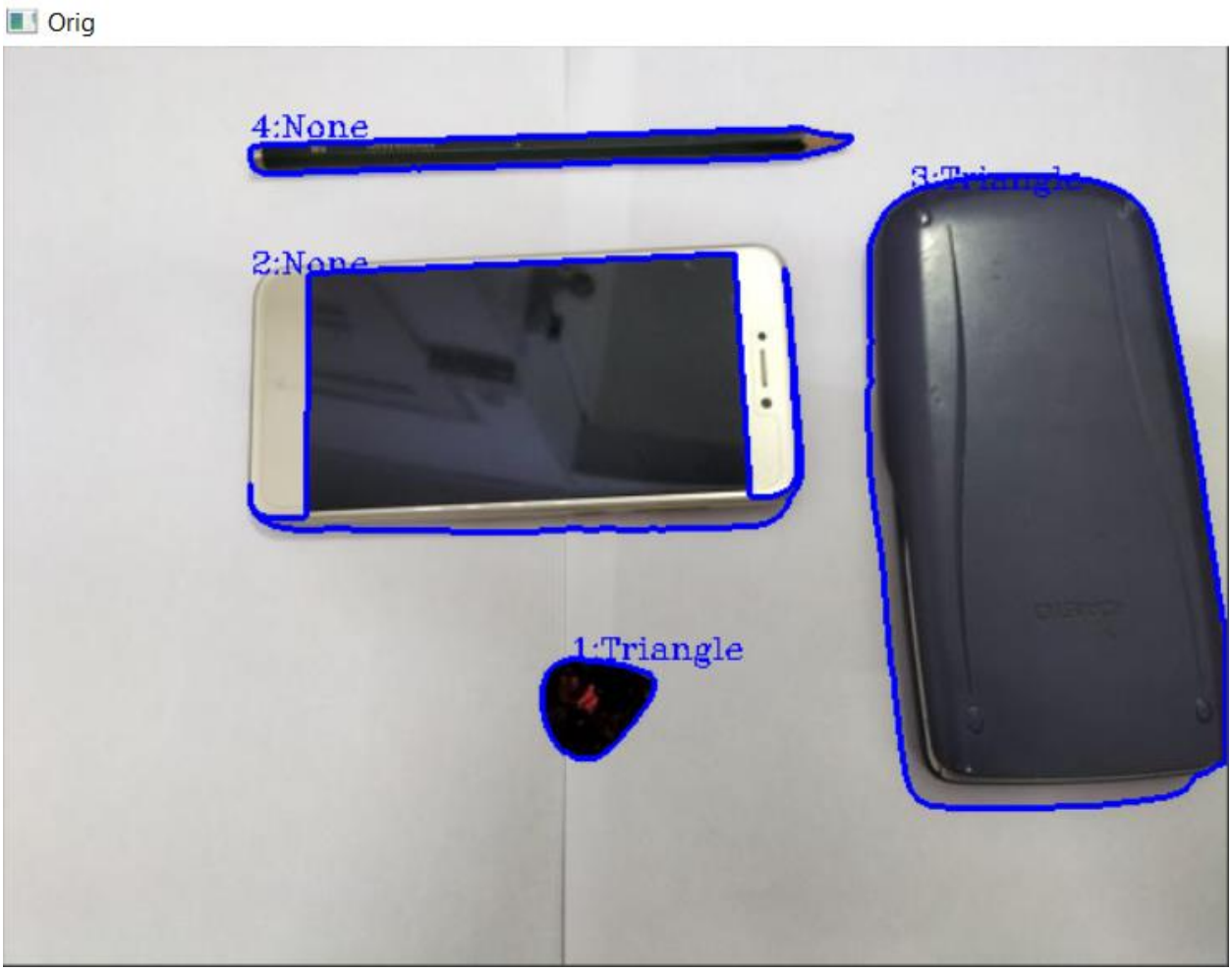

Figure 13: Test image 5 object detection result

```
The number of objects in the scene is:
4

Largest object in the scene is id:  3 Smallest object in the scene is id:  1
```

Figure 14: Test image 5 console result

The image above was used for the testing of the system and the output was semi-accurate as it correctly identify and outline all the objects in the image. Due to the object of index 2 had the color on its sides very close to the color of the background it was not outlined very accurately. The shape of the images were not identified very accurately except for object 1. Object 3 was identified as a triangular object most likely due to one of its edges being cut off in the image.

Test Image 6

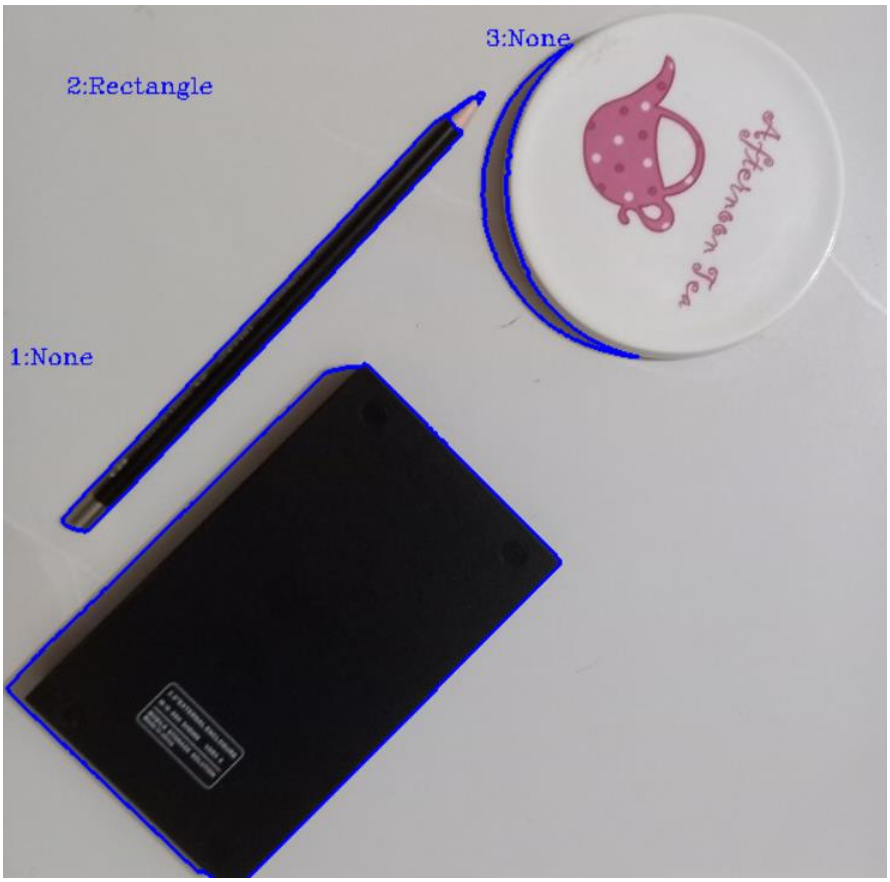

Figure 15: Test image 6 object detection result

```
The number of objects in the scene is:
3

Largest object in the scene is id:  1 Smallest object in the scene is id:  3
```

Figure 16: Test image 6 console result

As seen in Figure 15 and 16, the algorithm is able to detect contours of highly contrasting objects, such as the black objects with white background in this case. Incidentally, it detects shadows as being part of the object, with shadow casted by the white drink coaster being identified as a lunar-shaped contour. Object 1 was not identified as it was deemed to have 6 edges in total, likely in part due to the shadow casted and the object not fitting into the image. Object 3 was deemed to have a total of 10 edges, which stems from the curved shape of the lunar-shaped contour.

5.2 Color detection test

Fruits Image

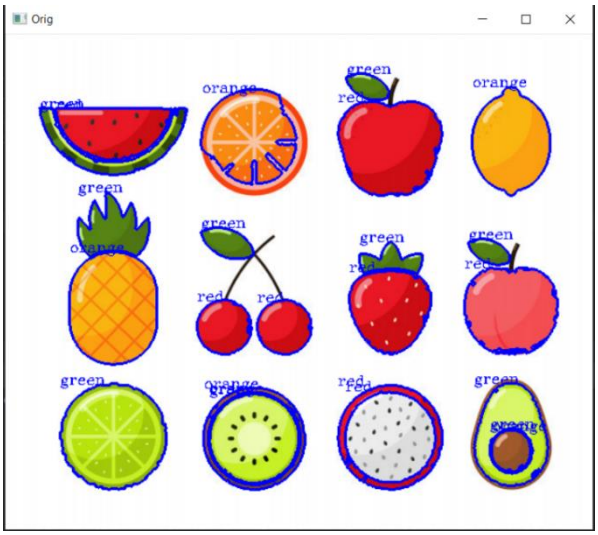

Figure 1717: Fruits image color detection result

For the fruits image, the algorithm works well by highlighting the objects and labeling the colors onto each object. The colors of the objects have high contrast and each color is accurately identified.

16 shapes Image

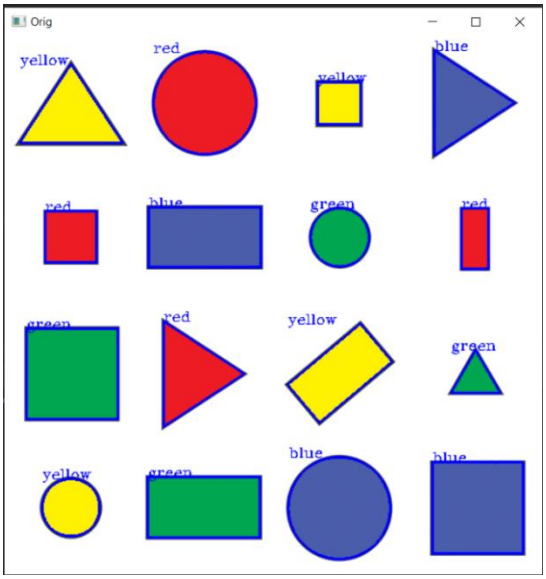

Figure 1818 16 shapes image color detection result

The color detection on the 16 shapes image works well as it has defined objects and contrasting colors.

Test Image 1

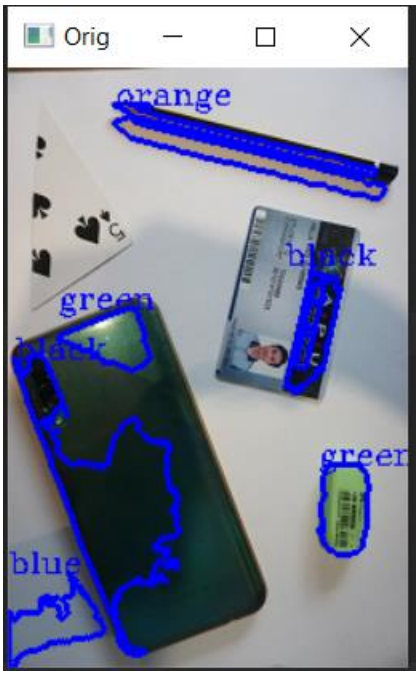

Figure 1919: Test image 1 color detection result

For test image 1, the algorithm was able to detect colors of objects to an extent. Because of the shadow caused by the camera, some parts of the object on the bottom left were labeled as black. The part of the object not covered by the shadow was labeled correctly as green. Previously in the object detection section for this image, the algorithm was not able to detect the bottom right object. In the color detection algorithm, it was able to detect the green color of the object.

Test Image 2

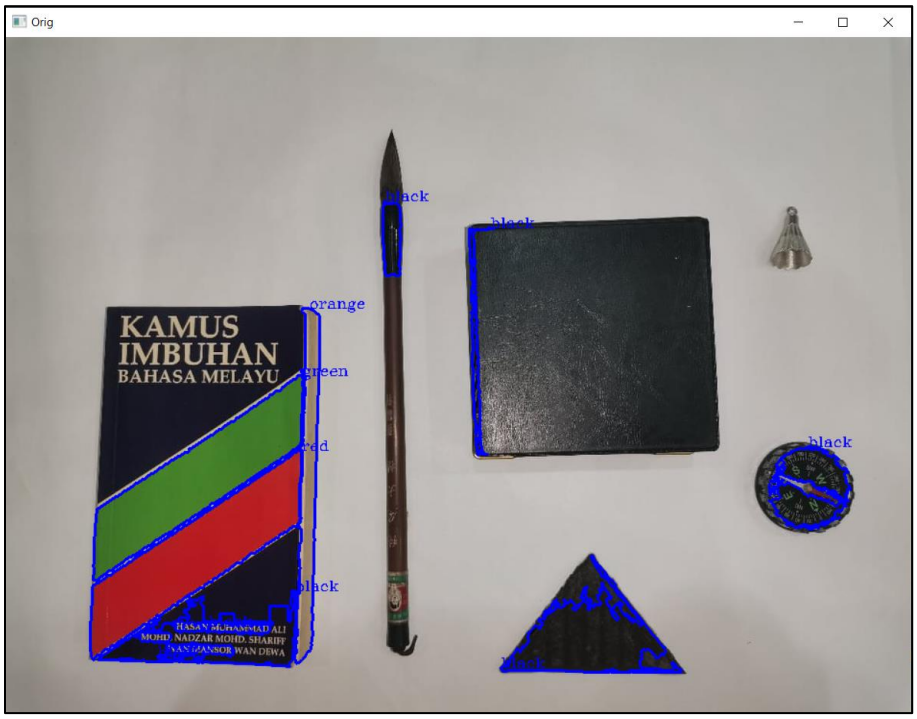

Figure 2020 Test image 2 color detection result

In Figure 20, we could see that the algorithm was able to detect the detail of the book cover, such as the red and green color. However, the right upper object was still unable to be detected as its small size. Most of the black color element was detected as it is obvious compared to the white background.

Test Image 3

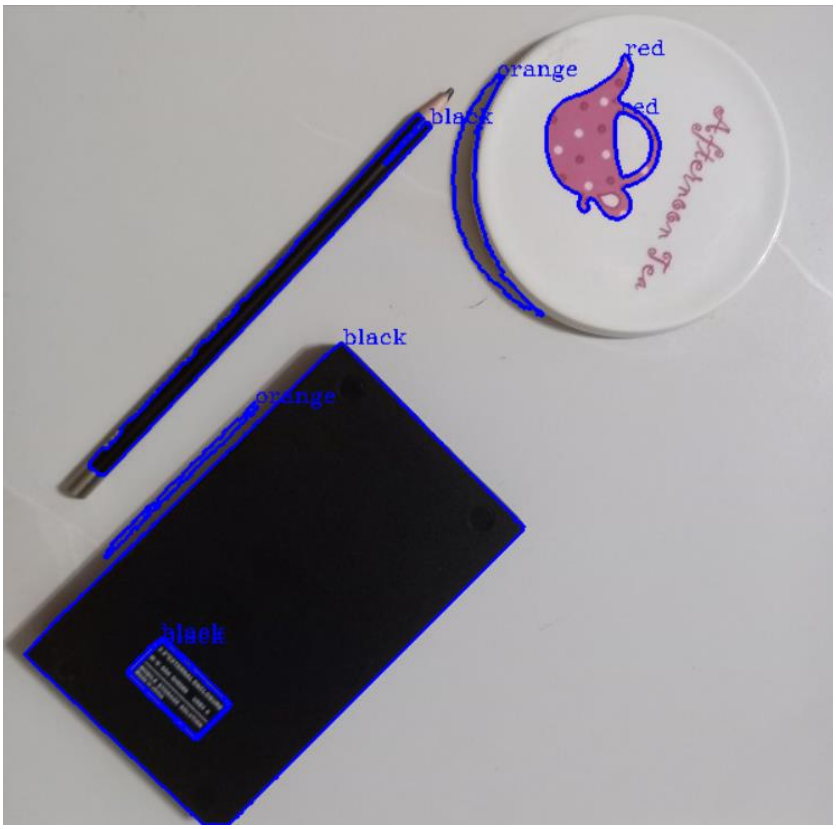

Figure 21 21Test image 3 color detection result

As shown in Figure 21, the algorithm was able to detect black surfaces well against the white background but faltered when it came to shadows and certain contours. Additionally, the algorithm only accounts for basic color detection and not more complex colors such as pink seen in the top right of the image. The orange detected was also non-existent, likely stemming from the gradient of the casted shadow not being completely black in color.

Test Image 6

Figure 2222 Test image 6 color detection

In the above image, we can see that the algorithm is able to detect the color in the image. However, the light reflection in the phone screen causes random drawing of contours around the objects reflected inside the screen. The guitar pick at the bottom of the image was detected to be just black even though the actual color is red due to the color being dark enough to be considered black by the system.

Test image 7

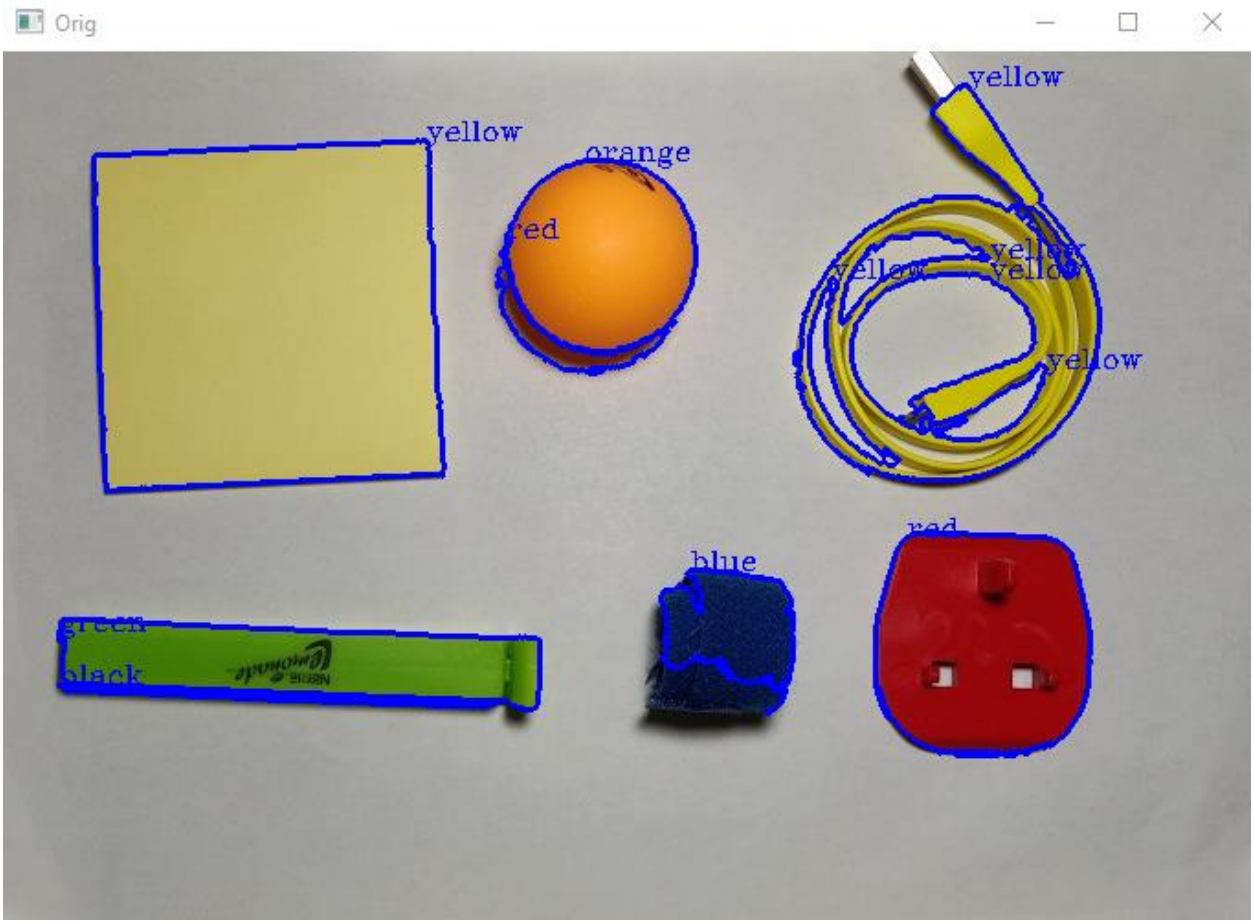

Figure 2323 Test Image 7 color test

For the final test image, an assortment of brightly colored items is placed on an A4 paper. In Figure 23 above, the system was able to perfectly discern the colors of every item displayed, except for some misidentification of the shadows of the objects.

6.0 Critical comments and Analysis

In this section, we review our test results to evaluate the performance of our algorithms. We have tested both features with numerous test images, each with different objects, backgrounds, and characteristics.

When pre-processing real-life test images, there were difficulties in which the patterns and colors of the background would heavily disturb the contouring process, thus affecting the following algorithms as well. For example, in Test image 3, where an assortment of objects is placed on top of a multi-patterned mouse pad, the algorithm detects the background as objects in the process. However, a solution has been proposed to deal with this problem. We can reduce the unwanted detections in the background by adjusting the parameters according to the image's background.

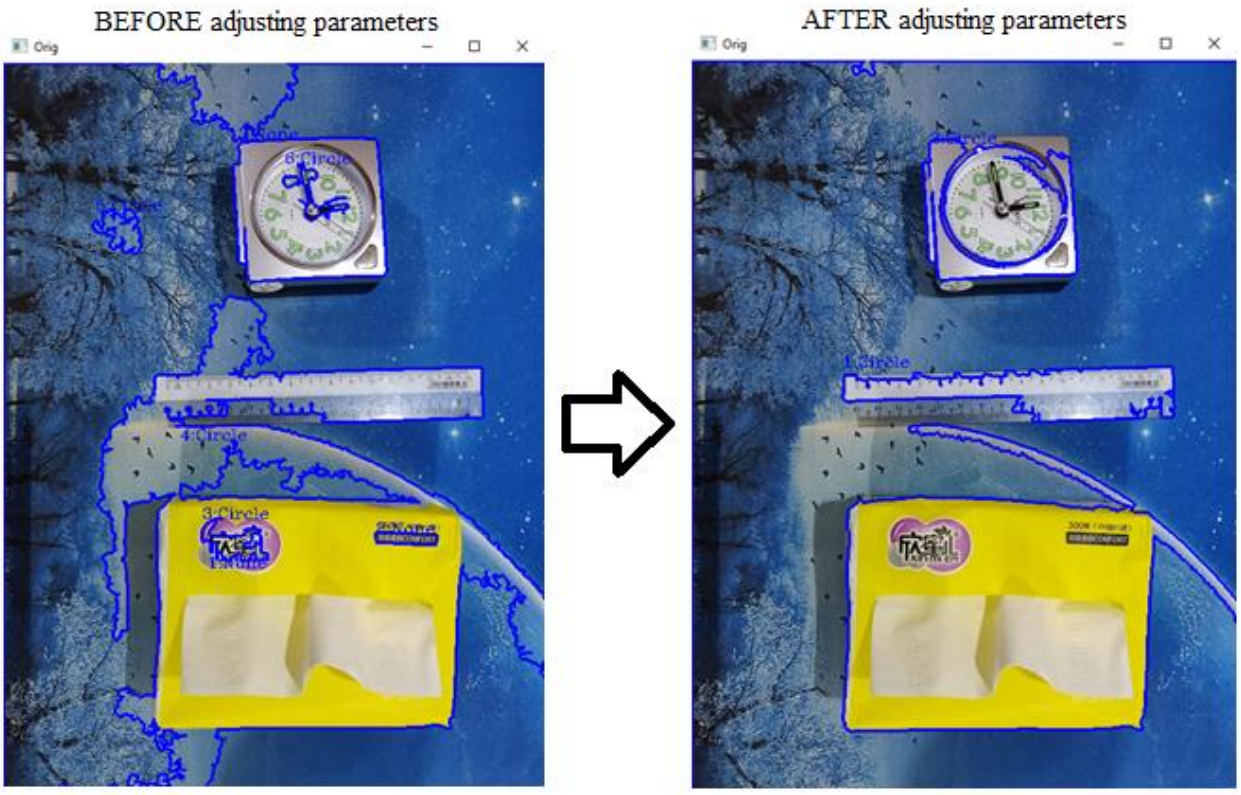

Figure 2424 Test image 3 re-test

This was done by increasing the threshold value, which in turn reduces the detection of finer details. Furthermore, we can also set the minimum area of contoured objects, so if the

objects are too small, it will not have contour lines, this can help remove unwanted tiny detections from the background. But this only works to an extent, as shown in the image above, the program still couldn't not discern the ruler from the background completely, as the ruler has little to no contrast to the background. In addition, images with poor quality, or high noise further decreases the accuracy of the algorithm.

7.0 Conclusion

This research has performed several image processing techniques, such as noise reduction, edge detection, object recognition, size comparison, and shape and color identification. The developed system will accept an image as the input and display the output of recognized object characteristics, including the color and shape of the object. While developing the algorithm, we have implemented findContours() function to identify the object's edge, approxPolyDP() function to determine the polygon's shape, boudingRect() function to find out the area of interest, and other function. The program is completed with the image processing concept obtained during our lectures and some online research sources. In this assignment, our group has shown teamwork and appropriate task allocation in order to complete the work effectively.

8.0 References

Deng, D. (2020). Image Recognition Algorithm Based on Information Fusion Combining Sparsity and Synergy. *2020 2nd International Conference on Information Technology and Computer Application (ITCA)978-1-6654-0378-8/20/$31.00 ©2020 IEEEDOI 10.1109/ITCA52113.2020.00042*. https://doi.org/10.1109/ITCA52113.2020.00042

Fu, K., Jin, J., Cui, R., Sha, F., & Zhang, C. (2017). *Aligning Where to See and What to Tell: Image Captioning with Region-Based Attention and Scene-Specific Contexts*. https://doi.org/10.1109/TPAMI.2016.2642953

Kang, L. W., Chou, K. L., & Fu, R. H. (2019). Deep learning-based weather image recognition. *Proceedings - 2018 International Symposium on Computer, Consumer and Control, IS3C 2018*, 384–387. https://doi.org/10.1109/IS3C.2018.00103

Singha, A. & Bhowmik, M. K. (2018). Object Recognition based on Representative Score Features. *2018 IEEE 18th International Conference on Advanced Learning Technologies* (ICALT), 2018, pp. 419-421, https://doi.org/10.1109/ICALT.2018.00106