Technical Assesment - WIND Project

Kristoforus Bryant Odang

December 5, 2021

1 Introduction

Below are the results for the WIND Project Technical Assesment. The main document is kept at 3 pages, but I include an appendix of all the result. The code implementation of this assesment can be found in `https://github.com/kristoforusbryant/cv-technical-assesment-wind`.

2 Task 1: Circle Detection

2.1 Method Used

For this task, I use the Circles Hough Transform, after blurring and converting the images to grayscale. The Circles Hough Transform does well on this problem. To optimise computation, I resized the image to 1/16 of its size before applying the transformations.

2.2 Results

Note following success cases.

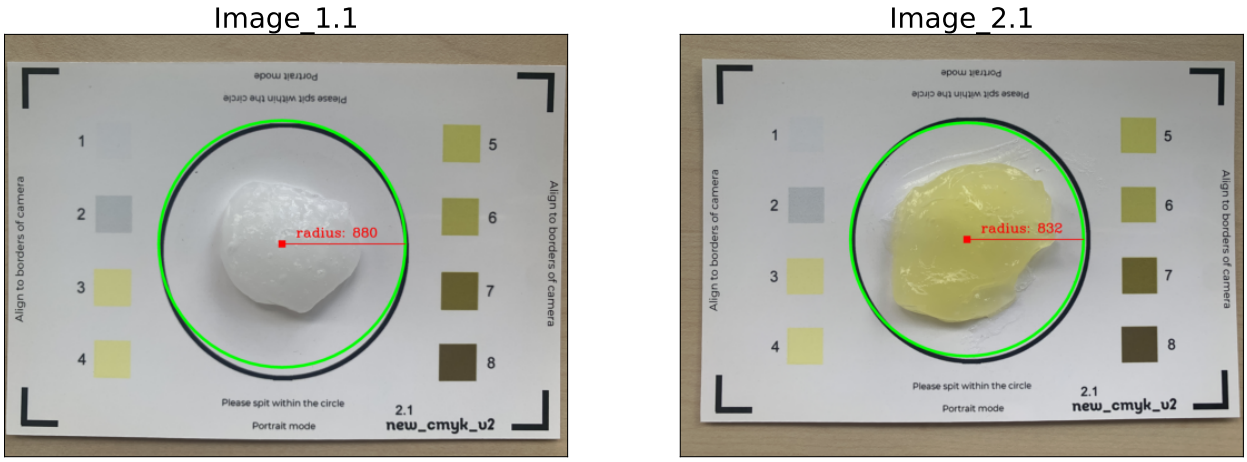

Figure 1: Examples of success cases in Task 1.

No serious failure cases detected.

3 Task 2: Squares Detection

3.1 Method Used

To obtain the boundary around the squares, the method follows three steps. Firstly, noting that the position of the squares remain the same across different images, from a grid (see Fig 7) I slice 8 rectangles, each containing exactly one square. This grid is arbitrarily chosen based on visual inspection of the samples given and thus sensitive to the boundaries of the camera.

For each of these slices, I split the image into its HSV and RGB channels. From testing on some examples (see Fig 8), I notice that for squares 3 - 8, the saturation (S) channel from the HSV decomposition separates the squares from the background most. Whilst, for square 2, the blue (B) channel from the RGB decomposition separates square 2 most from the background. Unfortunately, square 1 is too similar to the background, hence, like square 2 for this I also used the blue channel for further processing.

After getting the S or G channel depening on the squares, I find the contour and draw a bounding box around that contour. This is what is seen in the results.

3.2 Results

The following shows one success case and the most serious failure case from this task.

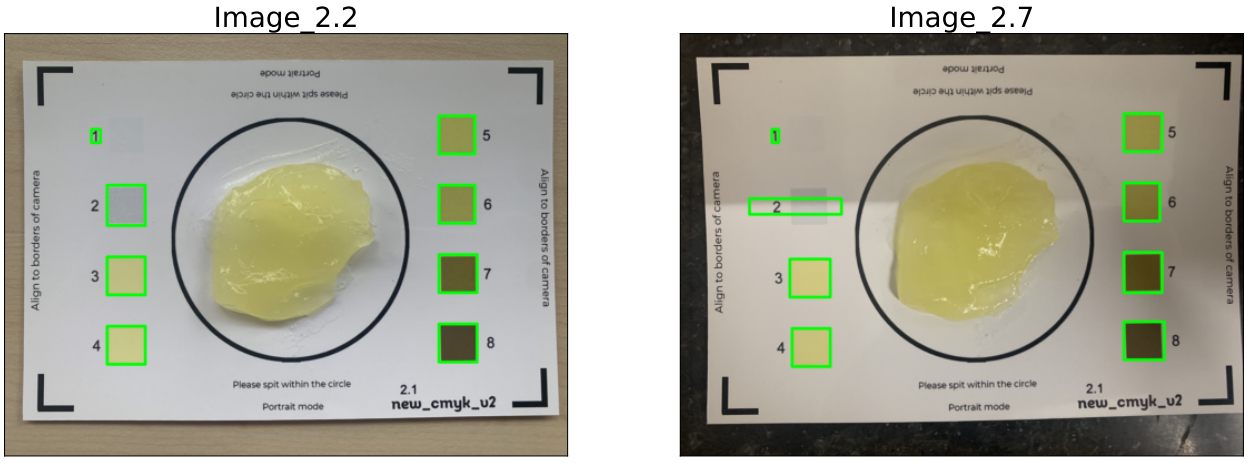

Figure 2: One success and one failure case each from Task 2.

Note that there are two serious failures with the implemented algorihtm. Firstly, the algorithm fails on most cases to detect the square labled 1. This is likely due to the lack of sufficient contrast between the colour of square 1 and the colour of thed background. This difference in colours is even difficult to detect with naked eye. Instead of detecting the square, the algorihtm draws a box around the number 1. This is because the number 1 is the largest detected object on the defined grid. This problem is difficult to solve since the image itself does not give enough contrast for the algorithm to detect. A heuristic that can be used is to take the x and y values from the bottom-left corner of square 3 and square 5, respectively, and draw a square with the width and height of one of the reliable square estimates (3 - 8), taking that point at its bottom-left corner. This can be a solution noting that the position of the squares are invariant relative (to one another) across different pictures.

The second serious failure happens when the picture contains a sharp shadow. For example, in Figure 2, prediction on square 2 is inaccurate as the algorithm detects the shadow as an edge. This can possibly solved using a shadow removal algorithm.

4 Task 3 (and Bonus): Sample Boundaries

4.1 Method Used

To obtain the boundary around the sample. Firstly, I apply the canny edge procedure to the grayscale-transformed image. Then using the method in Task 1, remove edges detected on the perimiter or outside the circle. Then, I find the contours of the remaining edges. The outlines seen around the samples are the convex hull of the union of points of every contour.

4.2 Results

The following shows one success case and the most serious failure case from this task.

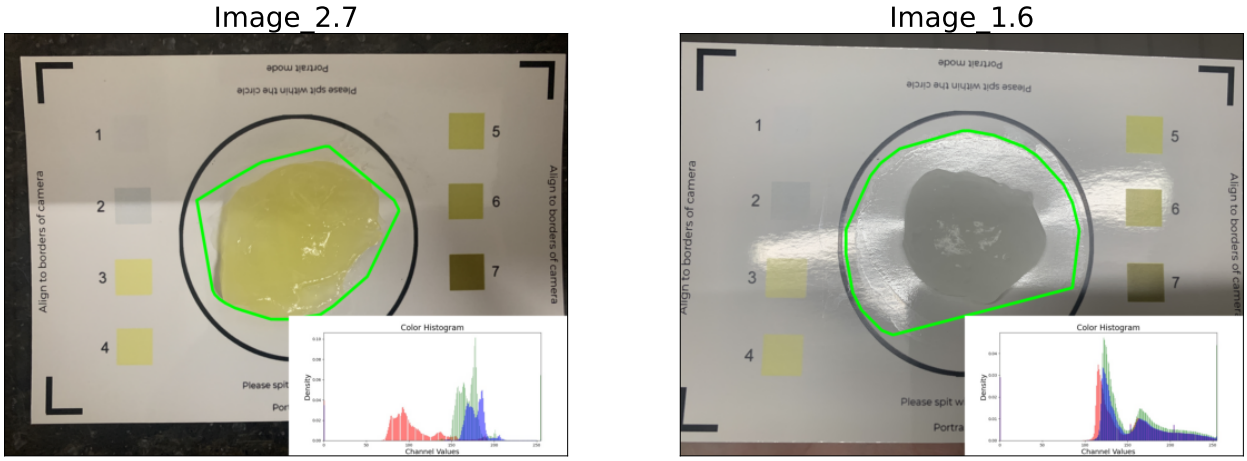

Figure 3: One success and one failure case each from Task 3.

As seen in Figure 3, the reflection of a light source from the background can prevent the algorithm from detecting the sample. This is because the algorithm detects the reflection as a contour, hence the convex hull includes the edges of the reflection. Being able to remove the reflection is necessary for the algorithm to perform in these cases.

5 Discussion

In this document I present the results the tasks given. From the failure cases, I noticed that the shadow and illumination is the primary cause for the failures of my algorithms. Thus, it might be of interest to explore shadow and glare removal methods. Moreover, while the algorihtm in Task 2 does reasonably well for squares 3-8, my algorithm rely being able to find grids that contains one square each. This is currently using a manual threshold, considering the samples given. However, an automated way can be devised to detect these grids, for example by first standardizing the image using methods such as getPerspectiveTransform in openCV.

6 Appendix

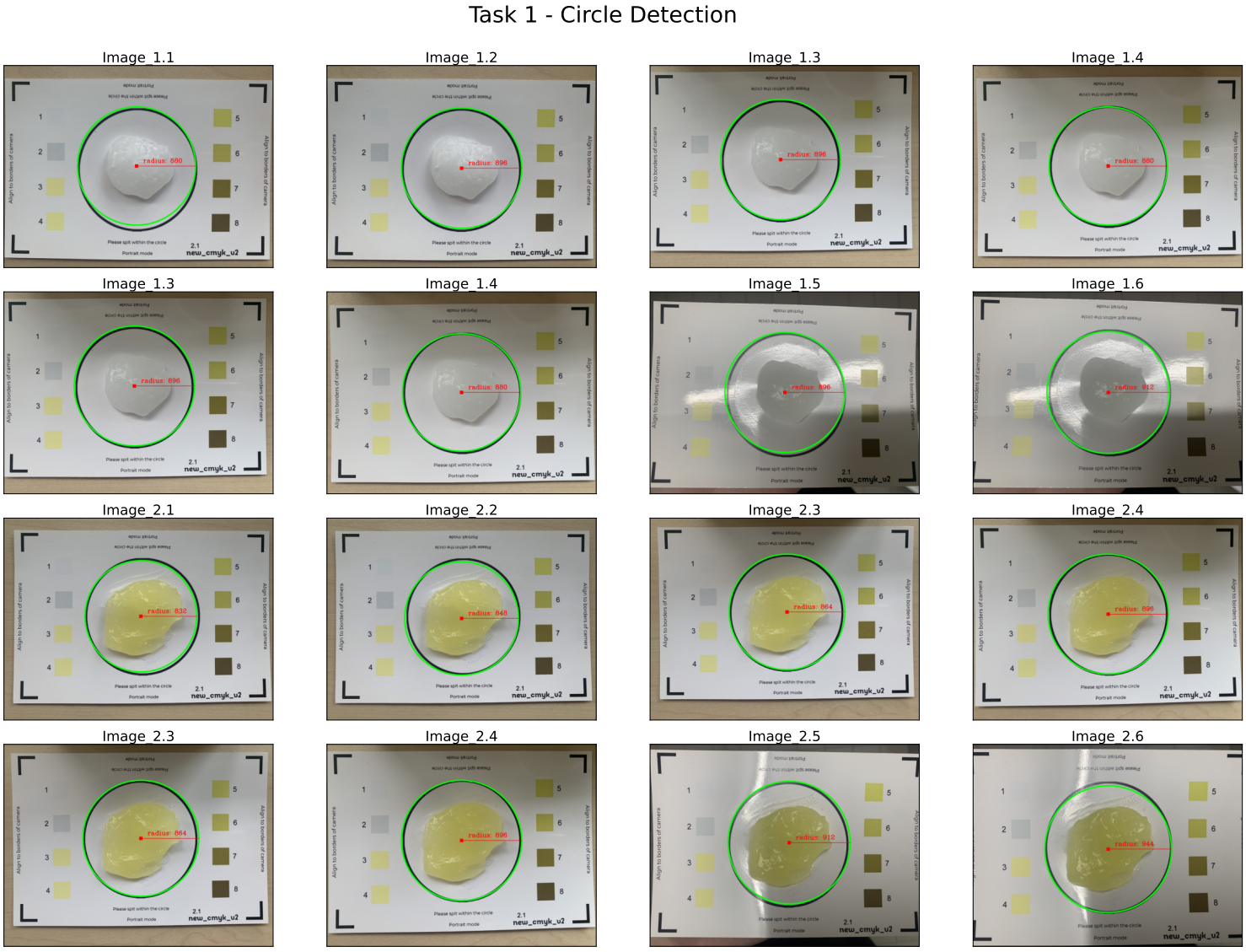

Figure 4: All results from Task 1.

Task 2 - Squares Detection

Figure 5: All results from Task 2.

Task 3 - Sample Boundaries

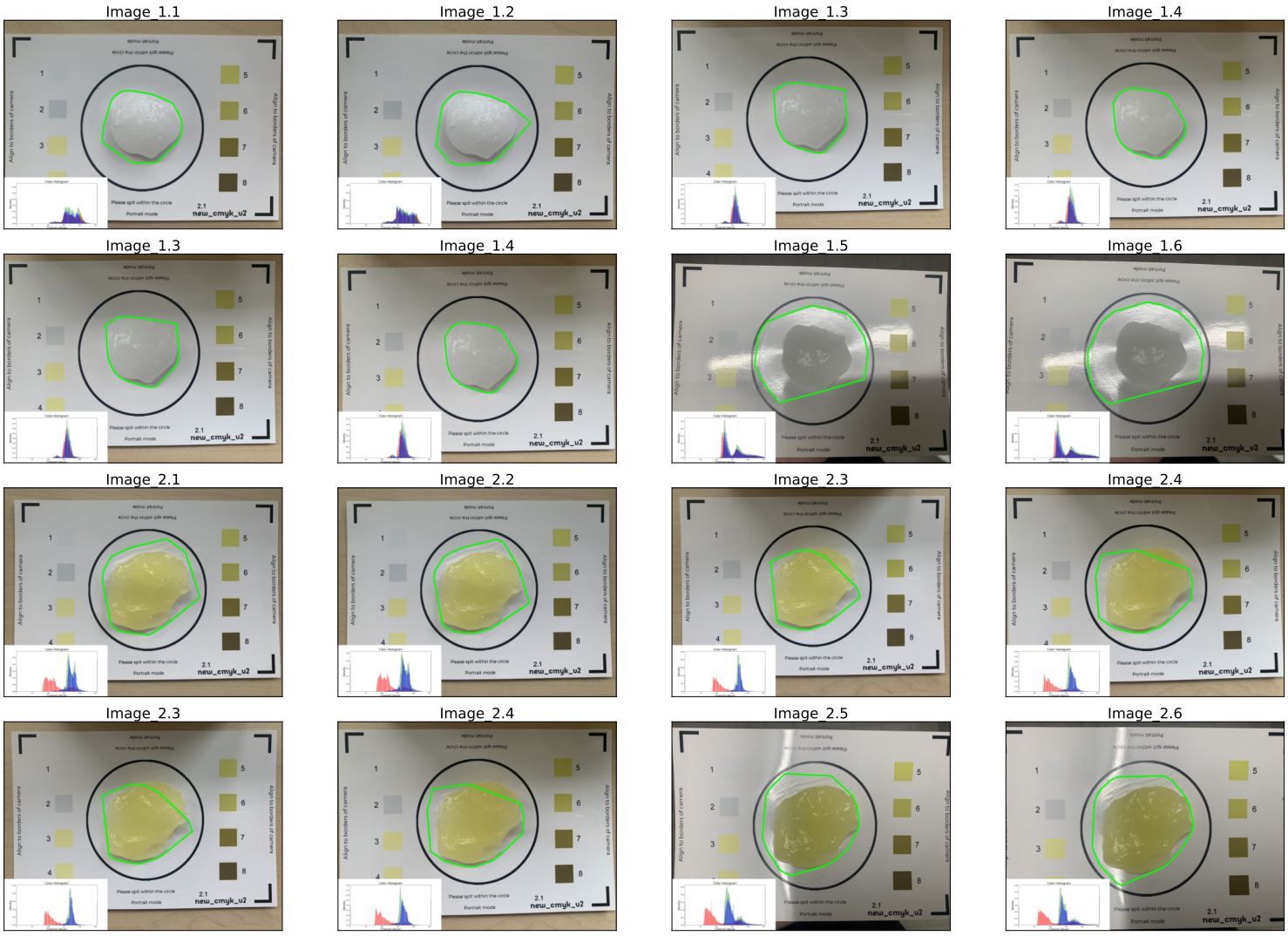

Figure 6: All results from Task 3.

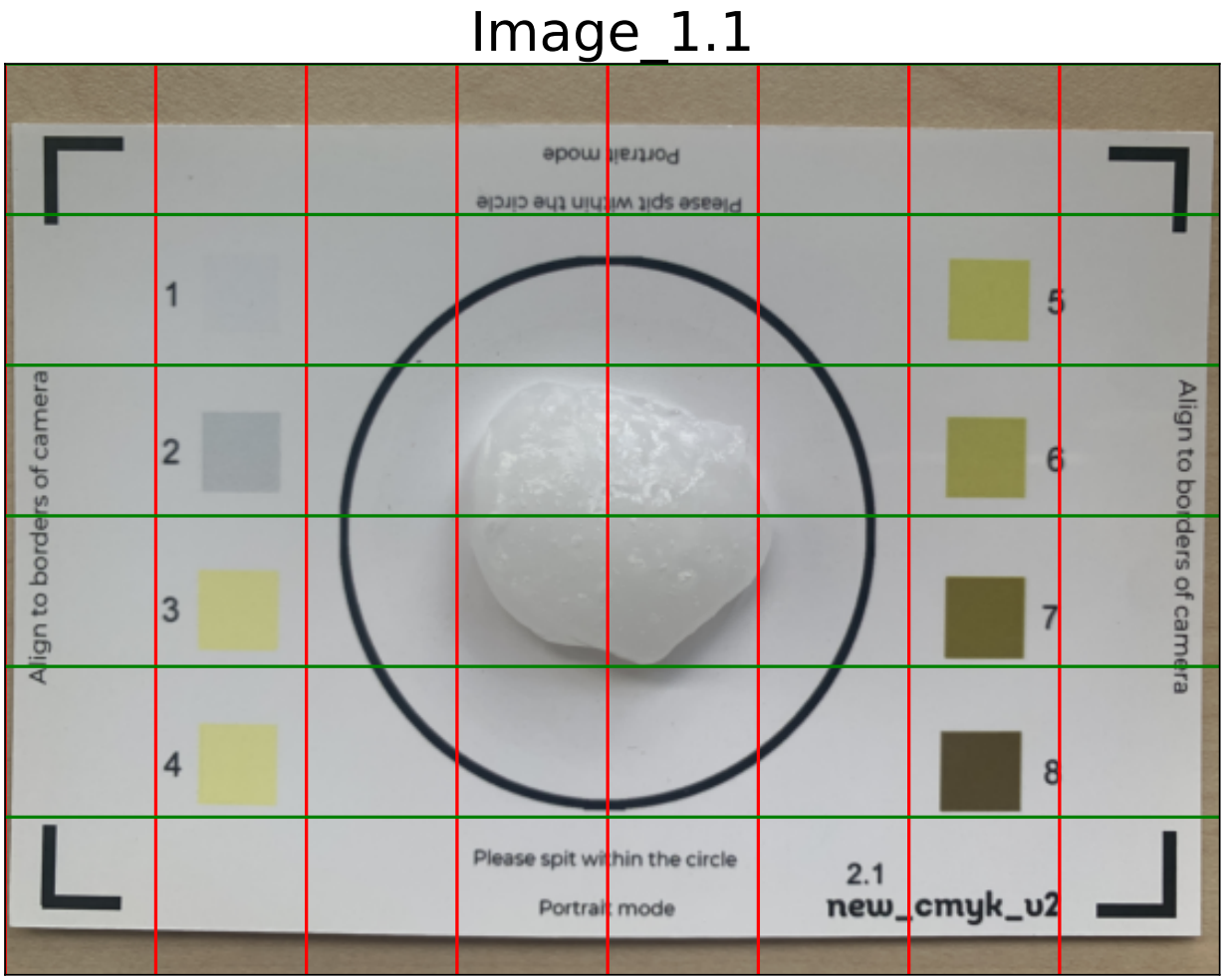

Figure 7: Grid is used to separate the squares, the grid is chose arbitrarily from visual inspection of the samples. This method can be made more robust by standardizing the images' boundaries first.

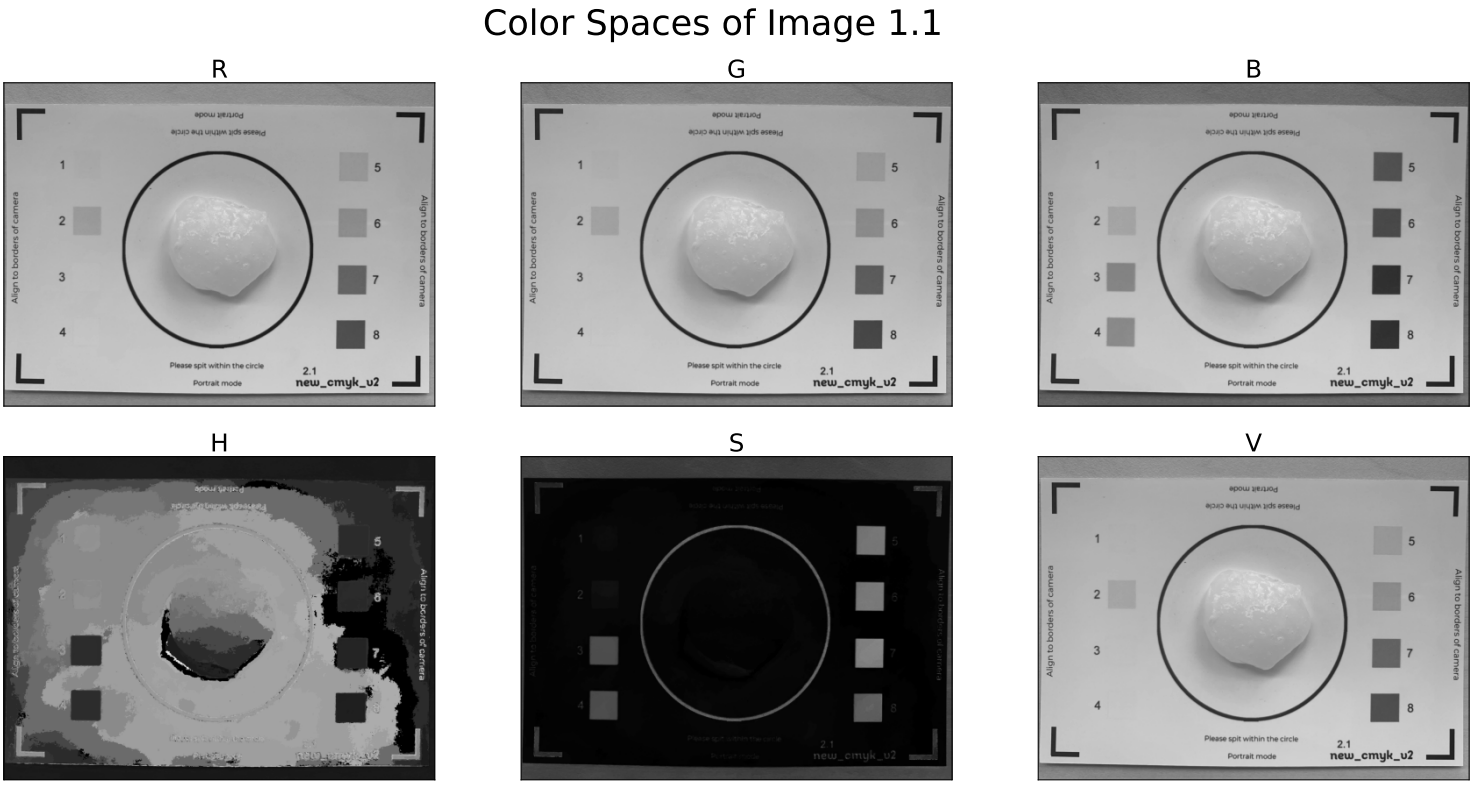

Figure 8: RGB and HSV color spaces of and Image. Notice that for square 2, the blue channel separates the square most from the bacground, while for square 3-8, the S channel separates the square most from the background. No channel clearly separate square 1 with the background.