

521H3 Image Processing Report

Zhengyang Jin
CandNo 250564
March 2022

1 Introduction

This report is for module 521H3 Image Processing assessment. The topic of this project is Lego's Life of George, an image identification game. In this game, players need to memorize the images that appear on the screen and output a colour data matrix without prompting. Each of these images is a square block containing 16 random colour blocks on a white background, and 4 black reference points. In this project, I use the computer to complete tasks on behalf of the player. I designed some functions according to the requirements to complete the task of colour data matrix generation, including the function `findCircles(image)` for finding reference points, and the function for image correction: `correctImage()`. Function `findColors(image)` that identifies colours and converts them to a matrix output of letters representing the colours. I get decent results on our standard test set and also perform well on some extended test sets.

My algorithm has been tested and proven to be able to detect all simulated images and to perform well in some real photos. But 2D transformation still doesn't work well in real photos in most environments, I'll discuss possible improvements in the Discussion section. I provide a full list of the documents provided along with a full table of results in the appendix section.

The actual coding project is based on the python 3.7 language of the Jupyternotebook environment, and the output is also all preserved in the Jupyternotebook file, which can facilitate the inspection of the data results. All functions are stored in the first cell of ipynb. The running code is also given in the same file. Running the corresponding module will get a combination of the original image and the transformed image, and the [4, 4] string array output.

2 Methods

2.1 2D Transformation

I observed that the test set images contain some images that have been rotated, projected, etc. Then I need to restore the image to the correct pose first. So, I applied 2D Transformation to the image, which contains 3 subdivision algorithms, namely keypoint detection, keypoint labelling, and image correction, which will be discussed in detail in the following chapters. Before that, I first performed gray value processing on the picture, because the picture is expressed in the form of RGB three-channel in the computer, and in

the key point detection, we only need to detect the black dots. Therefore, other colour information can be discarded.

2.1.1 Keypoint detection

The main method of this process uses SimpleBlobDetector provided by OpenCV. It first converts the source image to binary images by applying thresholding with distance thresholdStep between neighboring thresholds. It then extracts connected components from each binary image by findContours and computes their centers. Finally, the centers of multiple binary images are grouped by coordinates, the final center of the blob and its radius are estimated from these groups, and returned as the location and size of the keypoints. This function has 5 parameter conditions, namely colour Threshold, Area, Circularity, Inertia, Convexity. Among them, the value of Circularity is between 0 and 1, the closer to 1, the more circular the detected shape, and 0 is a triangle. The value of Inertia determines the roundness of the detected blob, and the closer it is to 0, the greater the difference between the size and diameter of the ellipse. Convexity, on the other hand, describes how complete a circle is.

2.1.2 Keypoint Labelling and Image correction

The image correction method adopts the warpPerspective method of OpenCV. The principle of this method is very simple. By inputting the coordinates of the four key points of the image to be corrected and the coordinates of the key points of the reference image, the perspective transform matrix is generated, and then the image is transformed. But there is a problem, that is, the method needs to determine the coordinate order of key points, and the arrangement of 4 points in the matrix must be clockwise or counter clockwise. I solved this problem through the function Keypoint Labelling designed by myself, see figure 2.1.2. My algorithm takes into account the situation of rotated or projected images, and adopts the method of angle analysis. First, find the blob closest to a vertex as point 1, and then calculate the point with the smallest angle between the vertex and the adjacent vertex. Then these two

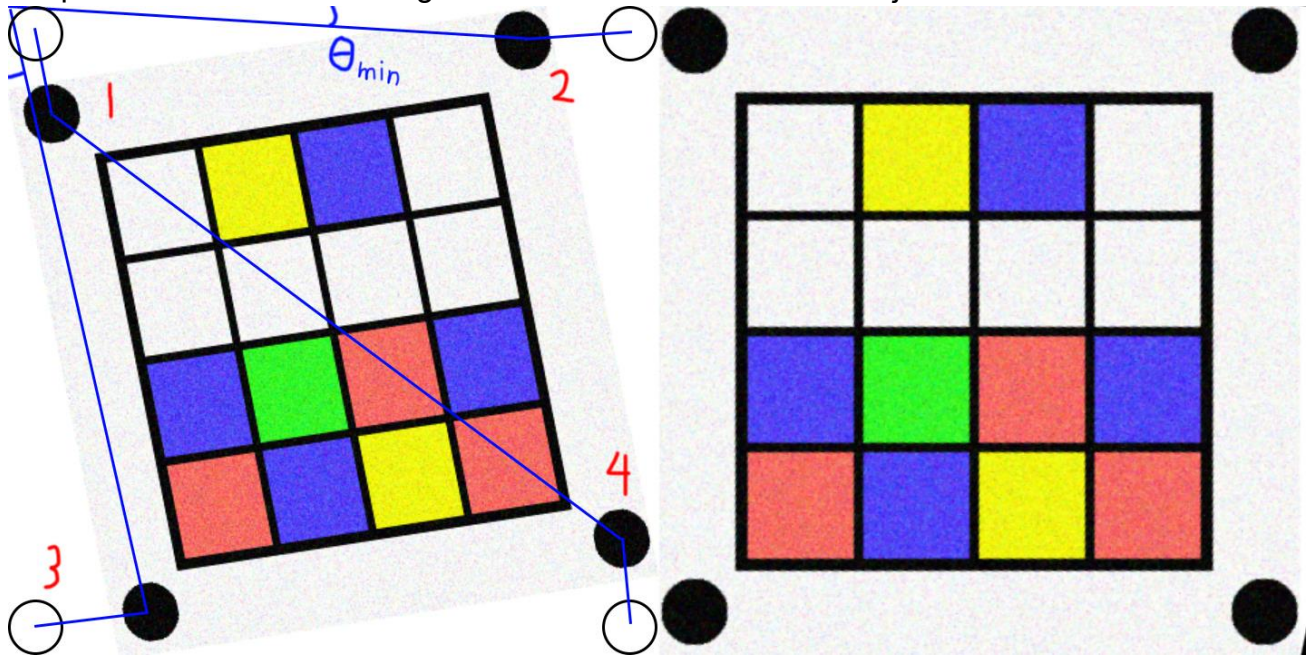


Figure 2.1.2 Keypoint Labelling and Image correction example

points are points 2 and 4. Finally, the point with the farthest distance is calculated as point 3.

2.2 Colour detection

Since we know the locations of the four keypoints and have image-corrected them. Then all the current test sets can be displayed as in figure2. Then I use opencv's boundingRect method to find the rectangle in the image. Since the largest rectangle is our image block, we can directly crop the rest. So now we only have the central colour block area, and this area has 16 colour areas of the same area, so we can directly clip it averagely, and then take a separate colour measurement for each independent colour block. I first modified the colour block to HSV format, and then measured the colour area where each colour is located for classification and labeling (see Figure 2.2 for colour division). I have found this method of classifying colours to be quite robust, but much simpler than with Lab colour space; which requires 2 histograms, 2D bounding boxes, or some other more complicated methods. In particular, we have found disambiguating red-yellow-green to be much easier with this

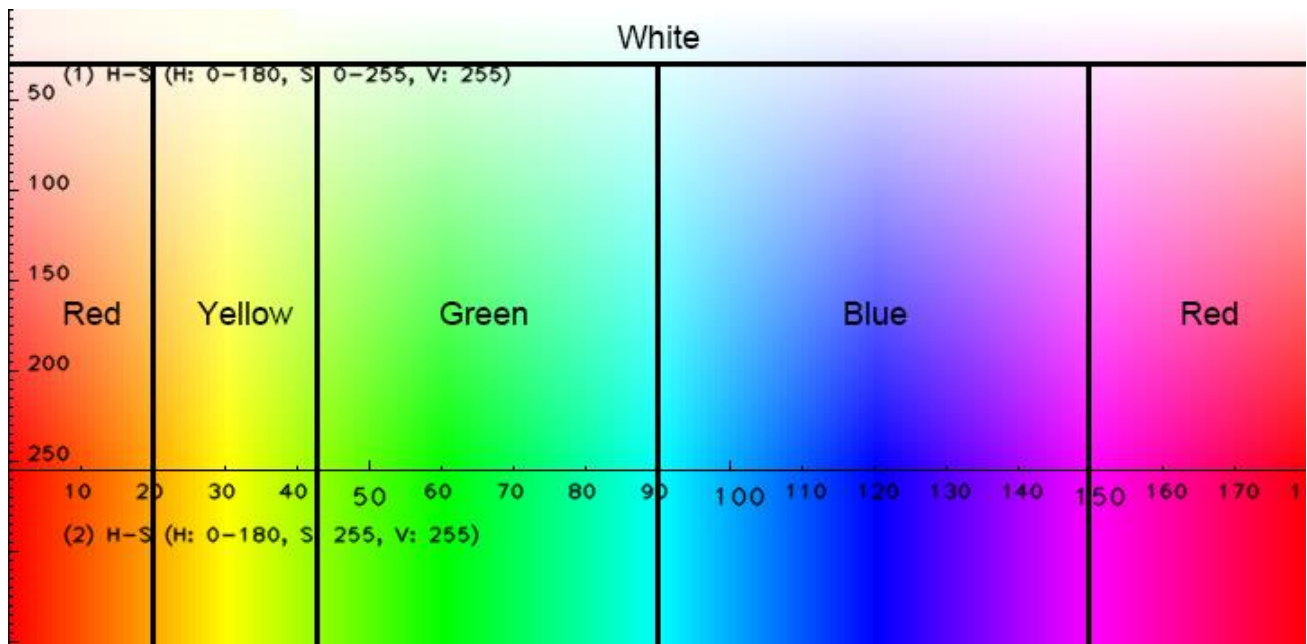
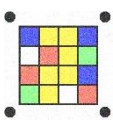
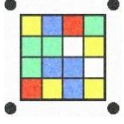
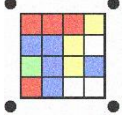
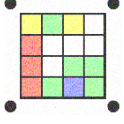
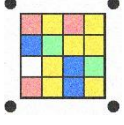
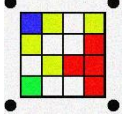
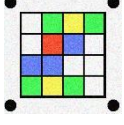
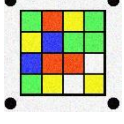
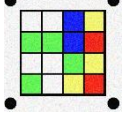
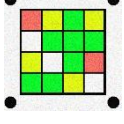







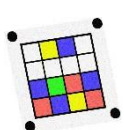
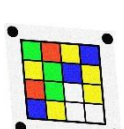
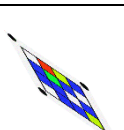
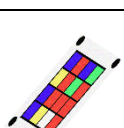
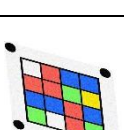
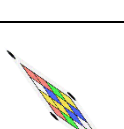
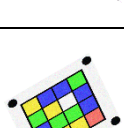
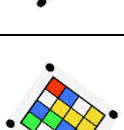
Figure 2.1.2 HSV color split

method, than in Lab space. I found that most of the colour blocks are mixed with many noise points, then we cannot calculate the average value of the area, it may be affected by the value of the noise points. So, I took the colour value with the most occurrences as output. This method can effectively filter out the influence of noise points.

3 Results

| Filename | Image | Output | Success | Notes |
|-------------|---|--|---------|-------|
| noise_1.png |  | [['b' 'y' 'y' 'b'] ['w' 'r' 'y' 'g'] ['r' 'y' 'y' 'b'] ['g' 'y' 'w' 'r']] | Yes | None |

| | | | | |
|-------------|---|--|-----|------|
| noise_2.png |  | [['y' 'b' 'r' 'g'] ['g' 'g' 'w' 'y'] ['g' 'b' 'b' 'w'] ['r' 'y' 'b' 'y']] | Yes | None |
| noise_3.png |  | [['r' 'r' 'r' 'y'] ['b' 'b' 'y' 'w'] ['g' 'b' 'y' 'b'] ['r' 'b' 'w' 'w']] | Yes | None |
| noise_4.png |  | [['y' 'g' 'y' 'w'] ['r' 'w' 'w' 'w'] ['r' 'w' 'g' 'g'] ['r' 'g' 'b' 'g']] | Yes | None |
| noise_5.png |  | [['r' 'y' 'r' 'y'] ['b' 'g' 'y' 'y'] ['w' 'y' 'b' 'g'] ['r' 'y' 'b' 'y']] | Yes | None |
| org_1.png |  | [['b' 'y' 'w' 'y'] ['y' 'w' 'w' 'r'] ['w' 'y' 'r' 'r'] ['g' 'w' 'w' 'r']] | Yes | None |
| org_2.png |  | [['w' 'g' 'y' 'g'] ['w' 'r' 'b' 'w'] ['b' 'b' 'w' 'w'] ['g' 'y' 'g' 'w']] | Yes | None |
| org_3.png |  | [['g' 'r' 'y' 'g'] ['y' 'b' 'g' 'g'] ['b' 'r' 'r' 'w'] ['g' 'y' 'w' 'y']] | Yes | None |
| org_4.png |  | [['w' 'w' 'b' 'y'] ['g' 'g' 'b' 'r'] ['w' 'w' 'g' 'y'] ['g' 'w' 'y' 'r']] | Yes | None |
| org_5.png |  | [['r' 'y' 'g' 'y'] ['w' 'g' 'g' 'g'] ['y' 'w' 'g' 'r'] ['g' 'g' 'y' 'w']] | Yes | None |
| proj_1.png |  | [['w' 'g' 'y' 'y'] ['w' 'g' 'y' 'g'] ['b' 'y' 'y' 'r'] ['y' 'w' 'b' 'y']] | Yes | None |
| proj_2.png |  | [['y' 'y' 'y' 'y'] ['y' 'w' 'r' 'y'] ['r' 'g' 'w' 'b'] ['r' 'w' 'r' 'g']] | Yes | None |

| | | | | |
|-------------|---|--|-----|------|
| proj_3.png |  | [['b' 'y' 'y' 'b'] ['w' 'r' 'g' 'y'] ['g' 'r' 'b' 'r'] ['w' 'y' 'g' 'b']] | Yes | None |
| proj_4.png |  | [['b' 'r' 'y' 'y'] ['r' 'g' 'b' 'r'] ['b' 'y' 'y' 'b'] ['g' 'g' 'g' 'y']] | Yes | None |
| proj_5.png |  | [['y' 'g' 'y' 'r'] ['y' 'r' 'w' 'b'] ['b' 'g' 'b' 'b'] ['g' 'b' 'g' 'r']] | Yes | None |
| proj1_1.png |  | [['w' 'y' 'b' 'w'] ['w' 'w' 'w' 'w'] ['b' 'g' 'r' 'b'] ['r' 'b' 'y' 'r']] | Yes | None |
| proj1_2.png |  | [['g' 'r' 'y' 'b'] ['y' 'g' 'b' 'y'] ['r' 'g' 'w' 'y'] ['b' 'y' 'w' 'w']] | Yes | None |
| proj1_3.png |  | [['w' 'r' 'y' 'b'] ['b' 'g' 'b' 'w'] ['b' 'b' 'w' 'b'] ['w' 'b' 'b' 'y']] | Yes | None |
| proj1_4.png |  | [['b' 'r' 'b' 'g'] ['y' 'r' 'g' 'r'] ['b' 'r' 'r' 'r'] ['b' 'y' 'w' 'r']] | Yes | None |
| proj1_5.png |  | [['w' 'r' 'b' 'b'] ['r' 'r' 'g' 'y'] ['b' 'b' 'r' 'b'] ['g' 'w' 'r' 'g']] | Yes | None |
| proj2_1.png |  | [['r' 'r' 'g' 'y'] ['w' 'b' 'y' 'b'] ['y' 'y' 'b' 'w'] ['r' 'g' 'w' 'g']] | Yes | None |
| proj2_2.png |  | [['y' 'b' 'b' 'g'] ['g' 'y' 'w' 'b'] ['b' 'g' 'g' 'b'] ['b' 'y' 'y' 'r']] | Yes | None |
| proj2_3.png |  | [['r' 'b' 'w' 'y'] ['w' 'y' 'y' 'y'] ['b' 'g' 'b' 'g'] ['g' 'w' 'r' 'w']] | Yes | None |

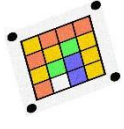
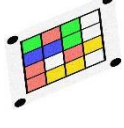
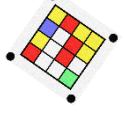
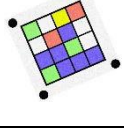
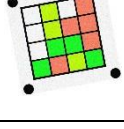
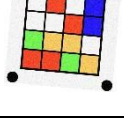
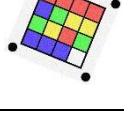
| | | | | |
|-------------|---|--|-----|------|
| proj2_4.png |  | [['y' 'r' 'r' 'r'] ['r' 'y' 'g' 'y'] ['y' 'g' 'b' 'y'] ['r' 'w' 'b' 'y']] | Yes | None |
| proj2_5.png |  | [['g' 'r' 'g' 'w'] ['b' 'b' 'g' 'w'] ['r' 'w' 'r' 'y'] ['r' 'y' 'y' 'w']] | Yes | None |
| rot_1.png |  | [['y' 'r' 'y' 'y'] ['b' 'w' 'r' 'y'] ['y' 'w' 'r' 'w'] ['r' 'w' 'w' 'g']] | Yes | None |
| rot_2.png |  | [['g' 'w' 'y' 'r'] ['w' 'r' 'b' 'w'] ['b' 'b' 'g' 'w'] ['g' 'b' 'b' 'b']] | Yes | None |
| rot_3.png |  | [['w' 'y' 'w' 'r'] ['w' 'y' 'r' 'r'] ['w' 'g' 'g' 'r'] ['g' 'r' 'y' 'g']] | Yes | None |
| rot_4.png |  | [['w' 'r' 'w' 'b'] ['w' 'w' 'r' 'b'] ['g' 'y' 'y' 'w'] ['r' 'r' 'g' 'y']] | Yes | None |
| rot_5.png |  | [['r' 'r' 'r' 'r'] ['b' 'g' 'y' 'y'] ['g' 'y' 'r' 'g'] ['b' 'b' 'b' 'w']] | Yes | None |

Table 1 Results Table

The final output is shown in Table 1. The final result can successfully detect all the test set images, and the accuracy is as high as 100%. This is a test result that does not take into account the actual situation. Since the format of the simulated pictures is regular, and will not be affected by light, distortion, colour block interference, false key point interference and other phenomena. It can be seen that my algorithm is not affected by operations such as blurring, noise, whitening, inversion, transformation, etc.

4 Discussion

After adding real photos as a test, the results of our algorithm seem to be less than satisfactory. Correct image segmentation cannot be performed on test images 0034, 0035, 0041, 0042, and 0044. The characteristics of these pictures are that they are illuminated by strong light, blocked by shadows, the paper is distorted, illuminated by blue light, and out of focus. It can be seen that in these cases, the key point detection will be affected by the

environment, and the identification will be wrong, or it will not be identified. And in image No 0032,0033,0036,0037,0038 the performance seems to be good. Especially for image 0038, the results can be found in Figure 4.1. We can observe that this image has a lot of noise, contains some colour blocks, some text, and black blobs drawn with a marker.

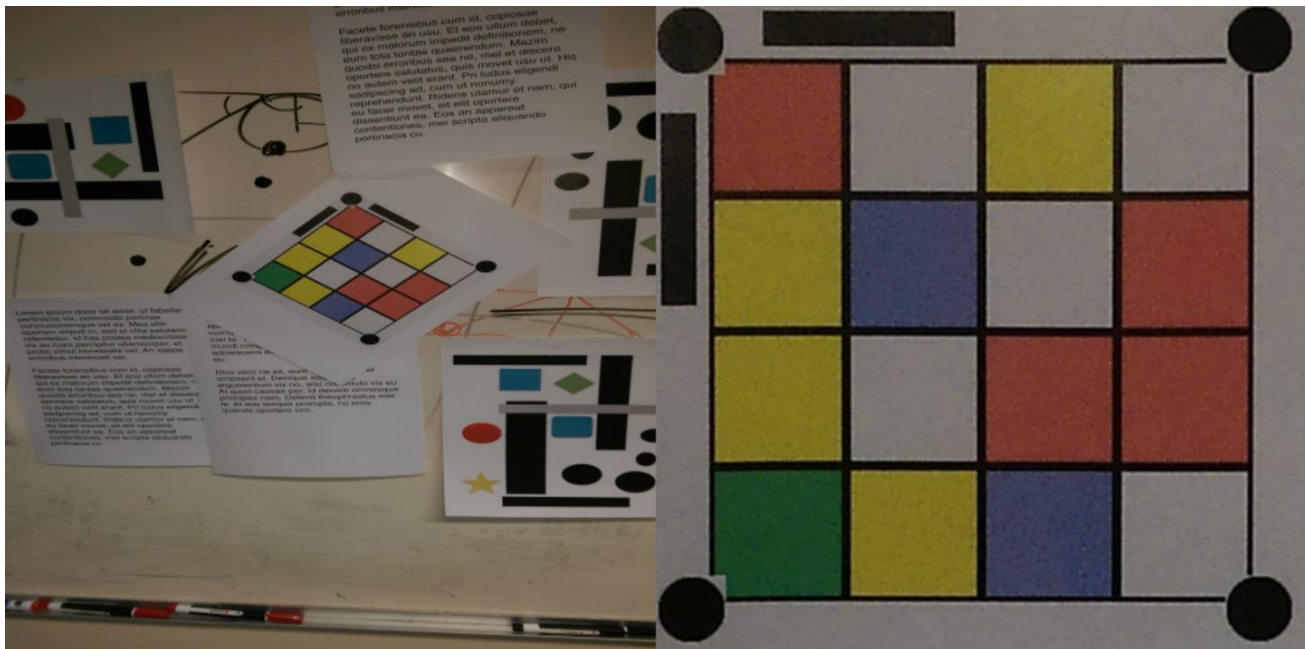


Figure 4.1 Test Result on IMAG0038.jpg

But the algorithm can still correctly identify the key points. So this shows that our algorithm does not have the ability to fine-tune colour, it is easily affected by uneven lighting. I would consider adding a method of dynamic identification of balance in future research, where the algorithm can adjust the threshold settings, analyze each part of the uneven light, and finally summarize the results. This may be able to address the effects of light. But for unfocused pictures, this method is not sure whether it will work.

In summary, our algorithm is able to achieve perfect recognition results in all simulated images, but only works well in some real photos. But this is enough to show that the algorithm is robust and sufficient for detection in most cases. In particular, the performance on the real picture 0038 shows that the algorithm has a very high accuracy in blob shape detection and a strong ability to filter noise.