Assignment 1

Wang Jinghan 23099359d

The code for the assignment is available on github: <https://github.com/sebastianvne/COMP4432-Assignment1>

In your report, these questions should be answered:

1. How do you design and test the program?
2. How do you ensure the robustness of the program in a real scenario?
3. What problems do you find and how do you solve them?
4. How do you use GenAI to assist you in the implementation?
5. How does GenAI understand and solve th e tasks?
6. What are the limitations and areas for improvement in GenAI’s solution?
7. Introduction

The assignment is meant for generating Lego style image from original image. The assignment simply mapped the original rgb vector to few discrete color vectors, determined by configurations by calculating the euclidean distance.

1. Design

module 1:

capturing the image using computer camera, which is embedded in opencv library.

module 2:

Task2 requires us to map the original image to a image with maximum 100 \* 100 bricks, each brick is of size 1\*1.

Hence, I designed the data flow as following: firstly determine the target size of downsampling, the function would calculate the long side and its size, then resize the long side to maximum of 100 bricks, while keeping the ratio of image long side size and short side size.

Secondly, calculating the euclidean distance of each original image pixels with the specific target mapping colors, then finding the color with the minimum distance and generate the Lego pixels.

Finally, resize the generated pixels to original size, and output the image.

module 3:

Task3 requires to use more colors and more size of bricks to generate the Lego style image. Based on task2, I implemented a greedy brick merge. The code is as follows, for each original 1\*1 bricks, the algorithm would iteratively use different size of bricks to check if using the specific brick would cover other bricks that originally not the same color as the initial 1\*1 brick.

Also, the original bricks is different from the implementation of task2, task3 would map the original image with more color vectors.

module 4:

For task4, I implemented a live mode that can capture the image from computer camera and transform it into Lego style in live mode.

1. Test

The test would only indicates the tests that conducted by executing python3 main.py in the repository, with suffix in test command part.

1. Task 1:
   1. Test if window of opencv can show the correct image captured from computer camera. Window showed as below. Run python3 main.py --mode capture



* 1. Test if Press q can be used to exit and press s can be used to save the image, and check if the image saved is correct. As shown in my github repository /outputs/test1-2.jpg, the image is correct.



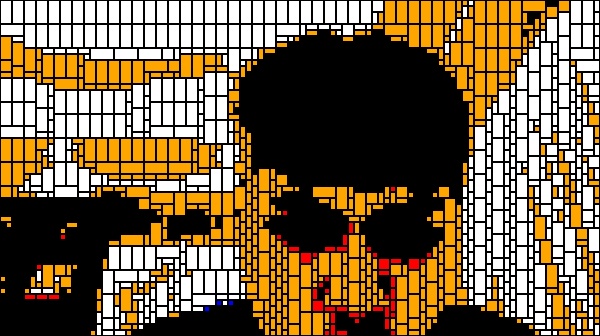
1. Task 2:

I used the image test1-2.jpg to test the output of task2. Running python3 main.py --mode task2 --input ./outputs/test1-2.jpg. The output is shown in the repository, within the outputs folder, test2-1.jpg shows the output of task2, the image uses black, white and gray to render the image.



1. Task 3:

Run command python3 main.py --mode task3 --input ./outputs/test1-2.jpg, the image will show in the output folder, brick\_summary\_3-1.txt shows the bricks summary, and test3-1.jpg is the output of task3



brick\_summary\_3-1.txt:

Total bricks: 1520

2x4: 426

2x2: 95

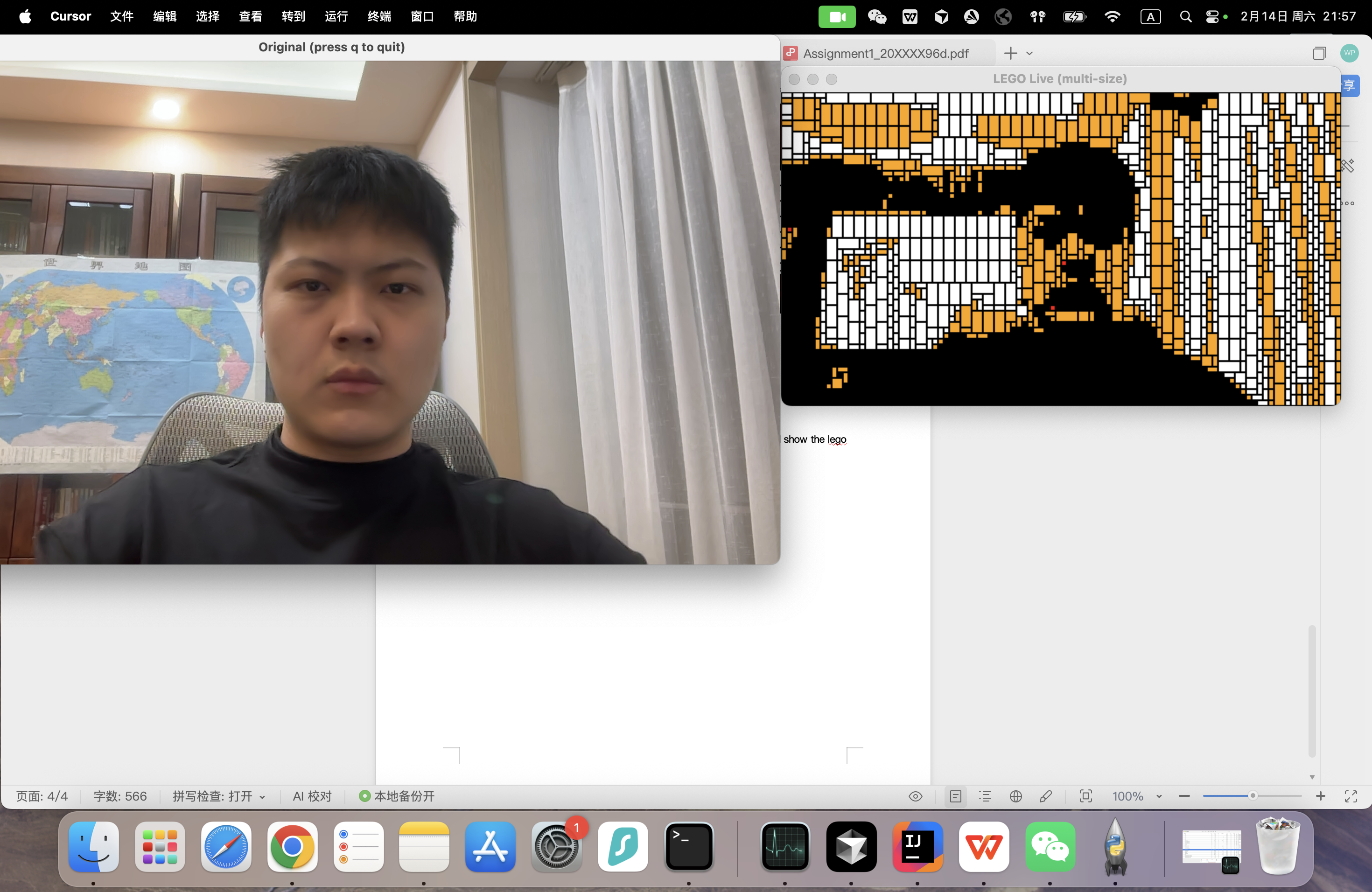
1x2: 813

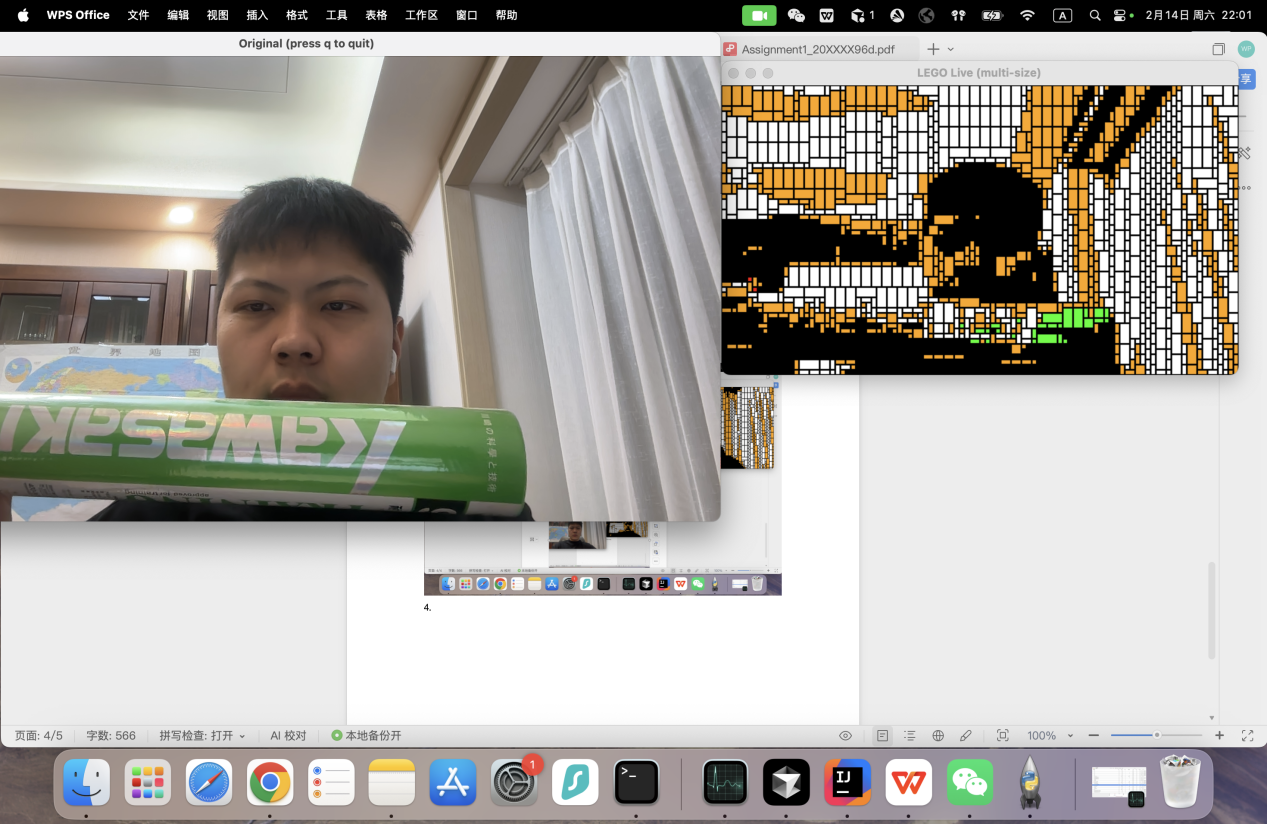
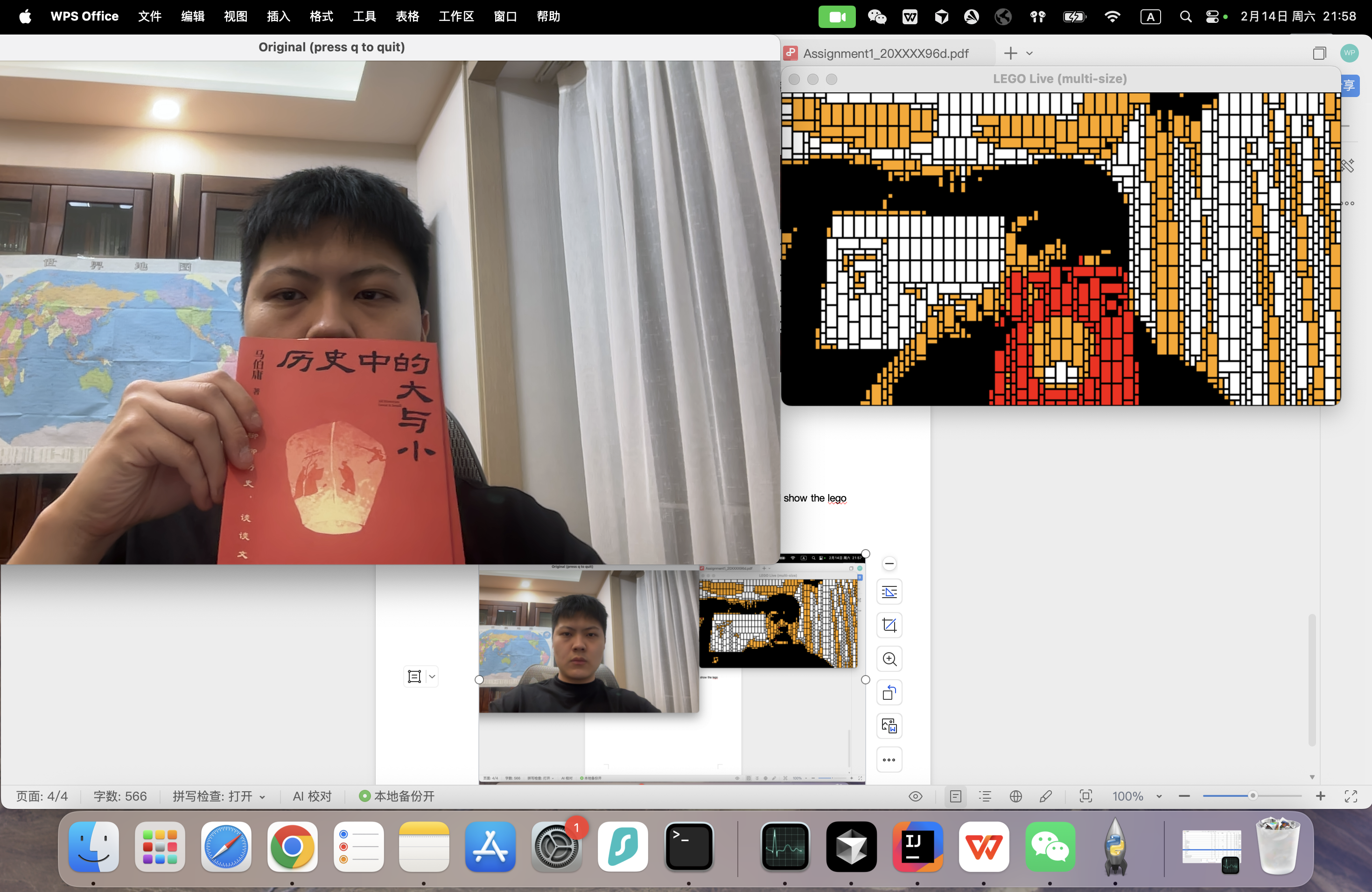
1x1: 186

1. Task 4:

Run python3 main.py --mode live, a window would pop up and show the lego style image and the original image from computer camera.

Some images are shown below.





1. Robustness

In real scenario, there could be too light, too dark, or the image could be too large or too small.