COMP90086 – Computer Vision – 2022 S2
Assignment 1 – Image Formation Filtering

Student: Jiahao Chen ID: 1118749

Task 1. Mapping between world and image coordinates

The height of the sculpture is calculated as 3.12 m using the pinhole projection model. The following figure shows the model for this scenario.

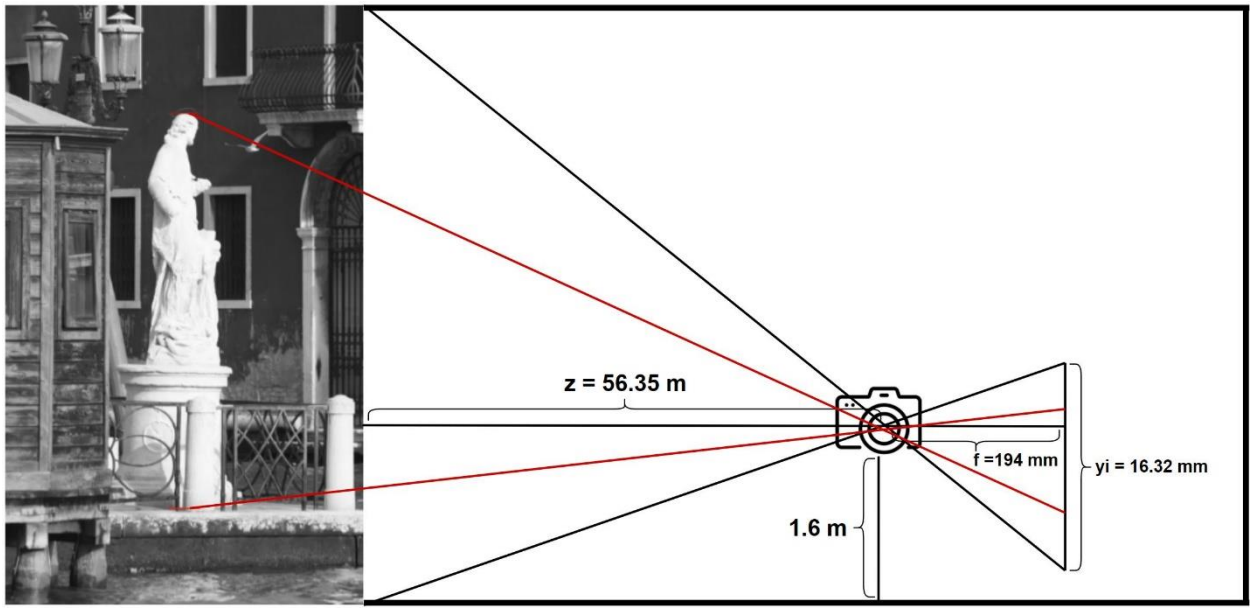

Figure 1: Pinhole projection model

Firstly, the actual height of the whole scene in the image is calculated. Since we only focus the height, the x axis is omitted. Given the data, the actual height of the scene can be calculated using properties of similar triangles, and the result is 4.74 m.

Next, the proportion of the sculpture's height in the height of whole scene is calculated to get the answer. To compute this proportion, two red lines in the image are useful because they represent the top and bottom of the sculpture respectively. In this case, if the y coordinates of two red lines can be found, the proportion is easy to compute. Therefore, each column of pixels is traversed until the first column contains red pixels are found, and there is no need to traverse the rest.

Simply compute the difference between y coordinates of two lines and divide it by the image's height, the proportion is achieved. The sculpture's height is the product of the proportion value and the actual height of the scene. Detailed calculation steps are included in the Jupyter Notebook.

Task 2. Thinking with filters

I. Image preprocessing

In the original picture, walls in the maze are two pixels wide, which may not be convenient when filters are applied. To facilitate subsequent operations, the image is first resized using nearest-neighbour interpolation so that walls have width of one pixel as figure 2 shows.

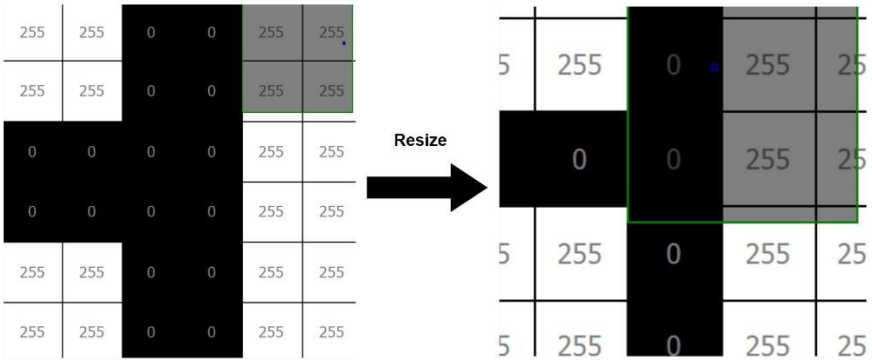

Figure 2: Resize picture

II. Detect intersections

One filter is applied to detect intersections, using the following kernel.

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 0 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Since intersections branches into 2 or more paths and there are 4 available branching directions (up, down, right, left), the kernel can give a small result for intersection due to the large negative weight on these 4 directions. As figure 4 depicted, each intersection becomes single white point, and the rest are all black. The number of white pixels is therefore the answer. Maze 1 has 87 intersections and maze 2 has 71 intersections.

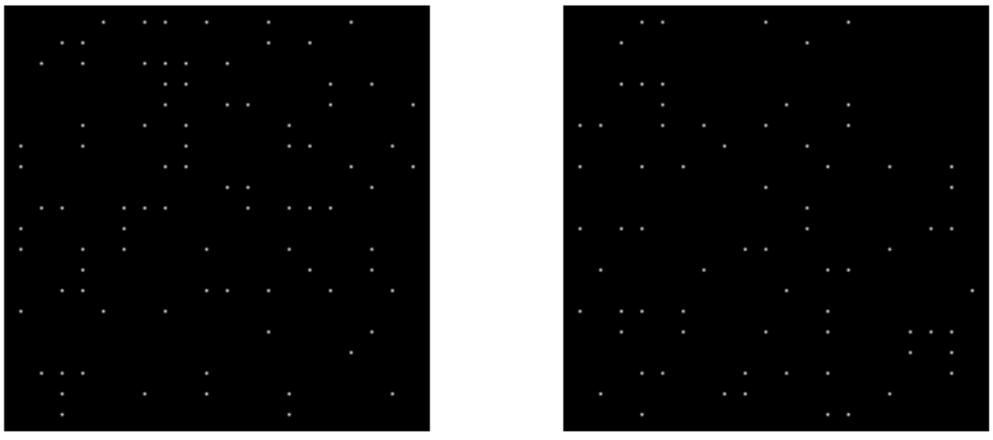

Figure 4: Result of detecting intersections

III. Detect dead ends

There mainly two steps in detecting dead ends. A filter is applied at first using the following kernel.

$$\begin{bmatrix} 1 & -1.1 & 1 \\ -1.1 & 0 & -1.1 \\ 1 & -1.1 & 1 \end{bmatrix}$$

The aim is to find a way to set dead end points apart from other pixels. The logic is similar to detecting intersections, however, the weight on 4 directions is smaller so that the pixels will not be scaled to either 0 or 255. Dead ends become the darkest in all wall pixels with value of 178 as figure 5 shows and figure 6 gives the results of maze 1 and maze 2 using this filter.

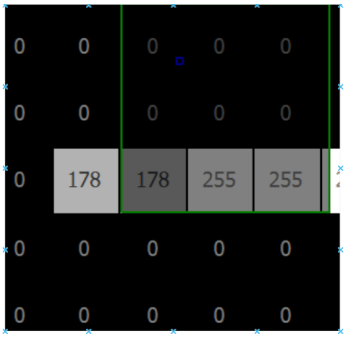

Figure 5: Dead end pixels

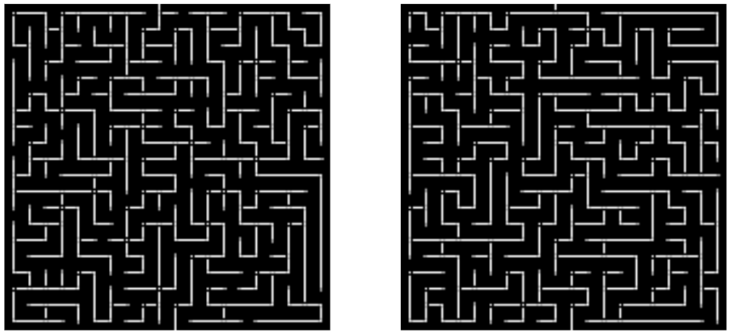

Figure 6: Apply filter on maze1 and maze2

Next, the threshold function is used to convert all pixels with value greater than 178 to 0 so that all pixels are black except dead ends. Figure 7 shows the result after applying the threshold function. Note that all dead ends are composed of 2 pixels, which indicates that the number of dead ends is half the number of pixels with value of 178. Maze 1 has 98 dead ends and maze 2 has 74 dead ends.

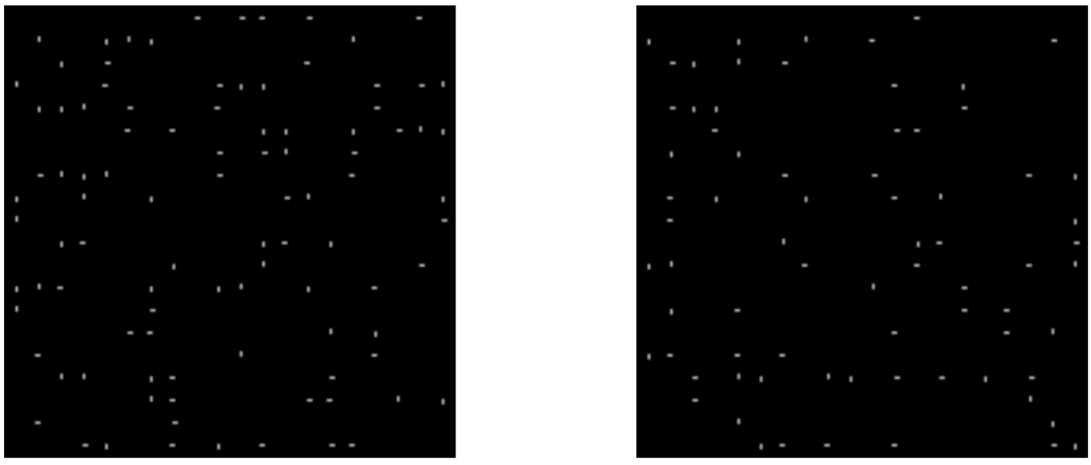

Figure 4: Result of detecting dead ends