CSC420: Assignment 2

Xiangyu Kong, kongxi16

October 9, 2019

Problem 1

See *question_1.py* for implementation.

1. The result of applying linear interpolation to up-sample the image by 4 times are shown in Figure 1.

   Figure 1a shows the first round of convolution along the first axis.

   Figure 1b shows the second round of convolution along the second axis on top of the result of the first round.

   Comparing to the original image, the up-sampled image maintains most of the information because the images look alike.

(a) First round (b) Second Round (c) Original Image

Figure 1

To achieve the same operation (up-sampling by a factor of 4), we can multiply the 1-dimensional filter by itself to get a 2-dimensional filter. This filter will be separable, and will produce the same result as convolving with the 1-dimensional filter twice on different directions.

In this specific case of up-sampling by a factor of 4, we calculate that

$$h = \begin{bmatrix} 0.25 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0.25 \end{bmatrix}$$

$$h \times h^T = \begin{bmatrix} 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.375 & 0.5 & 0.375 & 0.25 & 0.125 \\ 0.1875 & 0.375 & 0.5625 & 0.75 & 0.5625 & 0.375 & 0.1875 \\ 0.25 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0.25 \\ 0.1875 & 0.375 & 0.5625 & 0.75 & 0.5625 & 0.375 & 0.1875 \\ 0.125 & 0.25 & 0.375 & 0.5 & 0.375 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.1875 & 0.25 & 0.1875 & 0.125 & 0.0625 \end{bmatrix}$$

2. The generalized two dimensional linear interpolation reconstruction filter is

$$h = \begin{bmatrix} \frac{1}{d} & \dots & \frac{d-1}{d} & 1 & \frac{d-1}{d} & \dots & \frac{1}{d} \end{bmatrix}$$

$$h \times h^T = \begin{bmatrix} \frac{1}{d}^2 & \dots & \frac{d-1}{d} \times \frac{1}{d} & 1 \times \frac{1}{d} & \frac{d-1}{d} \times \frac{1}{d} & \dots & \frac{1}{d}^2 \\ \frac{d-1}{d} \times \frac{1}{d} & \ddots & \dots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & 1 & \dots & \dots & \vdots \\ \frac{d-1}{d} \times \frac{1}{d} & \dots & \dots & \dots & \dots & \dots & \frac{d-1}{d}^2 \end{bmatrix}$$

Since this is a separable filter (composed of $h \times h^T$), it is equivalent of applying the 1D linear interpolation filter twice in both direction.

The results of the 2D linear interpolation filter convolution is shown in Figure 2a. Comparing to the result of applying 1D linear interpolation filter twice in two directions (Figure 2b), the results do not have any difference.

(a) 2D Filter (b) 1D Filter (c) Original Image

Figure 2

Problem 2

See *question_2.py* for implementations.

1. The outputs of the corner detection functions are listed in Figure **??**.

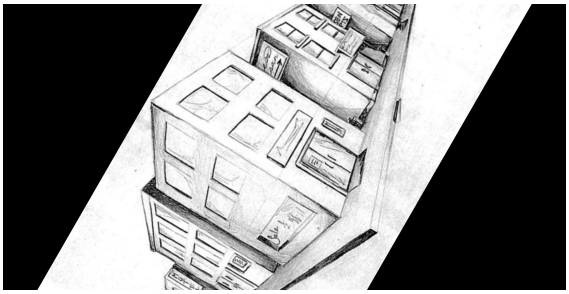
   After tuning with different alpha values, $\alpha = 0.06$ was picked because it produced the least amount of noise.

Figure 3: Harris corner detection

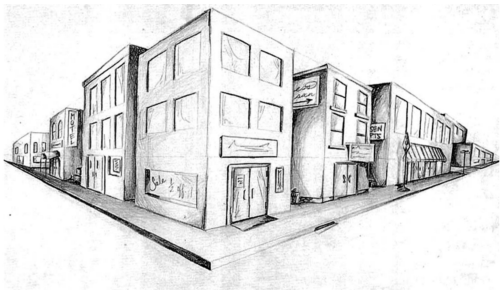
Figure 4: Brown corner detection

The only way for us to calculate the R scores without using determinant or trace is to calculate the eigenvalues for each window's error term. However, this is extremely computationally heavy and does yield a better result than computing the trace and determinant.

2. First we rotate the original image to produce Figure 5a



(a) Rotated image



(b) Original Image

Figure 5

By applying Harris corner detection to the rotated image, we get Figure 6. We can see that the detected corners are also rotated.

Figure 6: Rotated Haris corner detection

By rotating the detected corners, we get the following Figure 7.

To prove that the detected points are also rotated by the same angle, we count the number of detected corners in the rotated image ($num\_total\_rotated$). Then we also count the number of corners in the original image corresponding to the corners in the rotated image (num_match). Then we take the fraction of those two $\frac{num\_match}{num\_total\_rotated}$. The result was 99% which is sufficient to prove that the detected corners are the same.

Note that we didn't count the total number of detected corner in the original image ($num\_total\_original$) and compare it with the number of corners detected in the rotated image because rotating the image will caused some information loss, and this measurement won't be accurate.

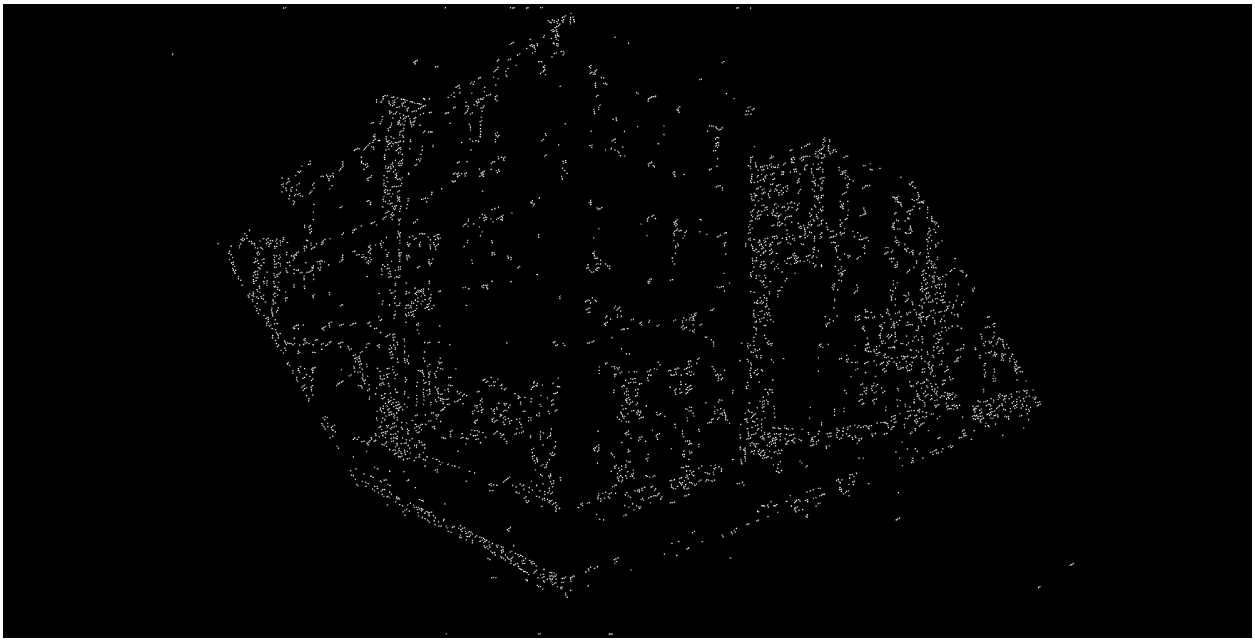


Figure 7: Un-rotate detected corners

Problem 3

1.