AA 274A: Principles of Robot Autonomy I
Problem Set 3

Name: Jiaqiao Zhang
SUID: qiao1997

Problem 1

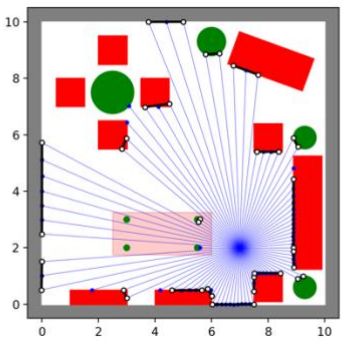
Please see the relevant code.

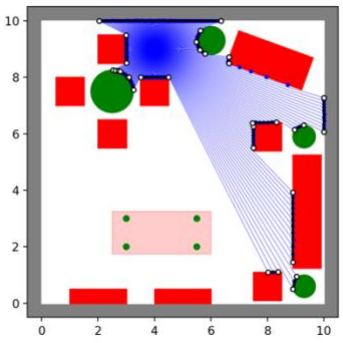
Problem 2

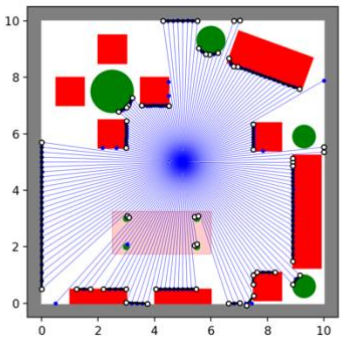
ii)

The segmentation parameters for the three plots are as follow:

MIN_SEG_LENGTH	0.02
LINE_POINT_DIST_THRESHOLD	0.03
MIN_POINTS_PER_SEGMENT	2
MAX_P2P_DIST	0.3







Problem 3

i)

a) $G = F \otimes I = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

b) $G = F \otimes I = \begin{bmatrix} 2 & 3 & 0 \\ 5 & 6 & 0 \\ 8 & 9 & 0 \end{bmatrix}$

c) $G = F \otimes I = \begin{bmatrix} 7 & 4 & -7 \\ 15 & 6 & -15 \\ 13 & 4 & -13 \end{bmatrix}$

It acts as derivative along x direction. It is useful for finding the differentiation when detecting the edges

d) $G = F \otimes I = \begin{bmatrix} 1.31 & 2.25 & 2.06 \\ 3.25 & 5 & 4.25 \\ 3.56 & 5.25 & 4.31 \end{bmatrix}$

It applies the Gaussian Filter to matrix I. It smooths the given image which can reduce the noise when finding the edge.

ii)

Vector f could be simply found as simply reshape the filter matrix F in a column vector. The vector f is shown as follow:

$$f = [F(0,0,0), F(1,0,0), F(2,0,0), \dots, F(k,l,c)]^T$$

The size of f^T is $1 \times (k \times l \times c)$.

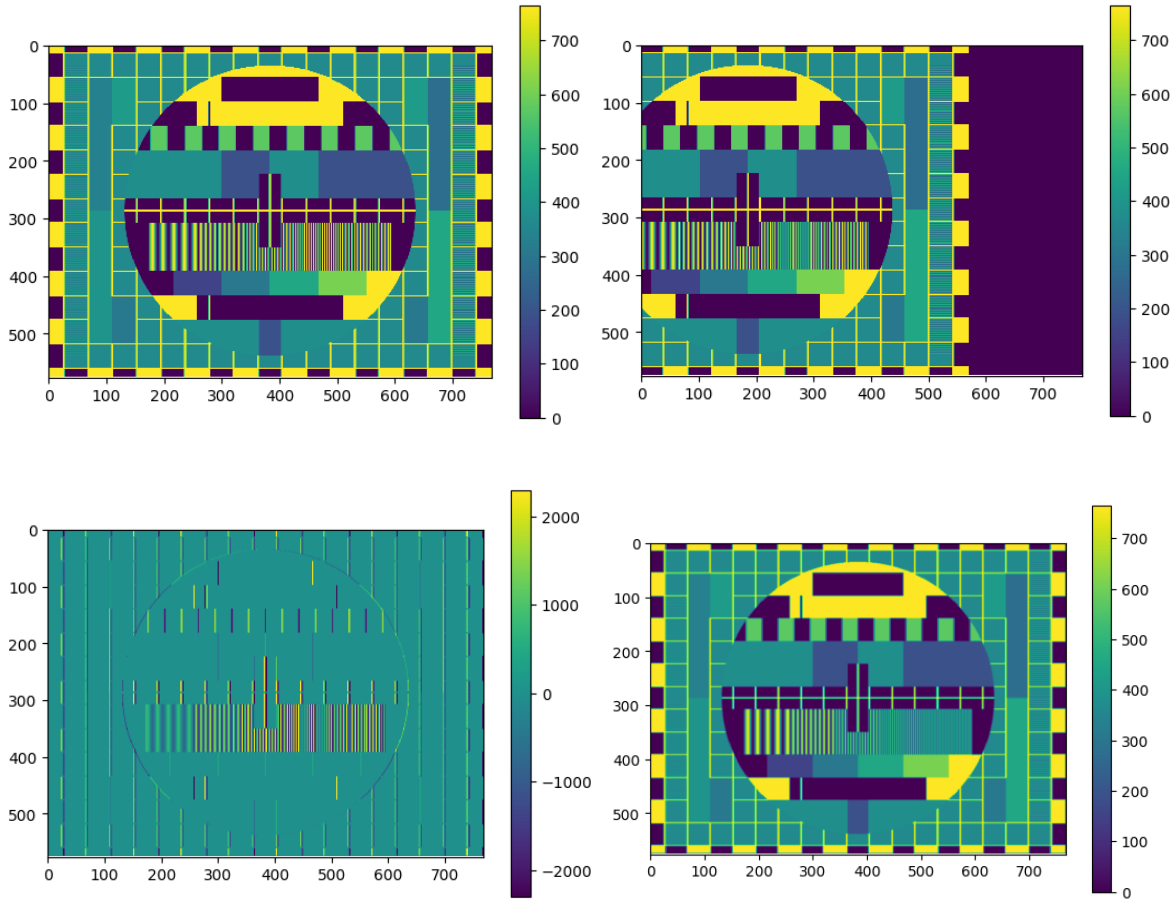
Vector t_{ij} is a column vector which contains all the corresponding pixels around (i,j). The t_{ij} is shown as:

$$t_{ij} = [I(i,j,0), I(i+1,j,0), I(i+2,j,0), \dots, I(i+k,j+l,c)]^T$$

The size of t_{ij} is $(k \times l \times c) \times 1$.

iii)

The plots are shown below.



iv)

The correlation time are

```
Correlation function runtime: 1.83349204063 s
Correlation function runtime: 2.63943099976 s
Correlation function runtime: 1.83000397682 s
Correlation function runtime: 1.91525816917 s
```

Speed up method:

1. We can rewrite the whole f and t_{ij} for each pixel in the image as a matrix.

$$F = f^T, \qquad T = [t_{00} \quad t_{10} \quad t_{20} \quad \dots \quad t_{wh}]$$

Where each column of T is t_{ij}. Therefore, we could obtain the G by matrix multiplication by F and T

$$G = F \times T = [G_{00} \quad G_{10} \quad G_{20} \quad \dots \quad G_{wh}]$$

Assume the image size is w x h

$$G \in R^{1\times(w\times h)}$$

We can simply reshape the G to get the desired image correlation.

2. The filter F can be separated as outer product of two vectors f.

$$F = f^T f$$

Where $F \in R^{k \times k}$, $f \in R^k$

Assume the mask is $k \times k$, and the image size is $w \times h$.

- The run time of the original filtering (without separation): $O(k^2 wh)$
- The run time of the separated filtering: $O(2kwh)$

Therefore, the runtime could be sped up in this way.

v)

The rank of the filter matrix F should be 1 to enable it to be separated as outer products of two vectors.

vi)

Please see the relevant code.

vii)

For convolution, it is similar to correlation, but it rotates the filter matrix by 180 degree. So, the implementation of convolution is same to that of correlation with the filter rotating 180 degree.

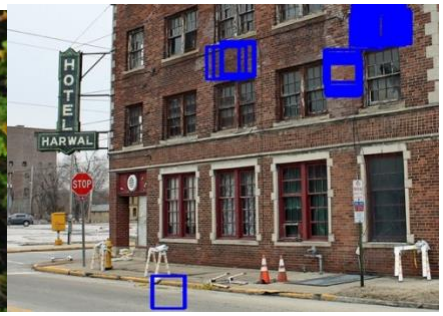
The results of convolution and correlation are the same when the filter matrix is symmetric.

Actually, correlation measures the similarity between the filter and image.
Convolution measures the effect of the filter on the image.

Problem 4

i)

ii)



iii)

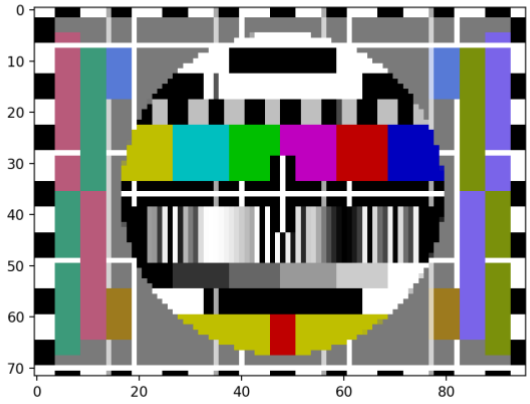
The perform of template match on other image is actually not good as shown in pictures above.

Improve template matching method:

1. Using multiple templates to reinforce the matches.
2. Break the template into parts and perform template match on each of the parts. And only accept when all of the parts are found.
3. Using CNN method

Extra Problem

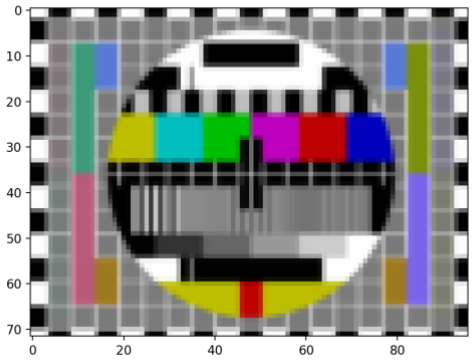
ii)



The quality of this image is not good compared to the original picture. We lose a lot of white and black bars. And also, the gray grid at background are lost. Because in this method, we only select even-indexed row and column to create the image. Therefore, we lose pixel information in the unselected rows and columns.

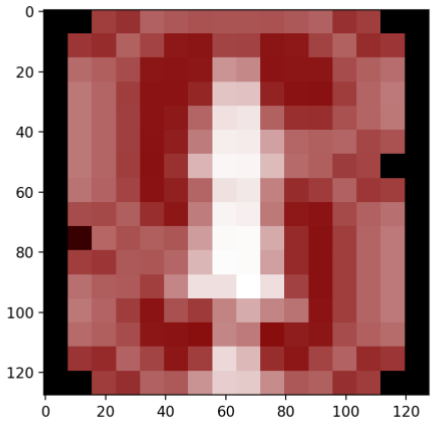
iii)

Blur the image at first will fix this issue since it takes the information from the adjacent pixels. Therefore, the pixel information in the lost columns and rows will be reserved after the blur and hence it will not completely lose the information.



Comparing to the picture with half_downscale, the quality of this picture is better since it contains the grid and most of the color bars. Comparing to the original picture, it contains most of the original pixel information, but the overall picture is blurred due to the method we are using.

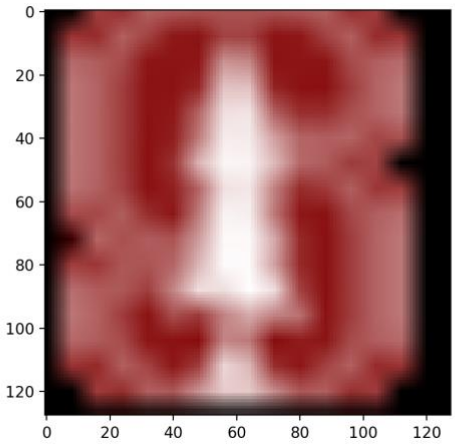
vi)



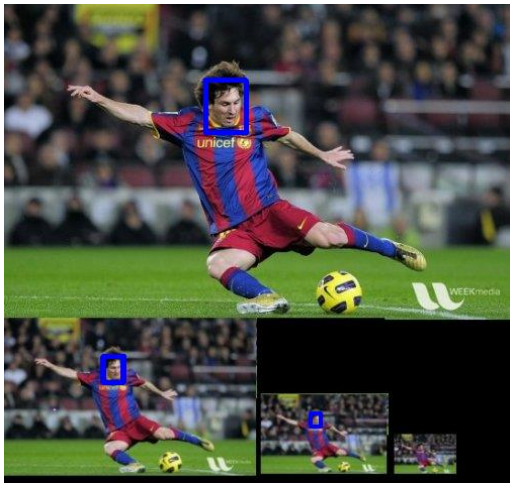
The picture using two_upscale is shown above, the quality is not very good since it is very blocky. The reason is that we simply repeat the rows and columns when upscale the image. It results in that there will be a lot of clusters with same pixels, which looks like a lot of blocks in the image. The color transition is not considered in this method.

vii)

For the bilinear interpolation, it takes the pixels at the “corners” and interpolate to get the remaining pixels. In this way, the transition between the pixels is smoother due to the interpolation



viii)



ix)

The perform of this algorithm on the stop sign images are not good. Its detection includes a lot of box contains red color. It recognizes the bricks on the wall as a stop sign. For some of the stop sign, it cannot detect it completely. The problem is that when we do the correlation, we consider every correlation larger than threshold as a match without filtering it. So, some of the matches is not actually a match for complex and irregular object.