Queen Mary
University of London

Introduction to Computer Vision

Coursework

Submission 1

Your name **GUANBIN HUANG**

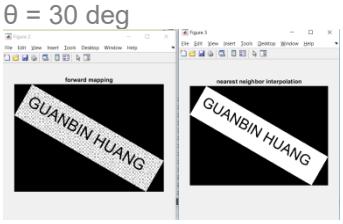
Student number **190620349**

Question 1(a):

GUANBIN HUANG

Rotated images:

θ = 30 deg



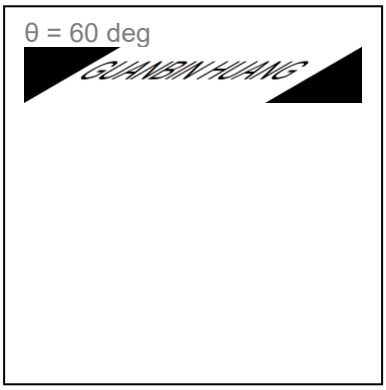
θ = 60 deg

θ = 120 deg

θ = -50 deg

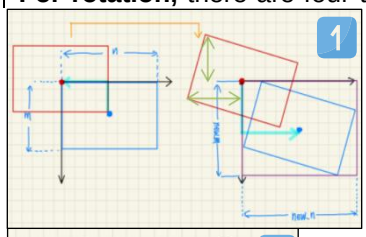
Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Skewed images:

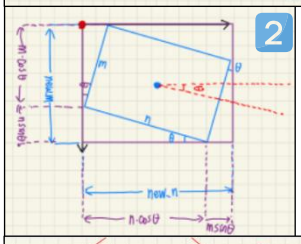


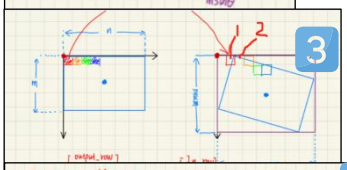
Your comments:

For rotation, there are four things needed to pay attention to. (1) how to get the new canvas; (2) how to do and undo the transformation; (3) forward mapping or inverse mapping.

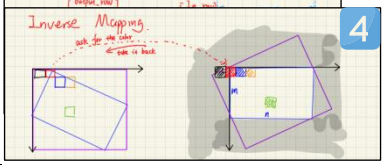


(1) rotating the image is multiplying the coordinates of old image by a rotation matrix, therefore, it's very likely to get some new coordinates beyond old borders. To extend our boundary, new canvas is needed. The methods experimented in lab are **[1]**.create a very large canvas; **[2]**. do the "minimum translation" the rotated red image (note the green arrows on the left image);**[3]**. trigonometry. However, the method **[1]** will make too much unnecessary black part which are pixels without corresponding values of the original image; the method **[2]** requires us to obtain the coordinates of image before the second translation. That means the integration of the three transformation matrices is impossible. The method **[3]** fits the image perfectly and canvas can be created at the very first beginning.



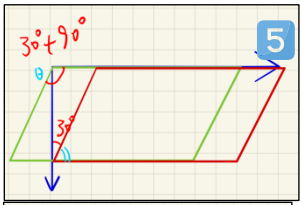


(2) Matrix multiplication follows the "from inside to outside" principle (T1-T2-T3: T3*T2*T1*M1).

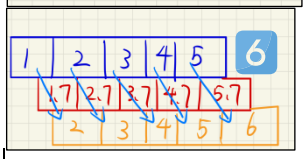


(3)Inverse mapping is clearly better. The reason has been visualized below(image 3 and image 4). The holes that exists in forward mapping are coming from the fact that after being rotation, the coordinates need to be rounded up, which leads to some of pixel values that used to be together separated(on image 3, the little red square might be position 1 or 2 after rotation. It all depends on the rotation matrix, when it is on position 1, the holes are generated). In a nutshell, due to the round up function, output image will have many pixels that cannot be mapped into.To avoid this, we can apply inverse mapping. The main idea of inverse mapping is assuming that the output is a perfect image, which means every pixel in rotated image should be filled with a certain value from the original image. Therefore, each pixel should be mapped back into the original image. Thus we can make sure the pixels are fully filled.

For shear, theoretically, the only difference is the transformation matrix that is easy to be replaced. In practice, shear degree and mapping strategy need to be considered.



(1) due to the unconventional coordinates axis and the need to conform with the default effect of shear, when we shear the image by 30 degree, we actually need to shear it by 120 degree and translate it to right(look at image 5).



(2) Here we don't need to use inverse mapping, because we will not have the trouble of "round up". For example, (look at the image 6 which represents a row of coordinates of pixels), no matter how much translation you do, the pixels end up with being together.

Question 1(b):

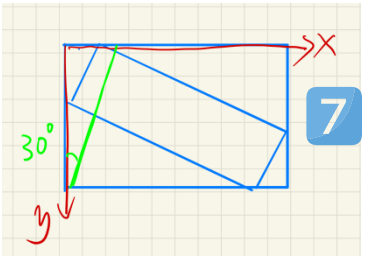
θ_1=20 clockwise and θ_2=50

θ_2=50 and θ_1=20 clockwise

Your comments:

They are not the same. Because the images are processed in two different operations. Although the elements of the operations are the same, multiplication of matrix is not commutative. In fact if what we really want is rotating our name by 20 degree and shearing our name by 50 degree, shear operation should be the first step. Because each operation should have its own baseline through which we measure the transformation.

When we do the rotation to our name whenever we do, we keep rotating our image around its center which is exactly the center of our name banner. That means we always conform to the same norm, which is what we want. However, when we shear our name banner, that is not the case. When putting shear as the first step, we skew our name banner by 30 degree like image 5. But when putting shear as the second step, the skew degree is not 30 degree anymore(look at image 7). Because you are skewing the whole image by 30 degree rather than the name banner.



Question 2(a):

Designed kernel:

Averaged filter

1 1 1
1 1 1
1 1 1

Notes: this filter needs to be multiplied by 1/9 as normalization.

Original image

Averaged image

Your comments:

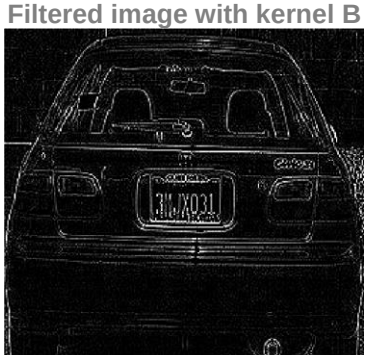
After being processed by the averaged filter, the output image is blurred. Because averaged filter means when we try to compute the value of a new specified pixel, we always need to consider the pixels around it and average these all pixel values. That means if the pixel value to be changed was higher than its neighborhood, it will end up being lowered and vice visa. Therefore, the gap of values between pixels become smaller. Thus we get the smoothing effect.

The need for normalization is to conserve the total “energy” of the image.

Question 2(b):

Filtered image with kernel A

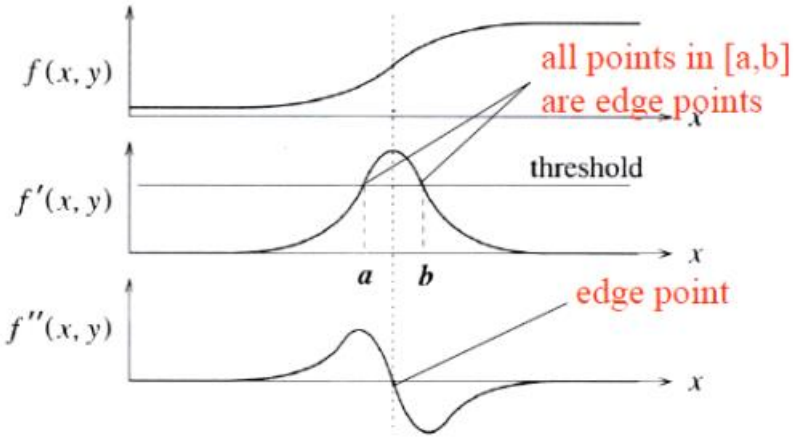
Filtered image with kernel B



Your comments:

1. For kernel A, we can see that Gaussian filter performs better than averaged filter when it comes to smoothness. It is because when computing a specified pixel, Gaussian filter puts more weight on its own value and the closer neighborhood, and less weight on the farther pixels. This design enables Gaussian filter to conserve more edge information. Because the neighborhood of edge will be more likely to be edge, edge value wouldn't be lowered too much when doing Gaussian filter(weighted averaging).

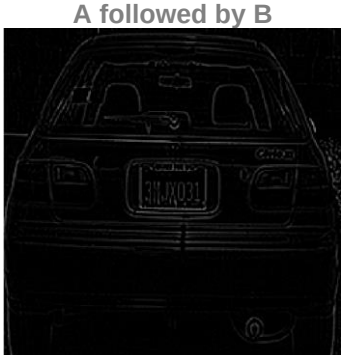
2. For kernel B, this kernel is used for edge detection, because the pattern of this kernel is coming from the second derivative of f(x,y). As we can see on the image below, whenever increase or decrease happens on f(x,y), there will be a corresponding edge point on the f''(x,y). That means we can detect the edge of an image by taking the second derivatives of the image.



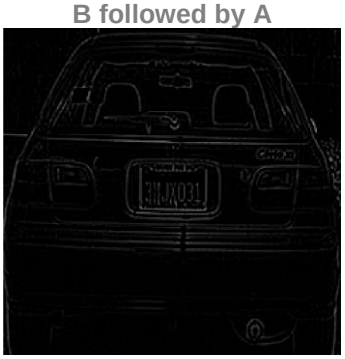
Question 2(c):

A followed by A

A followed by B



B followed by A



Your comments:

The first image are more blurred, because it is blurred by Gaussian filter twice; the second image and the third image show the same result, because convolution is commutative.

Compared with the case only using the kernel B, AB or BA has less noise due to the smoothing done by Gaussian filter before edge detection.

Note that all the way through the convolution, you need to make sure the data type you manipulate is double. You can imshow an image by uint8(0-255) or double type((0-1)), but you are not expected to convert the image filtered by A into uint8((0-255)) and input it into filter B. because when you do that, that means you convert 0-1 to 0-255, and then, again, convert 0-255 into 0-1, which leads to a loss of precision.

Question 2(d):

Extended kernels of A and B (5x5):

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

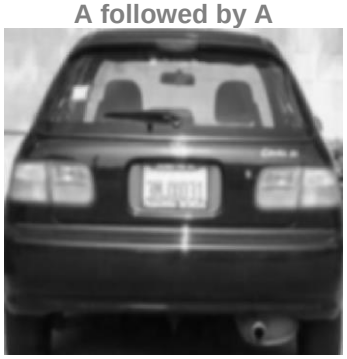
Filter A(5 x 5)

```
[0, 0, 1, 0, 0;
 0, 0, 1, 0, 0;
 1, 1,-8, 1, 1;
 0, 0, 1, 0, 0;
 0, 0, 1, 0, 0];
```

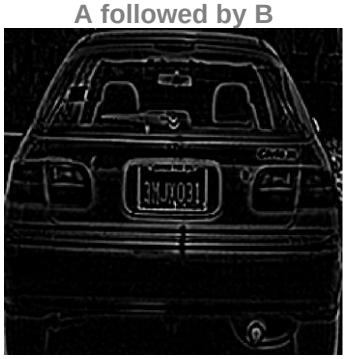
filter B (5x5)

Results obtained by applying 5x5 kernel:

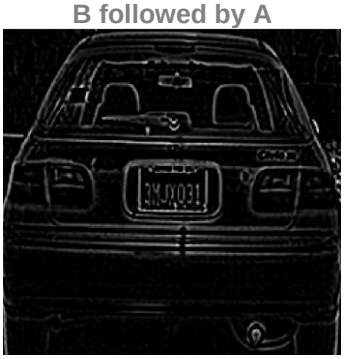
A followed by A



A followed by B



B followed by A



Extended kernels of A and B (7x7):

1.9651e-05	2.3940e-04	0.0011	0.0018	0.0011	2.3940e-04	1.9651e-05
2.3940e-04	0.003	0.013	0.022	0.013	0.003	2.3940e-04
0.0011	0.013	0.059	0.097	0.059	0.013	0.0011
0.0018	0.022	0.097	0.159	0.097	0.022	0.0018
0.0011	0.013	0.059	0.097	0.059	0.013	0.0011
2.3940e-04	0.003	0.013	0.022	0.013	0.003	2.3940e-04
1.9651e-05	2.3940e-04	0.0011	0.0018	0.0011	2.3940e-04	1.9651e-05

```
filter_7B = [
0 0 0 1 0 0 0;
0 0 0 1 0 0 0;
0 0 0 1 0 0 0;
1 1 1 -12 1 1 1;
0 0 0 1 0 0 0;
0 0 0 1 0 0 0;
0 0 0 1 0 0 0];
```

Results obtained by applying 7x7 kernel:

A followed by A

A followed by B

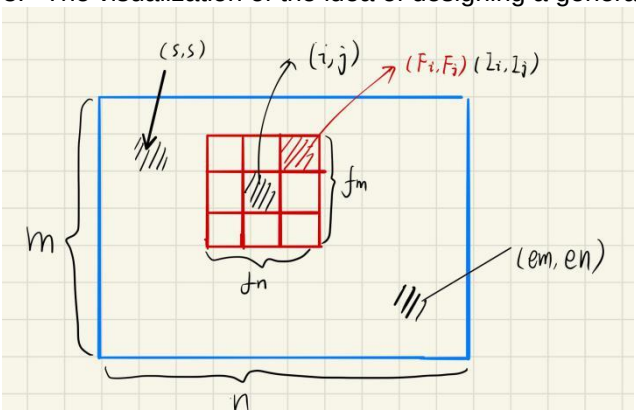
B followed by A

Your comments:

1. For gaussian filter, we can apply the formula $$G_\sigma(k,l) = \frac{1}{2\sigma^2\pi} e^{-\frac{k^2+l^2}{2\sigma^2}}$$. All you need to do is specify the sigma. Here sigma is specified to be equal to 1. Therefore we can get the 5x5 and 7x7 gaussian filter. However, the smoothing strength of these two are similar or more exactly, 7x7 has a slightly better smoothing because there are some very small weight on its far neighborhood that 5x5 gaussian filter doesn't have. However the determinant of smoothing should be the sigma.

2. For laplacian filter, intuitively, we can see that the value of 3x3 laplacian filter is negative, and its horizontal and vertical values are 1. And the most important thing is the sum of the filter is 0. Following this pattern, we can design our laplacian filter. After we have our filter, we can see that as our size gets bigger, our edge detected is clearer.

3. The visualization of the idea of designing a general filter function:(stride = 1)



Question 3(a):

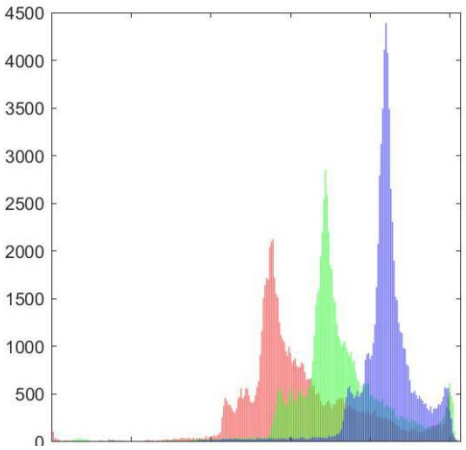
Two non-consecutive frames:

Image 1
Frame 7

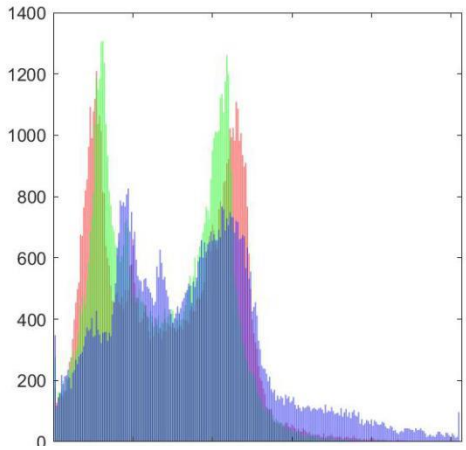
Image 2
Frame 12

Corresponding colour histograms:

Histogram 1
Frame 7



Histogram 2
Frame 12



Your comments:

As we can see, different scenes have different distribution of color histogram. For example, on the first image, intuitively, blue is the dominant color. So in histogram 1, in blue channel, more pixel values are distributed on the large pixel value. On the second image, since it seems like there is no color dominating the image, we get a more even distribution of histogram.

Question 3(b):

Example 1:

I_t

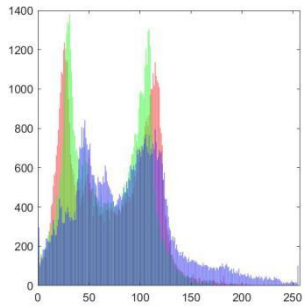
Frame 10

I_{t+1}

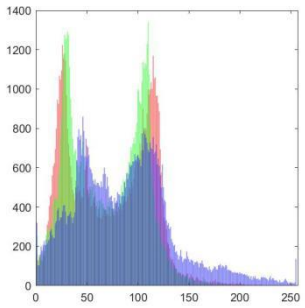
Frame 11

Histograms:

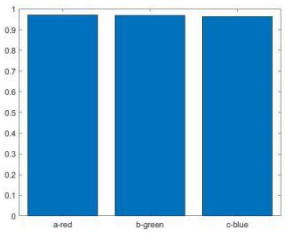
Histogram of I_t



Histogram of I_{t+1}



Intersection result



Example 2:

I_t

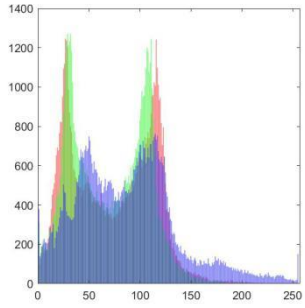
Frame 100

I_{t+1}

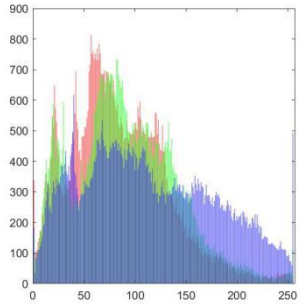
Frame 101

Histograms:

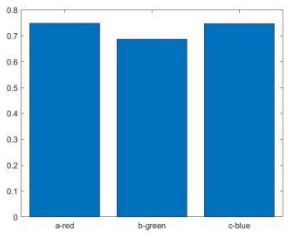
Histogram of I_t



Histogram of I_{t+1}



Intersection result



Your Comments:

For example 1, we can see we have a consecutive frames and its histogram. Since intuitively they look the same, their histogram are also similar. Therefore, their histogram intersection values are all very close to 1. That means the similarity is nearly 100%.

For example 2, the frames are two different shots and their histograms are very differently distributed. From their histogram intersection , we can see that they are all less than 80%. So we believe scene has been changed.

The two images of intersection result are both normalized by the total number of pixels of each frame. Before normalization, histogram intersection shows shows us the number of pixels that follow the same distribution. That means the values of each bar usually very large, up to more than several thousands. But after normalization, the maximum of them are all 1(also considered as 100%), which is used to measure similarity.

Question 3(c):

Comments: