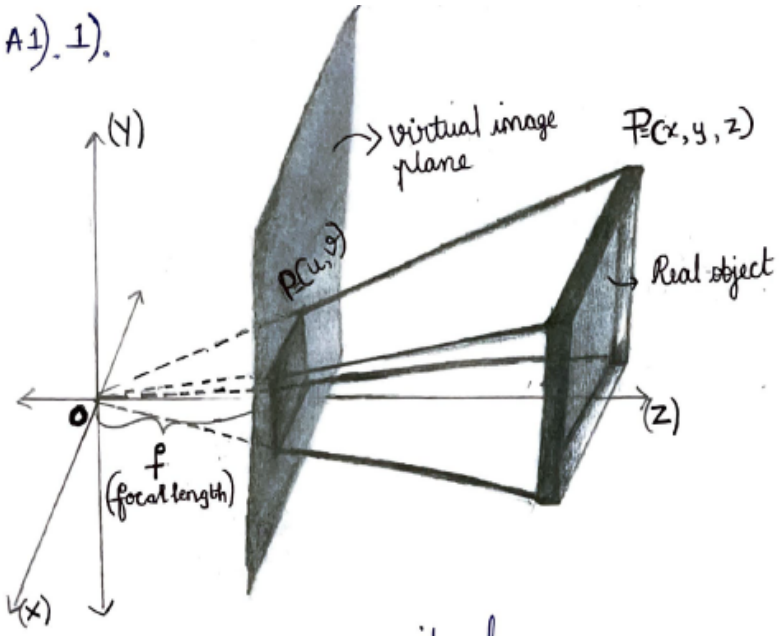
Date: 03/12/2021 **HOMEWORK-I** **Pruthviraj R Patil**

A1. Question 1.

Sketch a diagram of the pinhole camera model (either in 2D or 3D). Include camera origin O, world point P=(x,y,z), focal length f, and image point p=(u,v) on the image plane. Inclusion of the virtual image plane is optional. Derive expressions for image coordinates p=(u,v) in terms of 3D world coordinates P=(x,y,z) and focal length f.

A1). 1).



[Note: Virtual plane is considered such that the distance of O to virtual plane is: focal length "f"]

From the image, we come to know that the points O, p & P are collinear.

$\therefore \vec{OP} = \lambda \vec{OP} \rightarrow (1)$

p(u,v) are points on the virtual image plane & P(x,y,z) over real object

$\therefore$ from equation (1), we get:

$u = \lambda x \; ; \; v = \lambda y$

but distance from O to virtual image plane = focal length = f.

$\therefore f = \lambda z \rightarrow (2)$

hence: $\lambda = \frac{u}{x} \rightarrow (3)$

$\lambda = \frac{v}{y} \rightarrow (4)$

hence, eqn (2) can be re written as:

$f = \frac{u}{x} z$ or $f = \frac{v}{y} z$

Therefore: $P(u,v) = P\left(\frac{fx}{z}, \frac{fy}{z}\right)$

A1. Question 2.

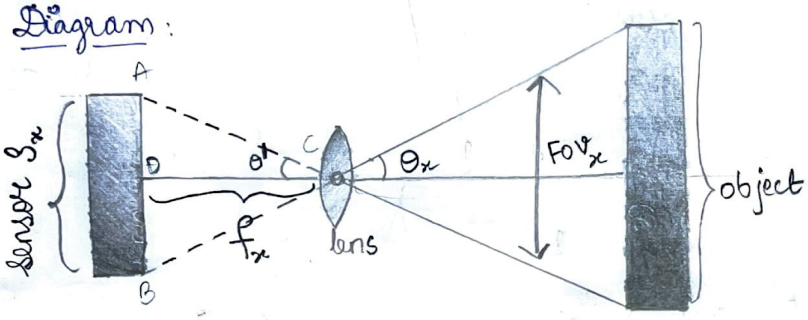
Zoe is becoming interested in photography, but currently only takes pictures using her smartphone. She is considering purchasing a full-frame 12-megapixel digital camera with a sensor size of 36mm x 36mm (square) and a focal length of 46mm. She is curious about how this compares to her smartphone, which is also 12 megapixels with a focal length of 5mm. She can't find information about the size of the sensor on her smartphone, but she knows that it is square and has the same field of view as the digital camera. That is, the digital camera and the smartphone capture an identical scene if placed at identical locations.

Part A. Using concepts from the pinhole camera model, calculate the size and area of Zoe's smartphone sensor. How does this compare to the area of the camera she is considering? Show your work and include explanations so Zoe knows how to do this herself next time.

Part B. Calculate the size of a sensor pixel element for the digital camera and smartphone camera. How do they compare?

Solutions:

2) Diagram:



a) given:

for the camera [digital]:

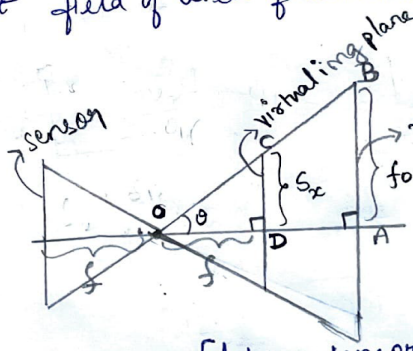
focal length f_1 = 46 mm.

Sensor size S_1 = 36mm × 36mm.

for the smart phone camera:

focal length f_2 = 5mm

also given that field of view of camera(digital) = Smartphone camera's fov.



CD = Sensor side len / 2

AB = fov / 2

OD = focal length

∴ from above diagram, [taking sensor size over virtual img plane]

$fov_1 = fov_2$ (given that fov of digital cam = fov of smart phone cam)

f_1 = 46mm ; f_2 = 5mm ; $S_1 = \frac{36}{2}$; S_2 = ??

from similarity of Δ'les we know that, for digital cam: $\frac{f_1}{S_1} = \frac{OA}{fov_1}$ → ①

similarly for smartphone camera: $\frac{f_2}{S_2} = \frac{OA}{fov_2}$ → ②

but $\frac{OA}{fov_1} = \frac{OA}{fov_2}$ (given) ∴ $\frac{f_1}{S_1} = \frac{f_2}{S_2}$ → ③

Solution continued…

hence Equation (3) becomes:

$$\frac{46\text{ mm}}{\left(\frac{36}{2}\right)\text{mm}} = \frac{5\text{ mm}}{S_2}$$

$$\therefore \quad S_2 = \left(\frac{36}{2} \times \frac{5}{46}\right)\text{mm} = \left(\frac{3.913}{2}\right)\text{mm}.$$

but S_2 is half × sensor side length of smart phone

$\therefore$ Size of sensor of smartphone = $(3.913\text{ mm} \times 3.913\text{ mm})$

b) For the digital camera; sensor size = $(36^{mm} \times 36^{mm})$.

For the smart phone camera; sensor size = $(3.913^{mm} \times 3.913^{mm})$

But both can accomodate same number of pixels in the o/p image. i.e 12 mega pixels = 12×10^6 pixels -

hence, per pixel sizes:

⊛ for digital camera:

$$\text{Area per pixel} = \frac{(36 \times 36) \times 10^{-6}}{12 \times 10^6}$$

$$= 108 \times 10^{-12}$$

$\therefore$ Size of pixel = $\sqrt{108 \times 10^{-12}}$

= $(10.39\ \mu m \times 10.39\ \mu m)$

⊛ for smart phone camera:

$$\text{Area per pixel} = \frac{(3.916 \times 3.916) \times 10^{-6}}{12 \times 10^6}$$

$$= 1.278 \times 10^{-12}$$

Size of pixel = $\sqrt{1.278 \times 10^{-12}}$

= $(1.13\ \mu m \times 1.13\ \mu m)$

Hence size of pixel in digital camera is approximately 84.13 times larger than size of pixel in smartphone.

A2) Convolution and Cross-Correlation

1)

$I =$	3	6	8	3	5	1

$f =$	-1	0	1

$g =$	-1	0	1

Using the above values for I, f, and g, show that convolution is associative while cross-correlation is not. Recall associativity means $f * (g * I) = (f * g) * I$. Show all work.

Solution:

A2: Convolution and cross-correlation

1). I = | 3 | 6 | 8 | 3 | 5 | 1 | f : | -1 | 0 | 1 | g = | -1 | 0 | 1 |

Convolution:

LHS: $f * (g * i)$ | RHS: $(f * g) * i$

LHS: $(g * i) \Rightarrow$

g = | 1 | 0 | -1 | flipped

i = | 0 | 3 | 6 | 8 | 3 | 5 | 1 | 0 | padded.

$(g*i)$ = | -3 | -6 | -5 | 3 | 3 | 2 | 5 | 1 |

$(g*i)$ = | 0 | -3 | -6 | -5 | 3 | 3 | 2 | 5 | 1 | 0 | padded.

~~$f*(g*i)$ =~~

calculations for $(g*i)$:

1] (1)(0) + 0(0) + (-1)(3) = -3

2] (1)(0) + 0(3) + (-1)(6) = -6

3] (1)(3) + 0(6) + (-1)(8) = -8

4] (1)(6) + 0(8) + (-1)(3) = 3

5] (1)(8) + 0(3) + (-1)(5) = 3

6] (1)(3) + 0(5) + (-1)(1) = 2

7] (1)(5) + 0(1) + (-1)(0) = 5

8] (1)(1) + 0(0) + (-1)(0) = 1

$f*(g*i) \Rightarrow f =$ | 1 | 0 | -1 |

$(g*i) \Rightarrow$ | 0 | -3 | -6 | -5 | 3 | 3 | 2 | 5 | 1 | 0 |

$(g * \delta)$ = | 0 | -3 | -6 | -5 | 3 | 3 | 2 | 5 | 1 | 0 |

(f) = | 1 | 0 | -1 |

$f * (g * i)$ = | 3 | 6 | 2 | -9 | -8 | 1 | -2 | 1 | 5 | 1 |

calculations:

1) $1(0) + 0(0) + (1)(-3) = 3$
2) $1(0) + 0(-3) + (-1)(-6) = 6$
3) $1(-3) + 0(-6) + (-1)(-5) = 2$
4) $1(-6) + 0(-5) + (-1)(3) = -9$
5) $1(-5) + 0(3) + (-1)(3) = -8$
6) $1(3) + 0(3) + (-1)(2) = 1$
7) $1(3) + 0(2) + (-1)(5) = -2$
8) $1(2) + 0(5) + (-1)(1) = 1$
9) $1(5) + 0(1) + (-1)(0) = 5$
10) $1(1) + 0(0) + (-1)(0) = 1$

RHS:

$(f * g)$

f = | 1 | 0 | -1 |

g = | 0 | -1 | 0 | 1 | 0 | (padded)

$f * g$ = | 1 | 0 | -2 | 0 | 1 |

calculations:

1) $1(0) + 0(0) + (-1)(1) = 1$
2) $1(0) + 0(-1) + (-1)(0) = 0$
3) $1(-1) + 0(0) + (-1)(1) = -2$
4) $1(0) + 0(1) + (-1)(0) = 0$
5) $1(1) + 0(0) + (-1)(0) = 1.$

$f * g$ = | 1 | 0 | -2 | 0 | 1 |

i = | 0 | 0 | 3 | 6 | 8 | 3 | 5 | 1 | 0 | 0 |

$(f*g) * i$ = | 3 | 6 | 2 | -9 | -8 | 1 | -2 | 1 | 5 | 1 |

1) 1(0) + 0(0) + (-2)(0) + 0(0) + 1(3) = 3
2) 1(0) + 0(0) + (-2)(0) + 0(3) + 1(6) = 6
3) 1(0) + 0(0) + (-2)(3) + 0(6) + 1(8) = 2
4) 1(0) + 0(3) + (-2)(6) + 0(8) + 1(3) = -9
5) 1(3) + 0(6) + (-2)(8) + 0(3) + 1(5) = -8
6) 1(6) + 0(8) + (-2)(3) + 0(5) + 1(1) = 1
7) 1(8) + 0(3) + (-2)(5) + 0(1) + 1(0) = -2
8) 1(3) + 0(5) + (-2)(1) + 0(0) + 1(0) = 1
9) 1(5) + 0(1) + (-2)(0) + 0(0) + 1(0) = 5
10) 1(1) + 0(0) + (-2)(0) + 0(0) + 1(0) = 1

Hence LHS = RHS. ∴ convolution is associative (proved)

Cross correlation:

$f \cdot (g \cdot i)$ = LHS

RHS = $(f \cdot g) \cdot i$

LHS: $(g \cdot i)$.

g = | -1 | 0 | 1 |

i = | 0 | 3 | 6 | 8 | 3 | 5 | 1 | 0 | (padded.)

$(g \cdot i)$ = | 0 | 3 | 6 | 5 | -3 | -3 | -2 | -5 | -1 | 0 |

1) $-1(0) + 0(0) + 1(3) = 3$

2) $-1(0) + 0(3) + 1(6) = 6$

3) $-1(3) + 0(6) + 1(8) = 5$

4) $-1(6) + 0(8) + 1(3) = -3$

5) $-1(8) + 0(3) + 1(5) = -3$

6) $-1(3) + 0(5) + 1(1) = -2$

7) $-1(5) + 0(1) + 1(0) = -5$

8) $-1(1) + 0(0) + 1(0) = -1$

$f =$ | -1 | 0 | 1 |

$f.(g.i) =$ | 3 | 6 | 2 | -9 | -8 | 1 | -2 | 1 | 5 | 1 |

Calculations :

1) $-1(0) + 0(0) + 1(3) = 3$

2) $-1(0) + 0(3) + 1(6) = 6$

3) $-1(3) + 0(6) + 1(5) = 2$

4) $-1(6) + 0(5) + 1(-3) = -9$

5) $-1(5) + 0(-3) + 1(-3) = -8$

6) $-1(-3) + 0(-3) + 1(-2) = 1$

7) $-1(-3) + 0(-2) + 1(-5) = -2$

8) $-1(-2) + 0(-5) + 1(-1) = 1$

9) $-1(-5) + 0(-1) + 1(0) = 5$

10) $-1(-1) + 0(0) + 1(0) = 1$

(*) RHS:

for (f . g)

g = | 0 | -1 | 0 | 1 | 0 |

f = | -1 | 0 | 1 |

f . g = | -1 | 0 | 2 | 0 | -1 |

Calculations

1) -1 (0) + 0 (0) + 1 (-1) = -1
2) -1 (0) + 0 (-1) + 1 (0) = 0
3) -1 (-1) + 0 (0) + 1 (1) = 2
4) -1 (0) + 0 (1) + 1 (0) = 0
5) -1 (1) + 0 (0) + 1 (0) = -1

i = | 0 | 0 | 8 | 6 | 8 | 3 | 5 | 1 | 0 | 0 |

(f . g) = | -1 | 0 | 2 | 0 | -1 |

(f . g) . i = | -3 | -6 | -2 | 9 | 3 | -1 | 2 | -1 | -5 | -1 |

Calculations:

1) (-1) (0) + 0 (0) + 2 (0) + 0 (0) + (-1) (3) = -3
2) (-1) (0) + 0 (0) + 2 (0) + 0 (3) + (-1) (-6) = -6
3) (-1) (0) + 0 (0) + 2 (3) + 0 (6) + (-1) (8) = -2
4) (-1) (0) + 0 (3) + 2 (6) + 0 (8) + (-1) (3) = 9
5) (-1) (3) + 0 (6) + 2 (8) + 0 (3) + (-1) (5) = 3
6) (-1) (6) + 0 (8) + 2 (3) + 0 (5) + (-1) (1) = -1
7) (-1) (8) + 0 (3) + 2 (5) + 0 (1) + (-1) (0) = 2
8) (-1) (3) + 0 (5) + 2 (1) + 0 (0) + (-1) (0) = -1
9) (-1) (5) + 0 (1) + 2 (0) + 0 (0) + (-1) (0) = -5
10) (-1) (1) + 0 (0) + 2 (0) + 0 (0) + (-1) (0) = -1

∴ LHS ≠ RHS. Hence, Cross correlation is NOT Associative

2)

- If we convolve an anisotropic (not symmetric) N X N Gaussian filter with an image of size R X C, how many add and multiply operations will be required in total to filter the entire image?

- What if we use a symmetric N X N Gaussian filter instead? How many add and multiply operations will be required in total to filter the entire image?

Write your answers in terms of N, R, and C. You can handle the boundary in any way of your choice. Make it clear in your answer your choice for dealing with the boundary. Show all work.

Solution:

Ans. 2] a) Given image size : R×C ; Rows = R. columns = C.

filter size : N×N.

for the symmetric gaussian kernels, they are separable

Eg: $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

hence, A filter, which is not seperable, is asymmetric (anisotrophic). hence, the N×N filter has to pass over all the rows and columns operating over N×N (sized) image patches every time.

hence, there will be N^2 additions & multiplication for every N×N sized patch hence, in total, we have $N^2 \cdot R \cdot C$ additions and multiplications in the case of anisotrophic filter.

b) In the case of isotrophic (symmetric) filters, due to its property of separability, we can parse through image with $N \cdot (R \cdot C)$ operations for each separated part from filter (one rows & one column matrix). ∴ There are total $2NRC$ operations.

B) Programming Questions

B1) Image & Kernel Indexing: Cross-Correlation vs Convolution

Indexing issues are a common point of misunderstanding and error when applying convolution and cross correlation to an image. Because of this, your company has compartmentalized the index selection process into a method findOperationIndexPairs(idx, n, opType) and has asked you to implement. The method is passed three variables: idx, n, and opType.

• **idx**: a tuple representing an x and y index into an image, of form (x, y). You can assume that the image has been properly padded and that idx is a point within the body of the image (no need to worry about edge conditions).

• **n:** an integer representing the dimension of an n*n square filter/kernel being applied to the image.

• **opType**: is a boolean representing the type of operation.

- o True: Cross Correlation
- o False: Convolution

Your method should return a single variable: indexPairs.

• **indexPairs**: a list of pairs of tuples, where each pair represents the index into the image (first) and the index into the filter/kernel (second).

o **Ex**: [((x_img_0, y_img_0)(x_kernel_0, y_kernel_0)), … , ((x_img_i, y_img_i)(x_kernel_i, y_kernel_i))]

Each pair of coordinates represents indices into the image and kernel that will be used to retrieve the values which will be multiplied and summed together in the performance of the given operation. You are not expected to calculate the sum of these products, your methods should only return the correct indices so that later in the image processing pipeline another method can perform those functions. The size of your returned list depends on the size of the kernel, so for a kernel of size n, you will return a list of n*n pairs of coordinates. It is fine to modify the input/output structure to work better with your language/development environment, as long as it implements the core functionality

CODE (Python3):

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import math
from skimage import io
from skimage.color import rgb2gray

def createGaussianFilter(n, stdev):
    #status: complete

    #task: creates a square gaussian filter
    #parameters:
    #n= rows and column count
    #stdev= standard deviation

    variance=stdev*stdev #variance
    gaussian_filter=np.zeros((n, n))
    x=np.linspace(-n/2.0, n/2.0, n)
```

```
    y=np.linspace(-n/2.0, n/2.0, n)

    for i in range(0, n):
        for j in range(0, n):
            gaussian_filter[i,
j]=(1/math.sqrt((2*math.pi*variance)))*math.exp(-(x[i]*x[i]+y[j]*y[j])/(2*v
ariance))
    return gaussian_filter/np.sum(gaussian_filter.flatten())

def correlation_pairs(idx, fltr, n):
    #status: complete
    # task: to compute the indexes of image and kernel used in Correlation
operation
    out_list=[]
    col=idx[1]-2
    row=idx[0]-2

    filter_col=0
    for i in range(col, col+5):
        filter_row=0
        for j in range(row, row+5):
            pairs=(("img", j, i),("kernel", filter_row, filter_col))
            out_list.append(pairs)
            filter_row+=1
        filter_col+=1

    return out_list

def convolution_pairs(idx, fltr, n):
    #status: complete
    # task: to compute the indexes of image and kernel used in Convolution
operation

    out_list=[]
    col=idx[1]-2
    row=idx[0]-2

    filter_col=n-1
    for i in range(col, col+5):
        filter_row=n-1
        for j in range(row, row+5):
            pairs=(("img", j, i),("kernel", filter_row, filter_col))
            out_list.append(pairs)
            filter_row-=1
        filter_col-=1

    return out_list
```

```
def findOperationIndexPairs(idx, n, opType):
    #status: complete

    #task:
    #Parameters:
    #1. idx: tuple representing  x, y index into an image
    #2. n: integer representing n*n sq filter/kernel
    #3. opType: True-Cross Correlation. False: Convolution
    #return - indexPairs: a list of pairs of tuples, where each pair
represents the index into the image (first) and the index into the
filter/kernel
    #output format: [((x_img_0, y_img_0)(x_kernel_0, y_kernel_0)), ... ,
((x_img_i, y_img_i)(x_kernel_i, y_kernel_i))]


    #filter_considered: Gaussian
    #n = considered= 5: standard deviation= 20:
    fltr=createGaussianFilter(5, 20)

    if(opType==True):
        indexPairs=correlation_pairs(idx, fltr, n)
    else:
        indexPairs=convolution_pairs(idx, fltr, n)

    return indexPairs

def main():
    idx=(5, 5)
    convolution_index_tuples=findOperationIndexPairs(idx, 5, False)
    correlation_index_tuples=findOperationIndexPairs(idx, 5, True)
    print()
    print("Indices pairs of convolution operation: \n\n",
convolution_index_tuples)
    print("\n\nIndices pairs of correlation operation: \n\n",
correlation_index_tuples)

if __name__ == "__main__":
    main()
```

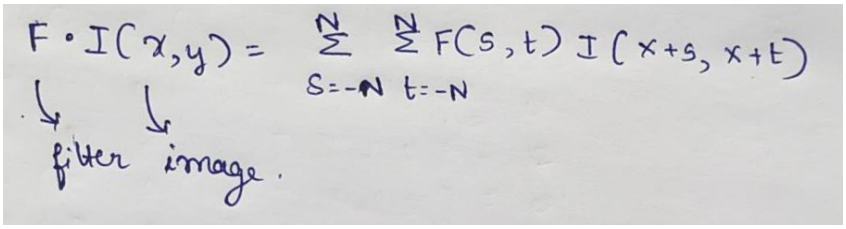
Part A. Show the output for a 5x5 filter at image index of your choice, for cross-correlation and convolution:

OUTPUT :

```
Indices pairs of convolution operation:

 [(('img', 3, 3), ('kernel', 4, 4)), (('img', 4, 3), ('kernel', 3, 4)), (('img', 5, 3), ('kernel', 2, 4)), (('img', 6, 3), ('ke
rnel', 1, 4)), (('img', 7, 3), ('kernel', 0, 4)), (('img', 3, 4), ('kernel', 4, 3)), (('img', 4, 4), ('kernel', 3, 3)), (('im
g', 5, 4), ('kernel', 2, 3)), (('img', 6, 4), ('kernel', 1, 3)), (('img', 7, 4), ('kernel', 0, 3)), (('img', 3, 5), ('kernel',
4, 2)), (('img', 4, 5), ('kernel', 3, 2)), (('img', 5, 5), ('kernel', 2, 2)), (('img', 6, 5), ('kernel', 1, 2)), (('img', 7,
5), ('kernel', 0, 2)), (('img', 3, 6), ('kernel', 4, 1)), (('img', 4, 6), ('kernel', 3, 1)), (('img', 5, 6), ('kernel', 2, 1)),
(('img', 6, 6), ('kernel', 1, 1)), (('img', 7, 6), ('kernel', 0, 1)), (('img', 3, 7), ('kernel', 4, 0)), (('img', 4, 7), ('kern
el', 3, 0)), (('img', 5, 7), ('kernel', 2, 0)), (('img', 6, 7), ('kernel', 1, 0)), (('img', 7, 7), ('kernel', 0, 0))]


Indices pairs of correlation operation:

 [(('img', 3, 3), ('kernel', 0, 0)), (('img', 4, 3), ('kernel', 1, 0)), (('img', 5, 3), ('kernel', 2, 0)), (('img', 6, 3), ('ke
rnel', 3, 0)), (('img', 7, 3), ('kernel', 4, 0)), (('img', 3, 4), ('kernel', 0, 1)), (('img', 4, 4), ('kernel', 1, 1)), (('im
g', 5, 4), ('kernel', 2, 1)), (('img', 6, 4), ('kernel', 3, 1)), (('img', 7, 4), ('kernel', 4, 1)), (('img', 3, 5), ('kernel',
0, 2)), (('img', 4, 5), ('kernel', 1, 2)), (('img', 5, 5), ('kernel', 2, 2)), (('img', 6, 5), ('kernel', 3, 2)), (('img', 7,
5), ('kernel', 4, 2)), (('img', 3, 6), ('kernel', 0, 3)), (('img', 4, 6), ('kernel', 1, 3)), (('img', 5, 6), ('kernel', 2, 3)),
(('img', 6, 6), ('kernel', 3, 3)), (('img', 7, 6), ('kernel', 4, 3)), (('img', 3, 7), ('kernel', 0, 4)), (('img', 4, 7), ('kern
el', 1, 4)), (('img', 5, 7), ('kernel', 2, 4)), (('img', 6, 7), ('kernel', 3, 4)), (('img', 7, 7), ('kernel', 4, 4))]
```

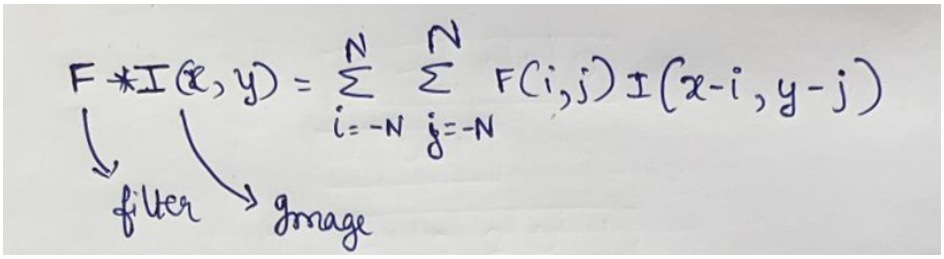
Part B. Explain the outputs with respect to the formulation/equation of 2D cross-correlation and convolution.

Cross-correlation:
The formula of the cross-correlation is given by:



Hence, For every value in the n*n sized image patch, the values in the corresponding index in the filter has to be multiplied (where filter size is n*n). This is shown in both the code as well as the respective output. Here, the input pixel index for **idx** parameter is (5, 5). Hence, the process starts from (3, 3) pixel of image and corresponding index in the filter kernel i.e. (0,0). Hence, it goes from range 3 to 7 with respect to image rows and columns covering the image patch of width and length of 5 and is the size of the filter as well that ranges from 0 to 4 in the loop.

Convolution:
The formula of the Convolution is given by:

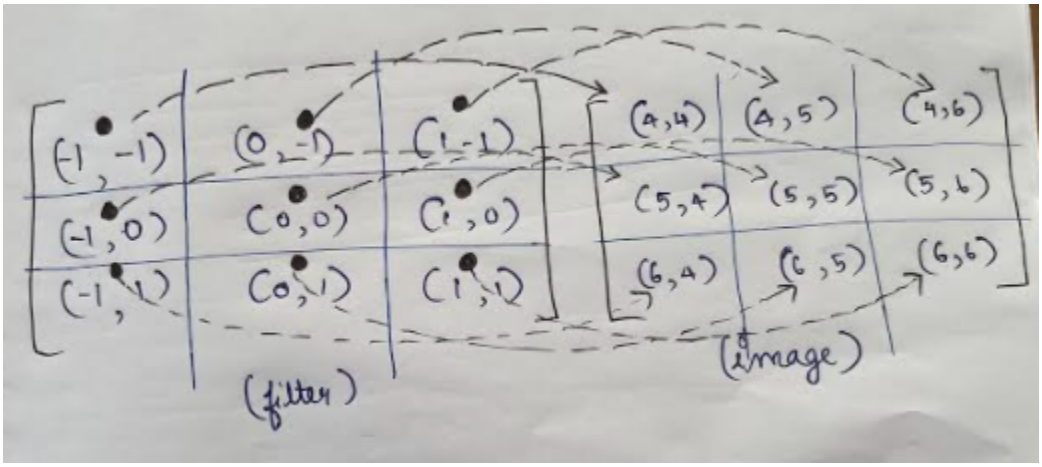


Hence, For every value in the image patch of size (n*n), the values in the corresponding flipped kernel's index in the filter has to be multiplied (where filter size is n*n). This is shown in both the code as well as the respective output. Here, the input pixel index for **idx** parameter is (5, 5). Hence, the process starts from (3, 3) pixel of image and corresponding index in the filter kernel i.e. (4,4). Hence, it goes from range 3 to 7 with respect to image rows and columns covering the image patch of width and length of 5 and is the size of the filter as well that ranges from 4 to 0 in the loop.

Part C. Include a drawing/sketch of your indexing example for cross-correlation and convolution. For example, you could sketch the grid of the image and filter and use numbers or colors to indicate indexing.

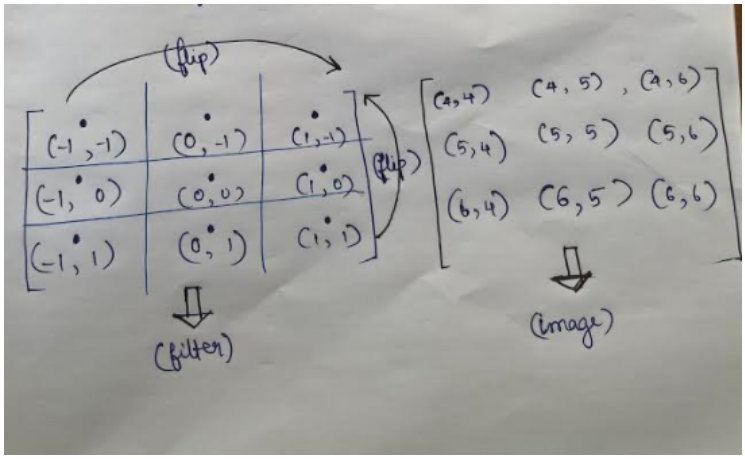
Cross-Correlation:

Here, for correlation, the pixel values in the image has to be multiplied by the values in the corresponding index in the filter. This is illustrated below using 3*3 filter and 3*3 image patch. The center of the image is (5,5) and similarly, the center of the pixel is (0, 0). Hence, the neighborhood pixels of the center image pixel is multiplied by the corresponding neighborhood of the center of the filter kernel.



Convolution:

In the case of Convolution, the filter has to be flipped from both top to down and left to right. In the other sense, the filter has to be rotated completely by 180 degrees. This is illustrated by using the same filter size and the image size as shown earlier (3*3 each) in the case of cross-correlation. (it is shown below).



This results in the following filter kernel and hence is normally multiplied and all the multiplied values are added further hence, leading to the convoluted value of this patch and filter.