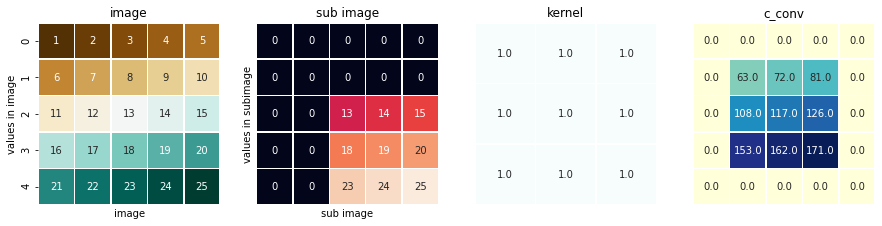
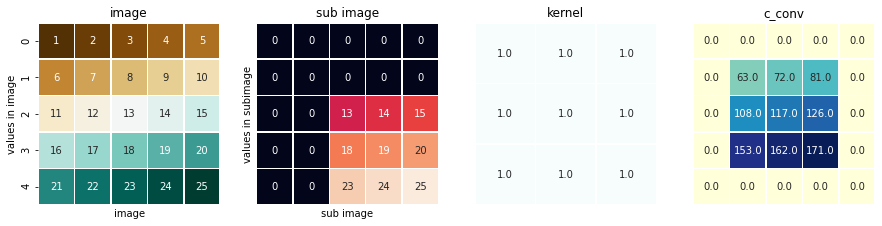
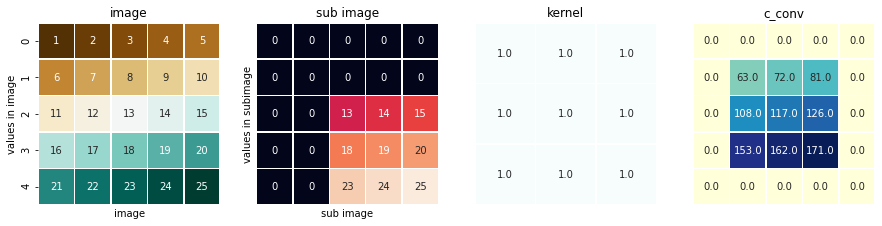
CV (Spring 2021)

Name:

Project 02. Convolution and Window-based Operations with Python/OpenCV

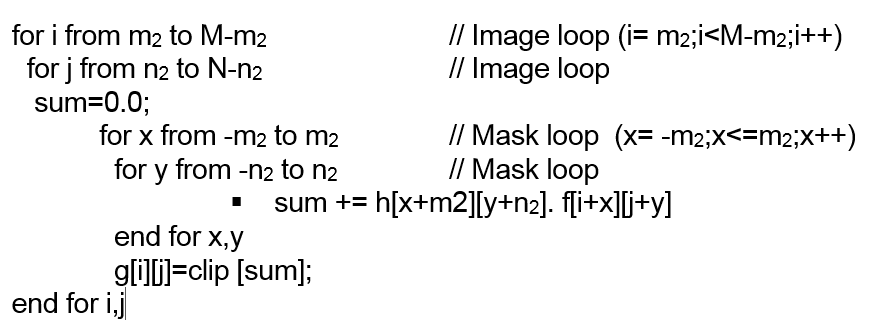
**Introduction:**

* + 1. Convolution and window operations are at the heart of DIP and many Computer vision operations. An example of convolution is shown below.



A general algorithm to implement convolution is given as follows:

* + 1. **Computer algorithms to implement convolution of an odd-size mask**
    2. For an image of size M x N
    3. Create a two-dimensional array h (the mask) of size m x n
    4. Fill h with the mask coefficients.
    5. Let m2=floor (m/2) & n2=floor (n/2)



**Project:**

Your goal in this project is three-fold:

1. To implement the above convolution algorithm (adjusted to Python strengths such as arrays, slicing, etc.). Once implemented, you will apply it on a noisy image (Gaussian noise of your choice of mean and variance) **(V)** with the following masks:

* Unweighted average of 5x5 **(V)**
* Weighted average of 5x5 **(V)**
* Gaussian mask of 5x5 **(V)**

1. To implement Median filtering 3x3 and 5x5) on a noisy image. **(V)** This can be done by finding the window of pixels around the pixel to be changed, sort them, and select the median value as the replacement of that pixel. Apply the algorithm on an impulsive noise of 10% p probability of noise equally divided among salt and pepper.
2. To implement the Sobel Edge extractor based on the following formula: **(V)**



Apply the algorithm on an a clean image with no noise.

* Apply your algorithms on a ***gray-level*** image. Although not required, you can also apply it on an RGB, by calling the function 3 times (R, G, B) and combine the results. OpenCV implementation seems to detect the number of channels and apply the convolution accordingly.

Here is a quick implementation of the different types of noise needed for this project.

**(V)**

**def add\_gaussian\_noise(mean,variance,image):**

row,col,ch= image.shape

st\_dev = math.sqrt(variance)

gauss\_noise = np.random.normal(mean,st\_dev,(row,col,ch))

gauss\_noise = gauss\_noise.reshape(row,col,ch)

noisy = image + gauss\_noise

noisy = noisy.clip(0,255).astype(np.uint8)

show\_image(noisy)

return noisy

**(V)**

**def add\_saltpepper\_noise(image,p=0.10, s\_vs\_p=0.50):**

row,col,ch = image.shape

image\_plus\_noise = np.copy(image)

# Salt mode

num\_salt = np.ceil(p \* image.size \* s\_vs\_p)

coords = [np.random.randint(0, i - 1, int(num\_salt)) for i in image.shape]

image\_plus\_noise[coords] = 255

# Pepper mode

num\_pepper = np.ceil(p\* image.size \* (1. - s\_vs\_p))

coords = [np.random.randint(0, i - 1, int(num\_pepper)) for i in image.shape]

image\_plus\_noise[coords] = 0

image\_plus\_noise = image\_plus\_noise.clip(0,255).astype(np.uint8)

return image\_plus\_noise

In your report, please have your implementation compared with that of OpenCV. I showed examples in the class on how to do it with OpenCV with 2-3 lines of code.

In all the above operations, you can ignore the boundaries by either leaving them at 0 (will show as black frame around the image) or copying them from the input image (more visually appealing). Remember that you have to create an empty output image, g(x,y), different from the input image, f(x,y), to store the result of your operation.

**Grading and Submission Guide:**

* Must submit the whole project (python folder with code, image dataset, and results) zipped using 7zip tools with the name: LastName\_FirstName\_Project-02.
* For this project, put a sample of screen shots of your program run in the report.
* This is an **individual** project: The work should represent your own: that you acknowledge that have not incorporated into this project any unacknowledged material from the work of another person, including papers, words, ideas, information, computer code, data, evidence-organizing principles, or style of presentation taken from the Internet, books, periodicals, or other sources.