

## Lab 6 Arbitrary Convolution Filter

1. หาภาพ อะไรก็ได้ที่มีวัตถุอยู่ในภาพ เพื่อนำมาเป็น Input ของ Lab โหลดภาพและแสดงบน Jupyter/Colab


```
import Libs

import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

[1] ✓ 0.4s

original_img = cv.imread('output.png')
RGB_img = cv.cvtColor(original_img, cv.COLOR_BGR2RGB)
plt.imshow(RGB_img)
plt.show()

[2] ✓ 0.3s
```



2. สร้าง Filter avg 3x3 5x5 7x7 9x9 11x11 ด้วย Numpy Array ที่เป็น Low Pass Filter

```
result_avg3x3_img = cv.filter2D(RGB_img, -1, kernel_avg3x3)
result_avg5x5_img = cv.filter2D(RGB_img, -1, kernel_avg5x5)
result_avg7x7_img = cv.filter2D(RGB_img, -1, kernel_avg7x7)
result_avg9x9_img = cv.filter2D(RGB_img, -1, kernel_avg9x9)
result_avg11x11_img = cv.filter2D(RGB_img, -1, kernel_avg11x11)

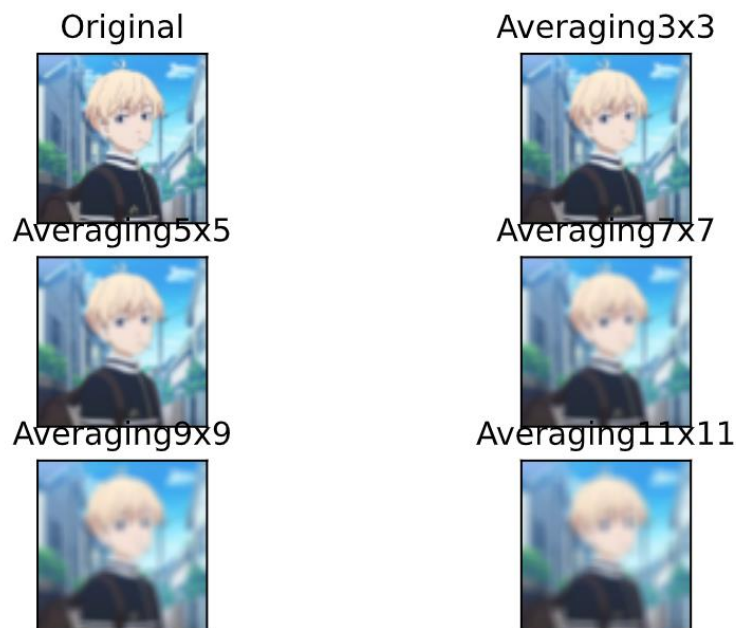
✓ 0.2s
```

### 3. นำ 2D Average Filter มาทำ Convolution กับภาพ และแสดงผลลัพธ์

#### 3.1. Code

```
#subplot(จำนวนแถวทั้งหมด จำนวนคอลัมน์ทั้งหมด ตำแหน่งในตาราง เรียงจากซ้ายไปขวา บนไปล่าง)
plt.subplot(321),plt.imshow(RGB_img),plt.title('Original')
plt.xticks([], plt.yticks([]))
plt.subplot(322),plt.imshow(result_avg3x3_img),plt.title('Averaging3x3')
plt.xticks([], plt.yticks([]))
plt.subplot(323),plt.imshow(result_avg5x5_img),plt.title('Averaging5x5')
plt.xticks([], plt.yticks([]))
plt.subplot(324),plt.imshow(result_avg7x7_img),plt.title('Averaging7x7')
plt.xticks([], plt.yticks([]))
plt.subplot(325),plt.imshow(result_avg9x9_img),plt.title('Averaging9x9')
plt.xticks([], plt.yticks([]))
plt.subplot(326),plt.imshow(result_avg11x11_img),plt.title('Averaging11x11')
plt.xticks([], plt.yticks([]))
```

#### 3.2. result



อธิบายผลลัพธ์ : จะให้ภาพ Averaging 3x3 มีความเบลอน้อยแต่ถ้าเป็น Averaging 11x11 เบลอมาก ๆ ทำให้สรุปได้ว่ายิ่งฟิลเตอร์มีขนาดยิ่งใหญ่ ภาพก็จะยิ่งเบลอ

#### 4. ทดลองสร้าง Sharpen Filter 3x3 7x7 ที่เป็น High Pass Filter ด้วย Numpy Array

```
kernel_sharpen3x3 = np.array([[0,-1,-0],[-1,5,-1],[0,-1,0]],np.float32) # sharpen 3x3
kernel_sharpen7x7 = np.full((7,7),-1,np.float32) # sharpen 7x7
kernel_sharpen7x7[3,3] = 49 # sharpen 7x7
print(kernel_sharpen3x3)
print(kernel_sharpen7x7)
```

✓ 0.3s

```
[[ 0. -1.  0.]
 [-1.  5. -1.]
 [ 0. -1.  0.]
 [-1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. 49. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.]
 [-1. -1. -1. -1. -1. -1. -1.]]
```

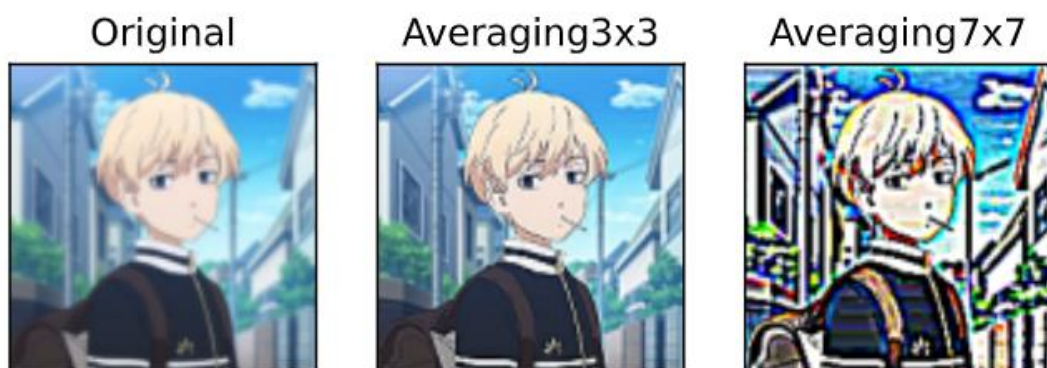
#### 5. นำ 2D Sharpen Filter มาทำ Convolution กับภาพ และแสดงผลลัพธ์

##### 5.1. Code

```
result_sharpen3x3_img = cv.filter2D(RGB_img,-1,kernel_sharpen3x3)
result_sharpen7x7_img = cv.filter2D(RGB_img,-1,kernel_sharpen7x7)
```

✓ 0.3s

##### 5.2. Result



**อธิบายผลลัพธ์ :** จะให้ภาพ Averaging 3x3 มีความคมชัดน้อยแต่ถ้าเป็น Averaging 7x7 ภาพจะคมมาก ๆ จนทำให้ภาพเปลี่ยนจากภาพต้นฉบับไปมาก ทำให้สรุปได้ว่ายิ่งฟิลเตอร์มีขนาดยิ่งใหญ่ ภาพก็จะมี ความคมชัดมากขึ้น

6. ให้นักศึกษา หา Filter ใดๆก็ได้มาทำการ Convolution กับภาพ แสดงผลลัพธ์ และอธิบายผลลัพธ์ที่เกิดขึ้นในJupyter/Colab ได้เลย

### 6.1. Code

```
ลองทำเอง
k3x3 = np.array([[5,5,5],[-3,0,-3],[-3,-3,-3]],np.float32) # sharpen 3x3
print(k3x3)

[24] ✓ 0.8s
...
[[ 5.  5.  5.]
 [-3.  0. -3.]
 [-3. -3. -3.]]

result_mykernel_3x3 = cv.filter2D(RGB_img, -1, k3x3)

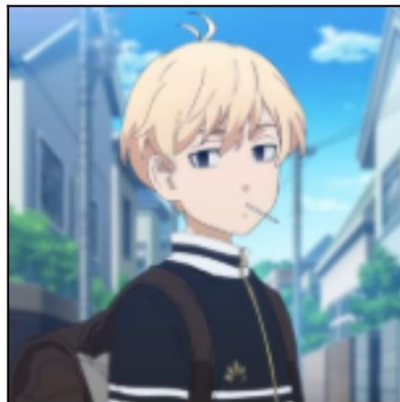
[25] ✓ 0.4s

plt.subplot(121),plt.imshow(RGB_img),plt.title('Original')
plt.xticks([], plt.yticks([]))
plt.subplot(122),plt.imshow(result_mykernel_3x3),plt.title('3x3')
plt.xticks([], plt.yticks([]))

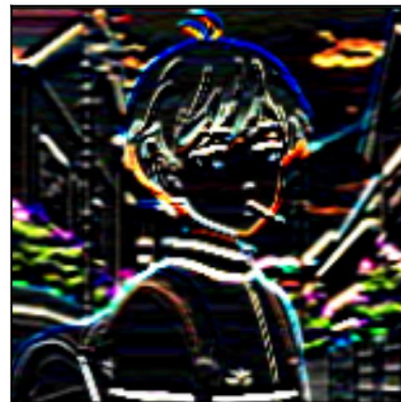
[26] ✓ 0.2s
```

### 6.2. Result

Original



3x3



อธิบายผลลัพธ์ : จากแหล่งข้อมูลของ Filter อธิบายว่า เป็นการสร้างเคอร์เนลในวิธีการไล่ระดับที่ต้องการ เพียง 2 เคอร์เนล และหนึ่งเคอร์เนลที่ไวต่อขอบในทิศทางแนวนอน