Bài giảng môn học:

Thị giác máy tính (7080518)

CHƯƠNG 2:
XỬ LÝ VÀ NÂNG CAO CHẤT LƯỢNG ẢNH
(Phần 3)

Đặng Văn Nam

dangvannam@humg.edu.vn

Nội dung chương 2 – phần 3

II/ Cải thiện chất lượng ảnh (tiếp)

1. Làm mờ - mịn ảnh
   - Làm mịn trung bình
   - Làm mịn trung vị
   - Làm mịn Gaussian
2. Nhiễu và khử nhiễu trong ảnh
3. Giới thiệu Nhân tích chập
4. Nhân tích chập 2D trong OpenCV

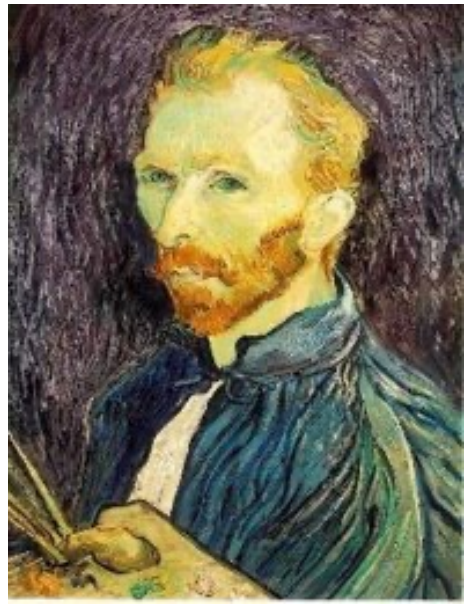
1. Làm mịn – mờ ảnh (blur images)

Giới thiệu

De-noising

Salt and pepper noise

Super-resolution



In-painting

Giới thiệu

Ý nghĩa của việc làm mịn:

- **Hình thành ảnh mới sao cho những giá trị điểm ảnh đạt được hiệu ứng nào đó từ ảnh gốc.**
- **Nhiều thông tin hữu ích sẽ được thu nhận thông qua quá trình lọc như:**
  - Làm nổi bật các đặc trưng trên ảnh: Biên, góc, hình khối….
  - Cải thiện/tăng cường chất lượng ảnh: Khử nhiễu trong ảnh, thay đổi kích thước ảnh…
  - Tạo hiệu ứng: Ảnh độ phân giải cao, sửa ảnh…

a. Lọc trung bình

Lọc trung bình → làm mịn ảnh.

- Thực hiện tính trung bình các pixel lân cận của pixel trung tâm.
- Sử dụng để loại bỏ nhiễu, làm nổi các chi tiết lớn.

23	25	30	35	30
25	30	35	37	40
45	40	37	43	45
38	40	43	42	46
35	40	42	45	47

		39		

Value = (30 + 35+ 37+ 40 + 37+ 43+ 40+ 43+ 42) /9 = 38.55 = 39

Original

$*$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$=$

Blur (with a mean filter)

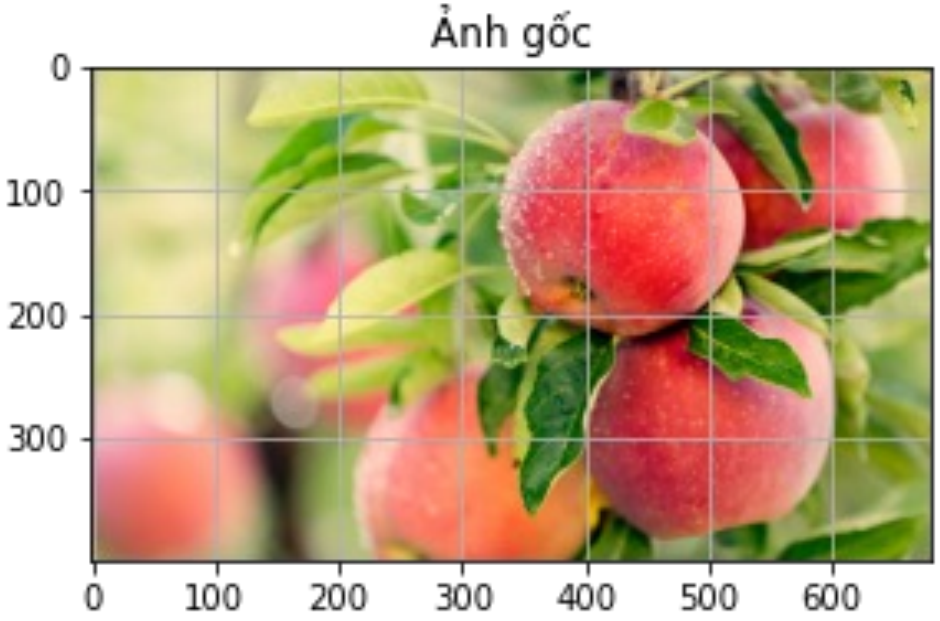
a. Lọc trung bình

Trong Python để lọc trung bình sử dụng phương thức:

cv2.blur(src, kernel)

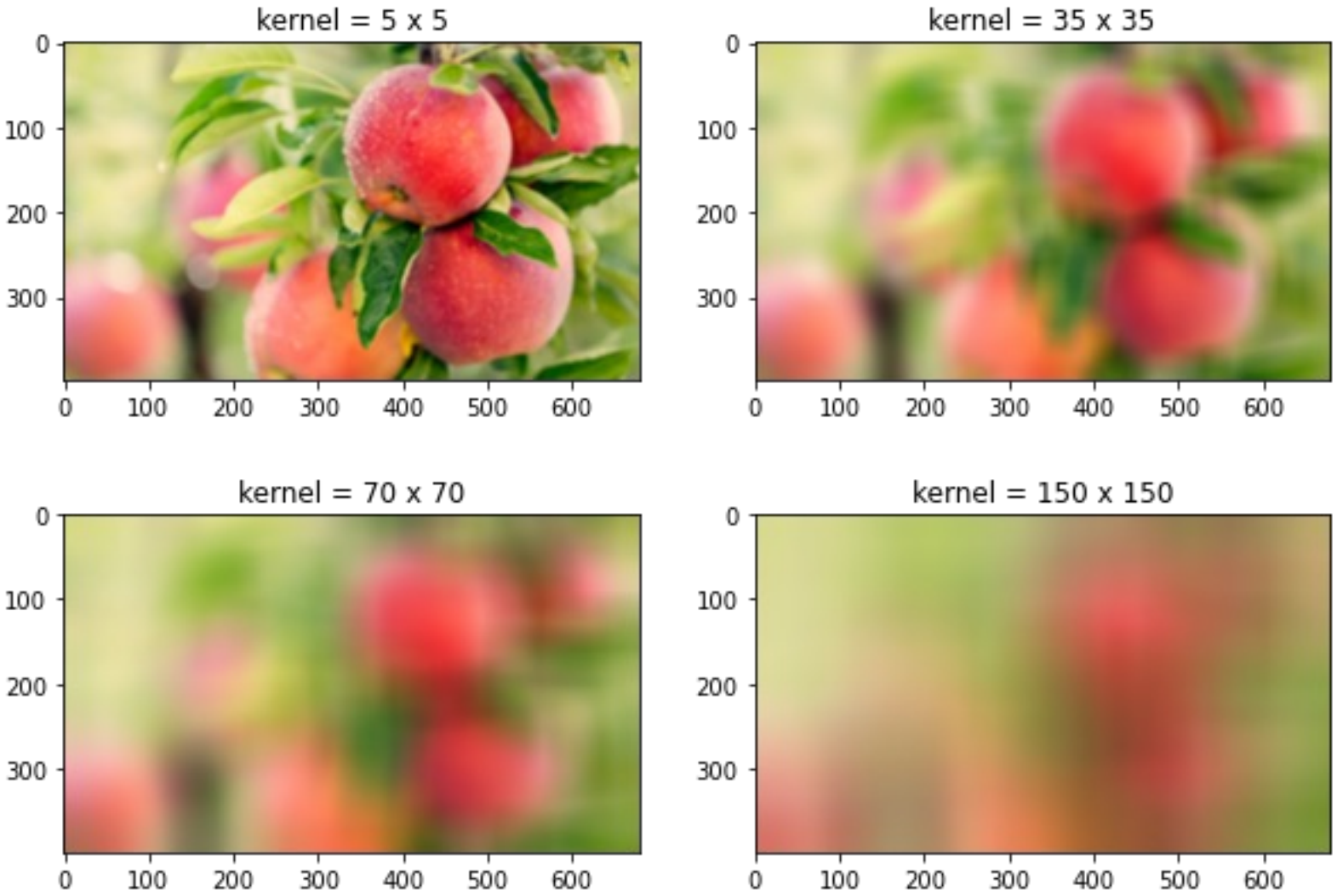
Trong đó:

- src: ảnh gốc ban đầu
- kernel(width,height): Kích thước mặt nạ lọc



a. Lọc trung bình

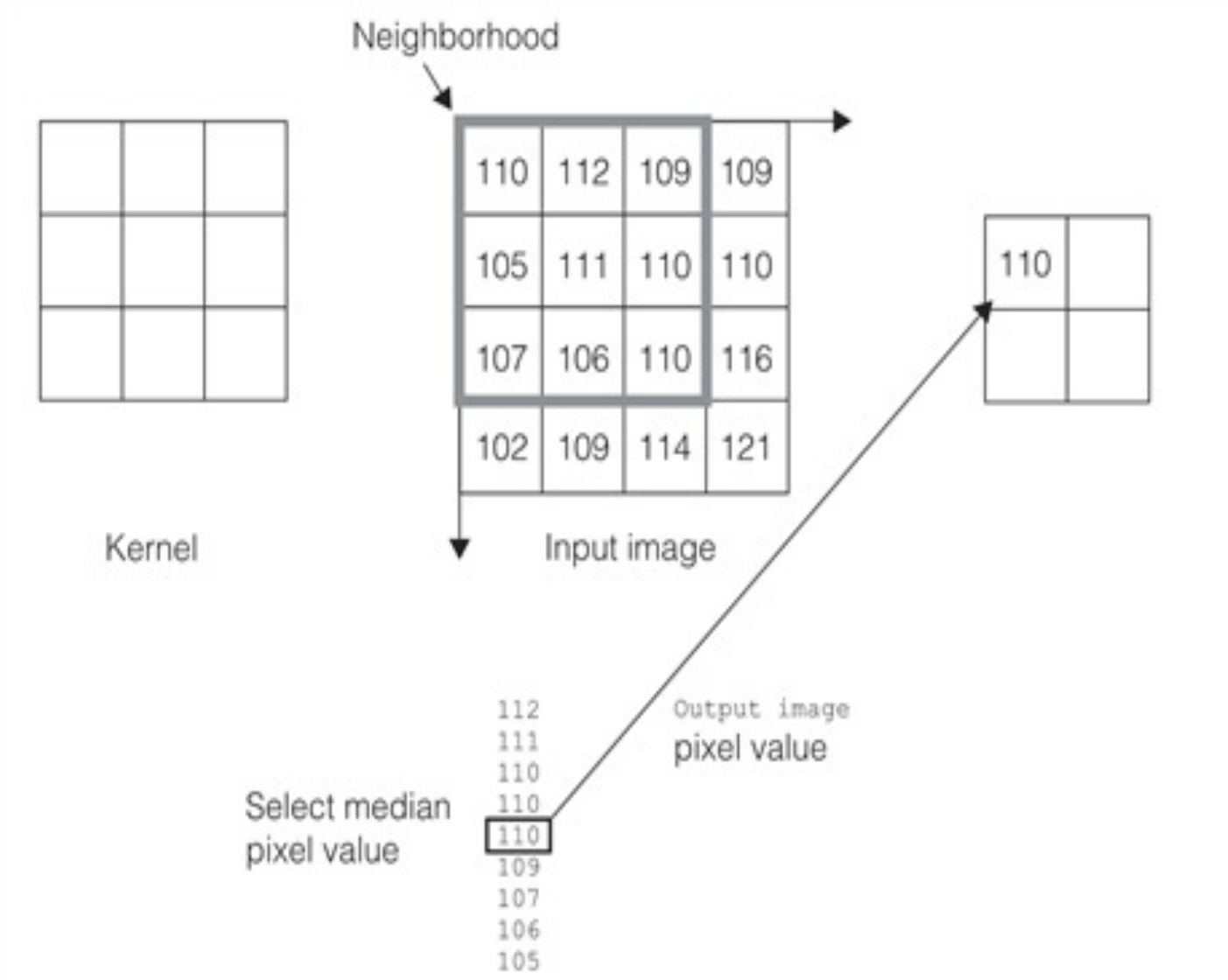
Kích thước mặt nạ lọc càng lớn, ảnh càng mờ:



b. Lọc trung vị

Lọc trung vị.

- Giá trị trung vị X của một tập hợp là giá trị sao cho một nửa giá trị trong tập hợp nhỏ hơn hoặc bằng X và một nửa còn lại lớn hơn hoặc bằng X.
- Thực hiện: Sắp xếp giá trị của các pixel trong vùng lân cận (có kích thước bằng kích thước bộ lọc). Xác định giá trị trung vị của chúng và gán giá tri cho pixel trong ảnh được lọc



b. Lọc trung vị

Trong Python để lọc trung vị sử dụng phương thức:

cv2.medianBlur(src, ksize)

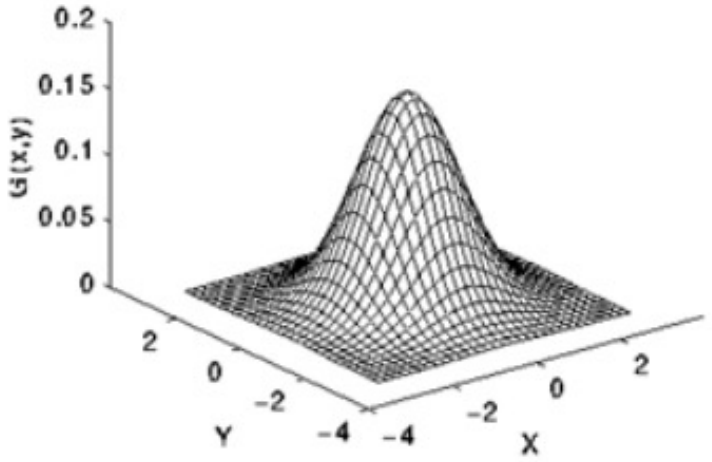
Trong đó:

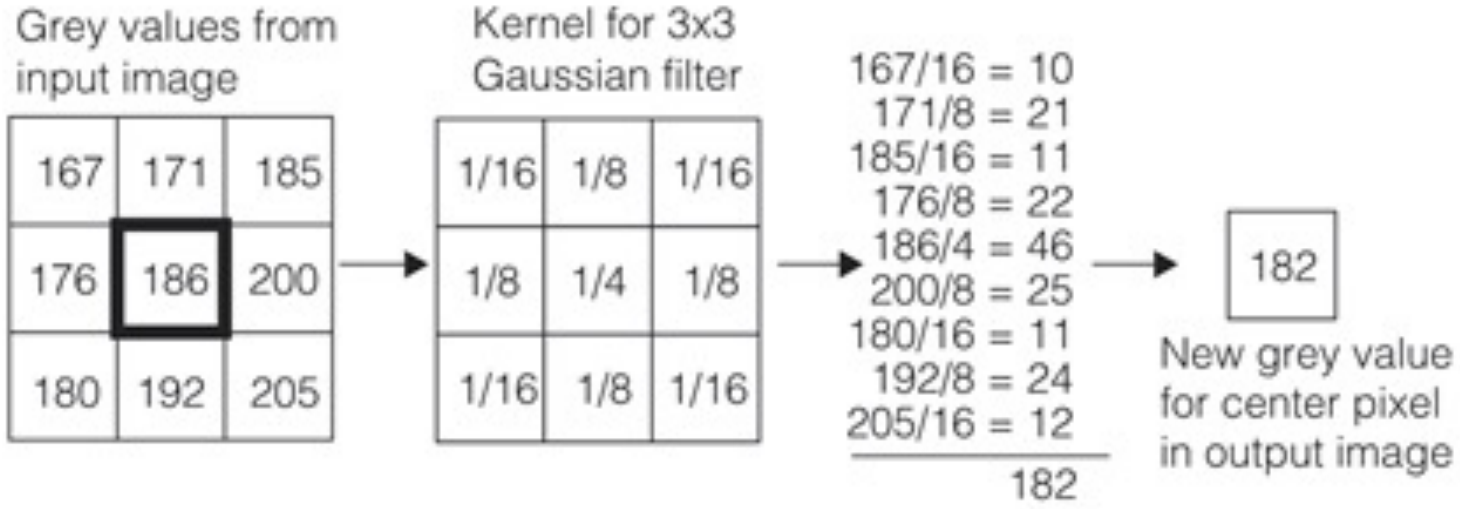
- src: ảnh gốc ban đầu
- ksize: Kích thước mặt nạ lọc, một giá trị phải là một số lẻ lớn hơn 1 (vd: 3, 5, 7…)

c. Lọc Gaussian

Lọc Gaussian.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





c. Lọc Gaussian

Trong Python để lọc Gaussian sử dụng phương thức:

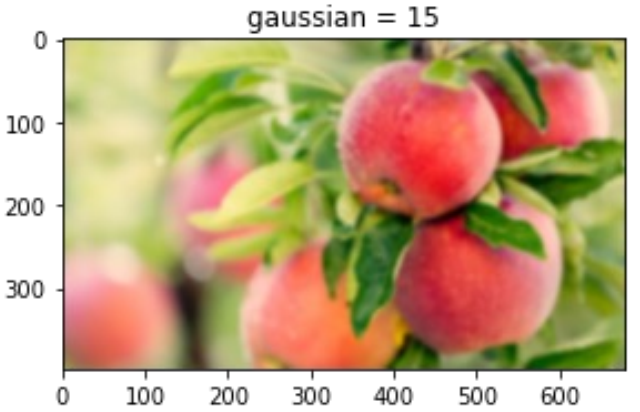
cv2.gaussianBlur(src, kernel, sigmaX, sigmaY)

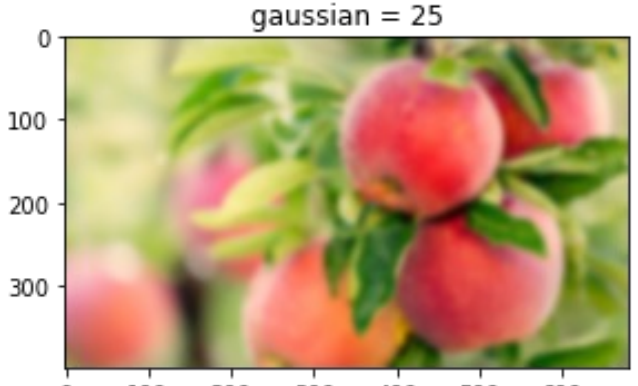
Trong đó:

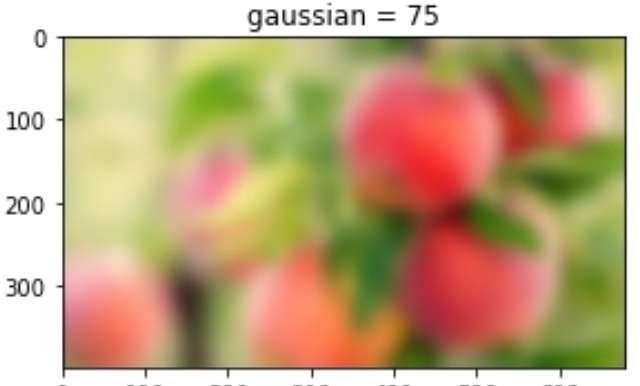
- src: ảnh gốc ban đầu
- kernel: Kích thước mặt nạ lọc, phải là số dương lẻ.
- sigmaX, sigmaY: độ lệch chuẩn của kernel theo hướng X, Y

c. Lọc Gaussian

```python
#Làm mịn ảnh với các kernel khác nhau sử dụng bộ lọc Gaussian:
#kernel = 5 x 5:
gaussian_5 = cv2.GaussianBlur(img_orignal,(5,5),0)

#kernel = 15 x 15:
gaussian_15 = cv2.GaussianBlur(img_orignal,(15,15),0)

#kernel = 25 x 25:
gaussian_25 = cv2.GaussianBlur(img_orignal,(25,25),0)

#kernel = 75 x 75:
gaussian_75 = cv2.GaussianBlur(img_orignal,(75,75),0)
```



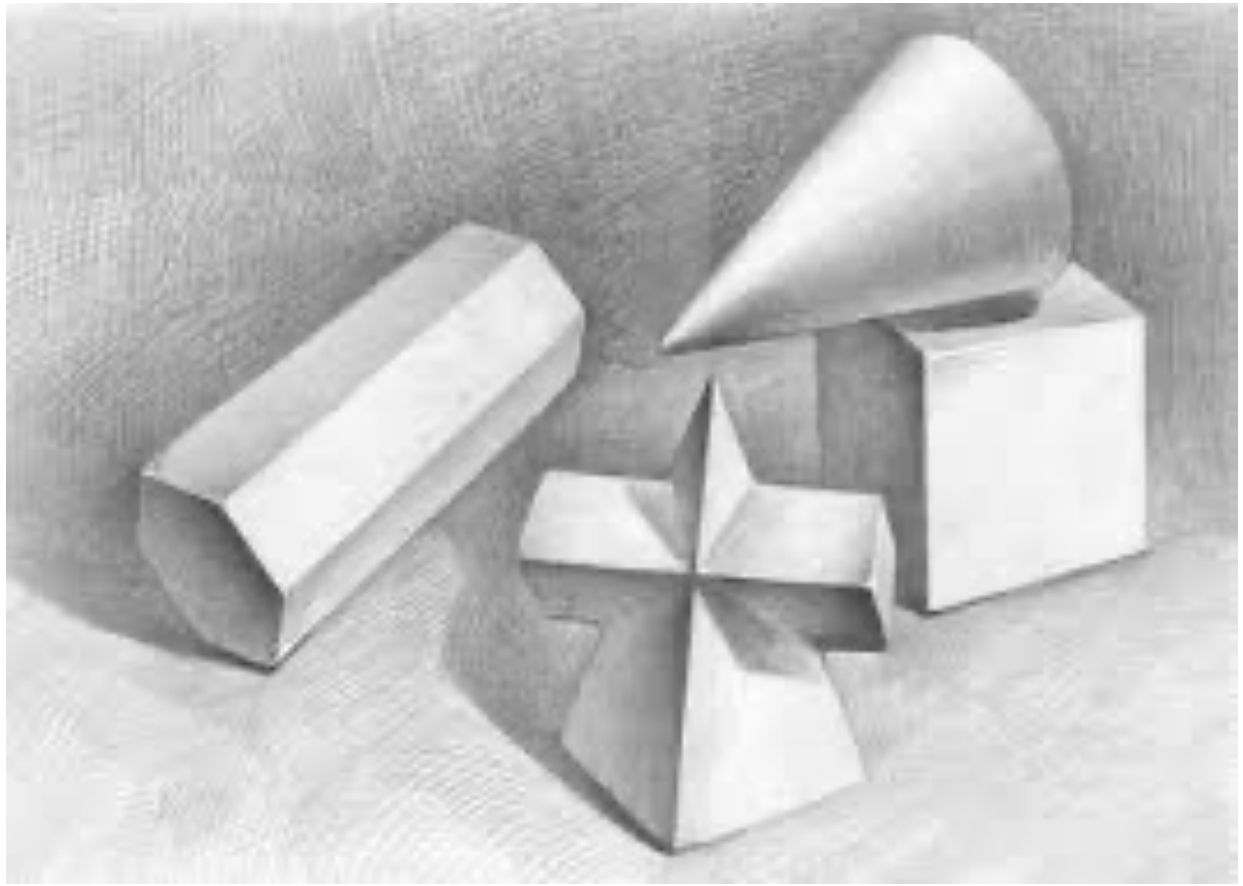




Thực hành số 2.8

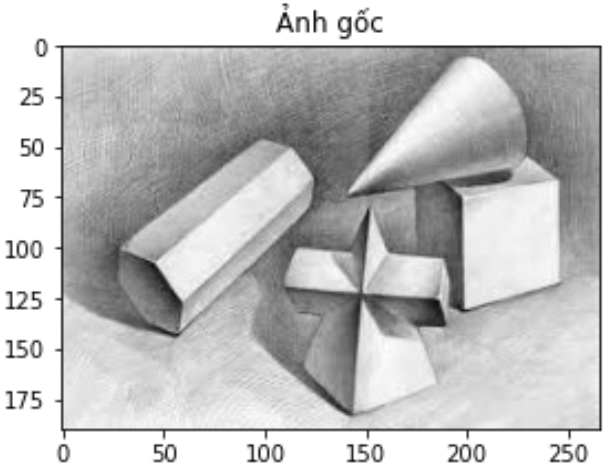
Thực hành 2.8

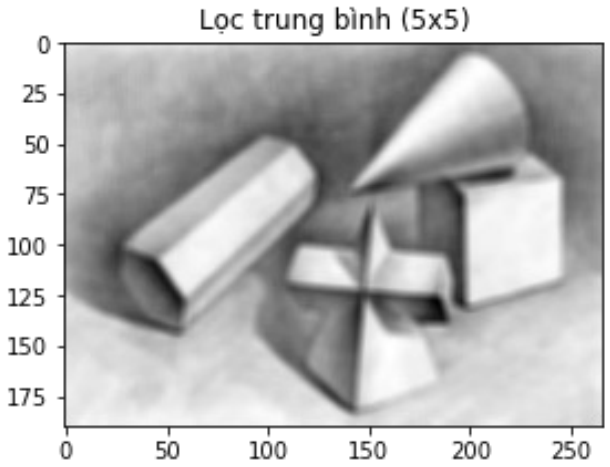
Yêu cầu 1: Sinh viên đọc ảnh images/Thuchanh2_8.jpeg và hiển thị ảnh

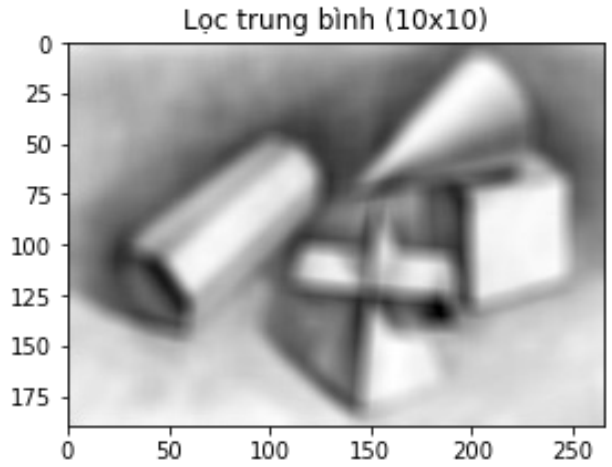


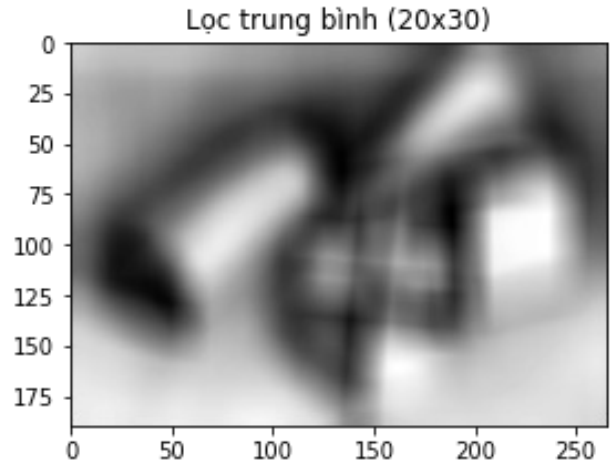
Thực hành 2.8

Yêu cầu 2: Sử dụng phương pháp làm mịn trung bình với kernel 5x5, 10x10, 20x30 và hiển thị kết quả



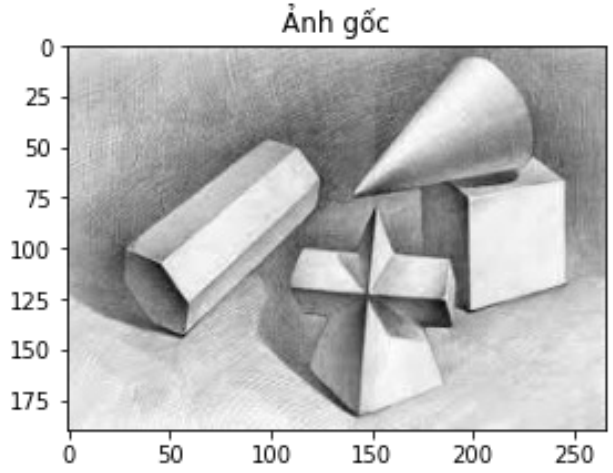


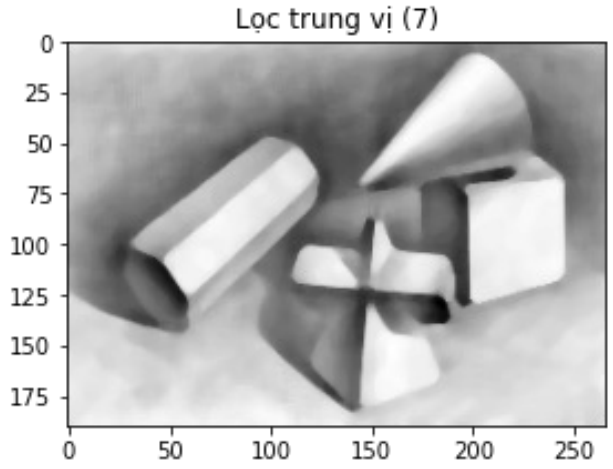


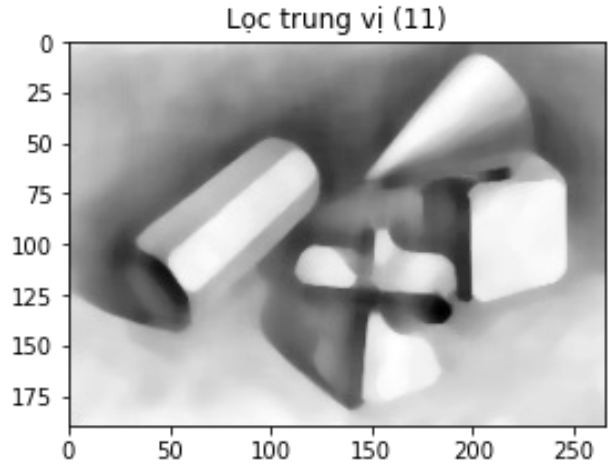


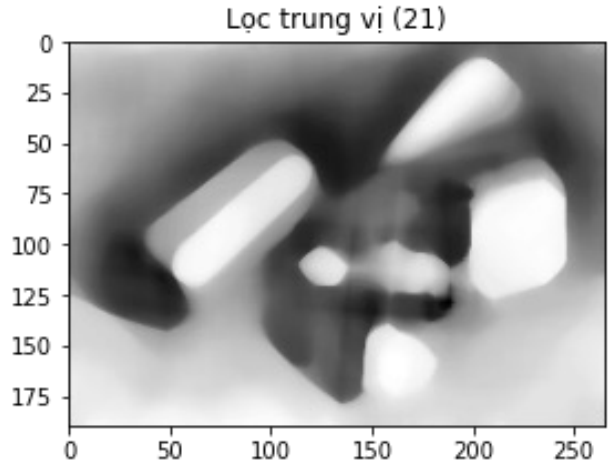
Thực hành 2.8

Yêu cầu 3: Sử dụng phương pháp làm mịn trung vị với k = 7, 11, 21 và hiển thị kết quả



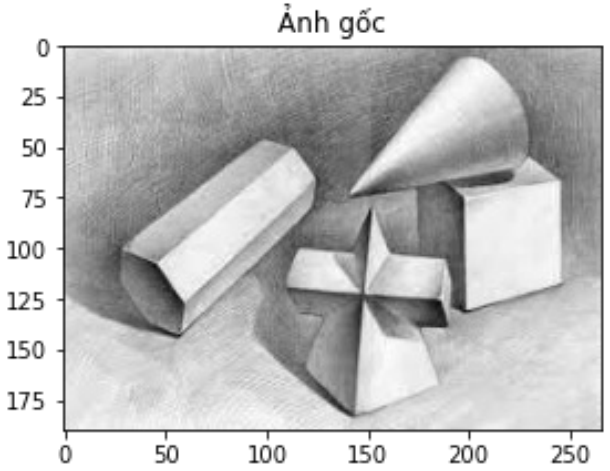


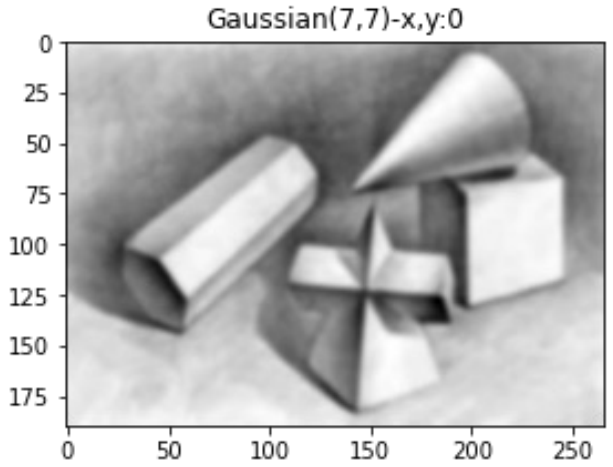


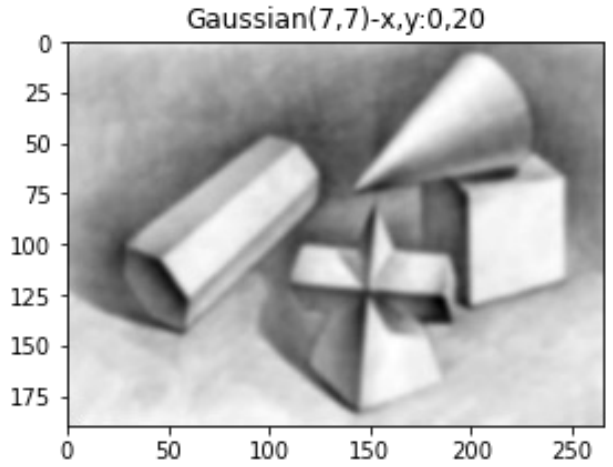


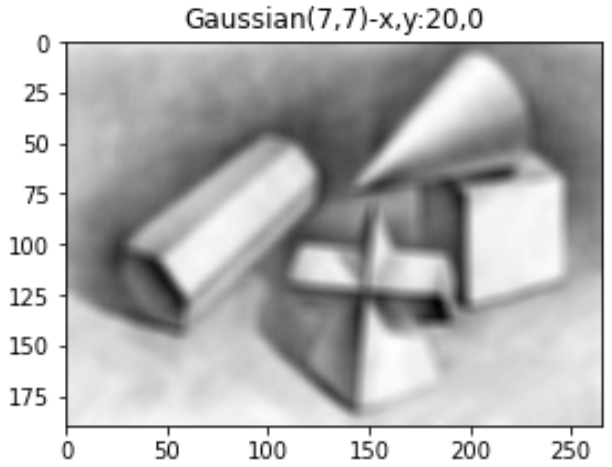
Thực hành 2.8

Yêu cầu 4: Sử dụng phương pháp làm mịn Gaussian với kernel = (7,7) và sigmaX, sigmaY lần lượt là (0,0); (0,20); (20,0) và hiển thị kết quả









2. Nhiễu và khử nhiễu trong ảnh

Nhiễu trong ảnh là gì?

Khái niệm: Nhiễu xuất hiện trong quá trình thu nhận ảnh, số hoá và truyền ảnh.

- Cảm biến ảnh có thể bị ảnh hưởng bởi các điều kiện môi trường.
- Nhiễu có thể can thiệp vào ảnh trong quá trình truyền ảnh.
- Lượng tử hoá
- Số hoá.

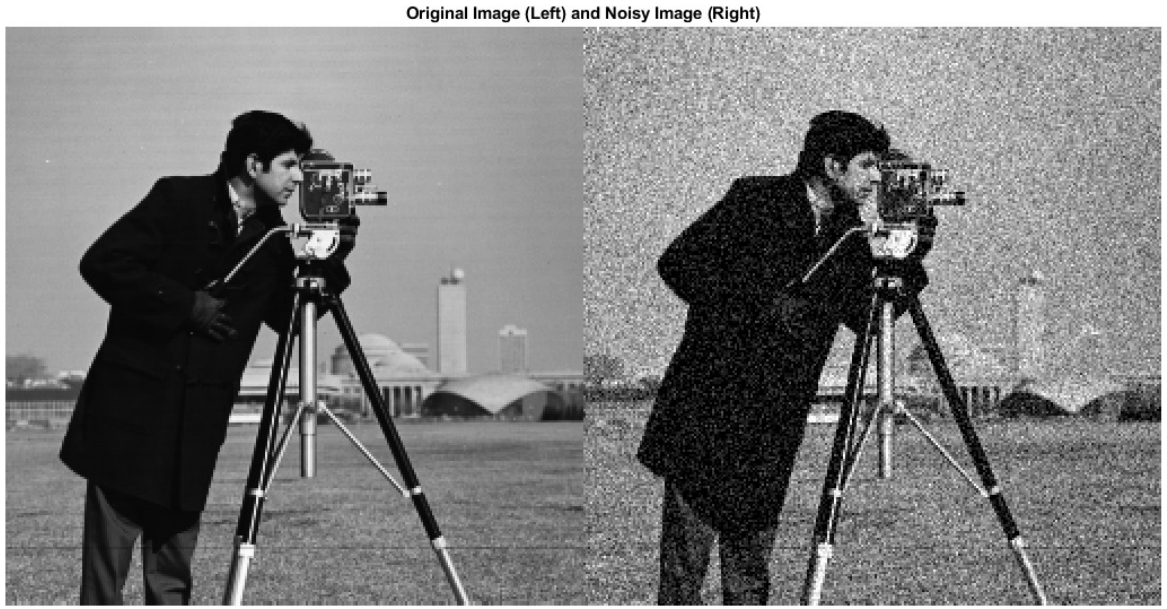
Nhiễu trong ảnh là gì?

Ảnh nhiễu được biểu diễn bằng biểu thức:

$$g(x,y) = f(x,y) + \alpha(x,y)$$

Trong đó:

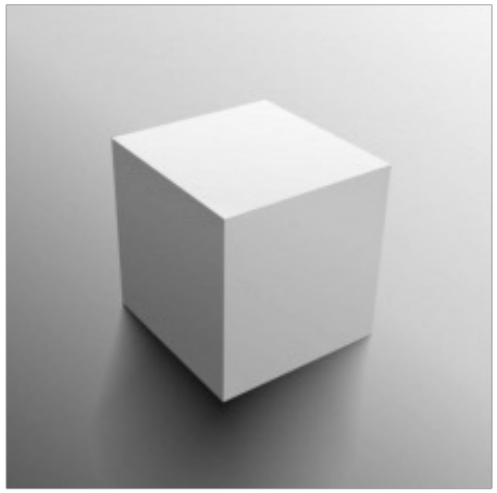
- f(x,y): Ảnh gốc
- α(x,y): Nhiễu
- g(x,y): Ảnh sau khi bị nhiễu tác động.



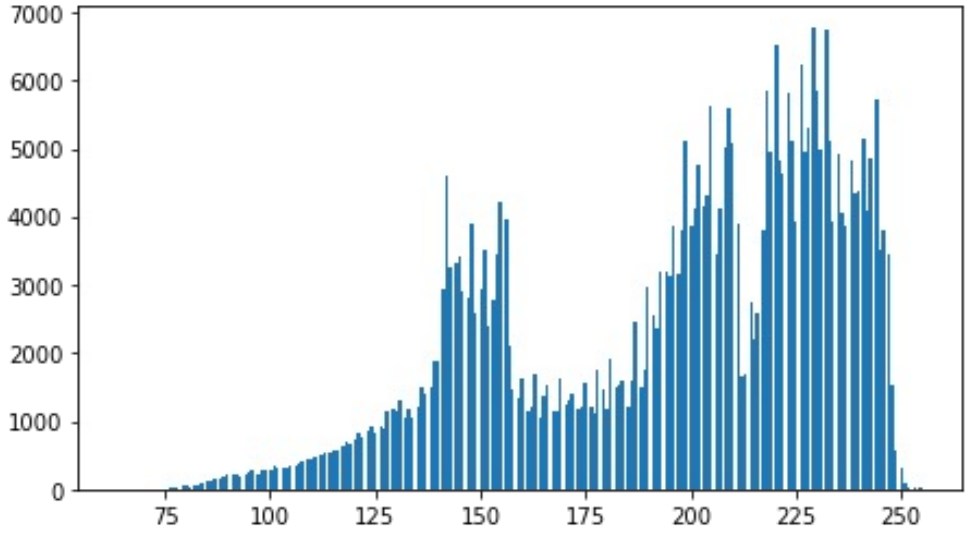
Một số loại nhiễu trong ảnh

1. Nhiễu gaussian

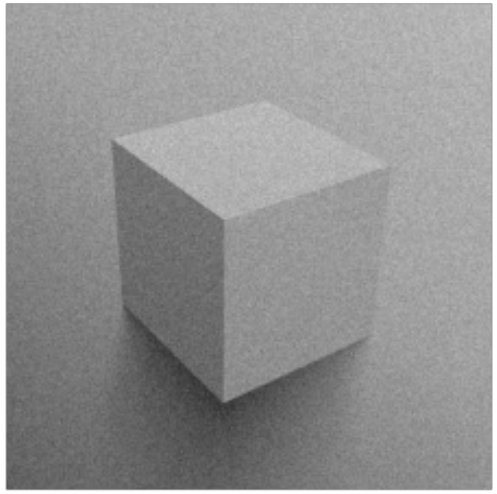
Ảnh gốc



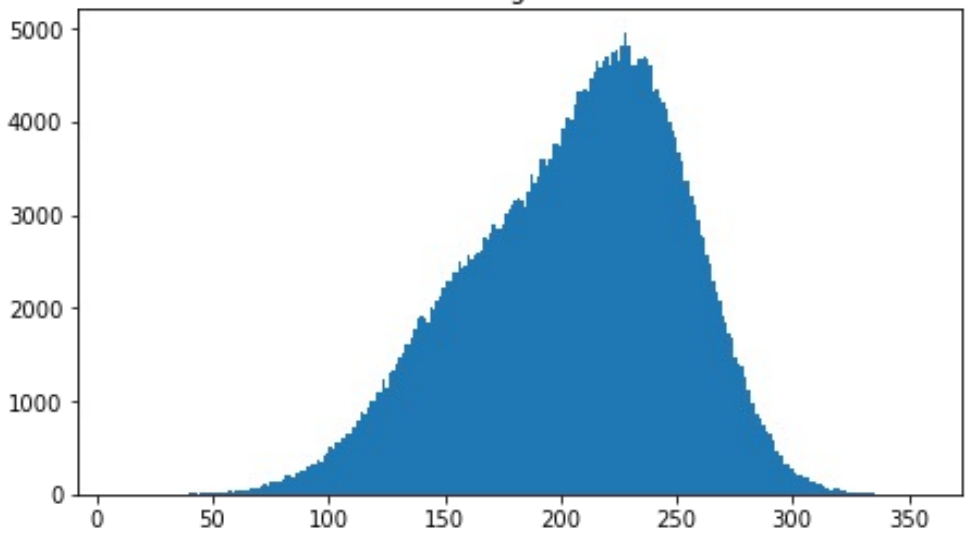
Histogram



Ảnh nhiễu Gaussian



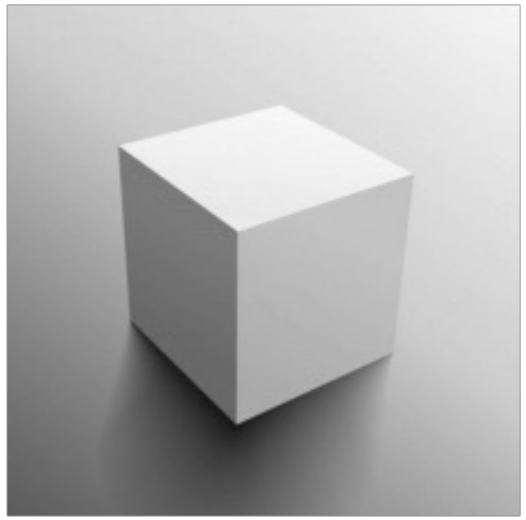
Hitogram



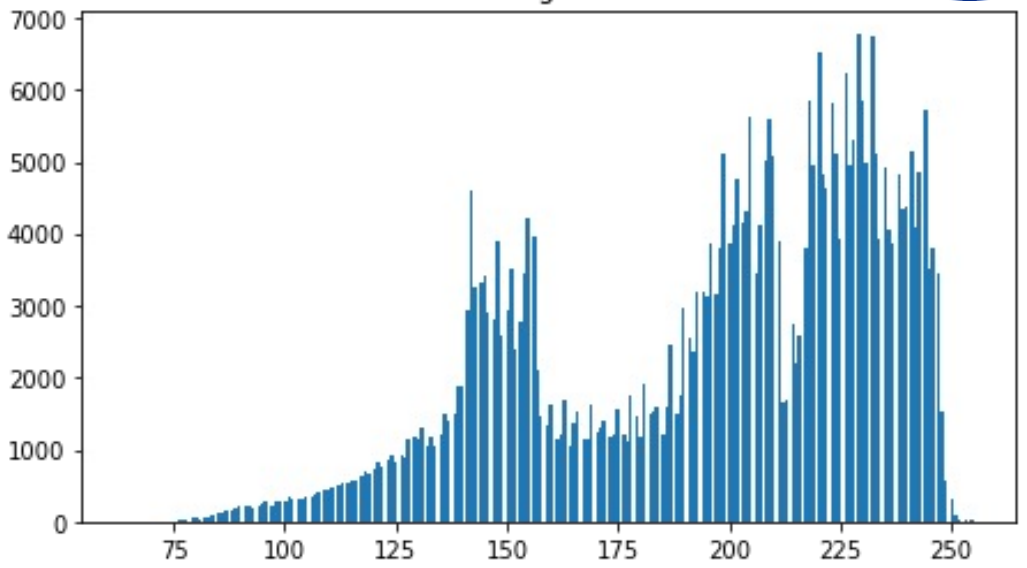
Một số loại nhiễu trong ảnh

2. Nhiễu Rayleigh

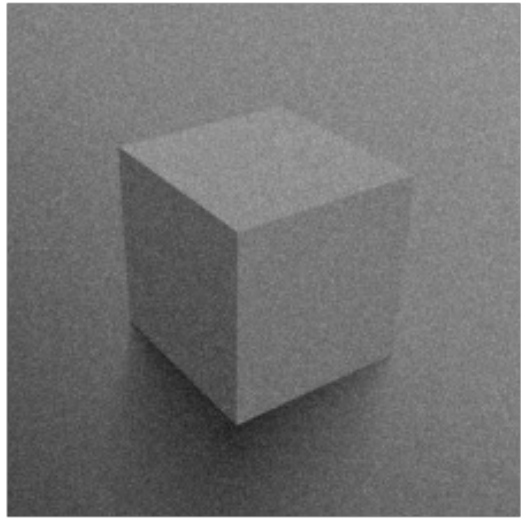
Ảnh gốc



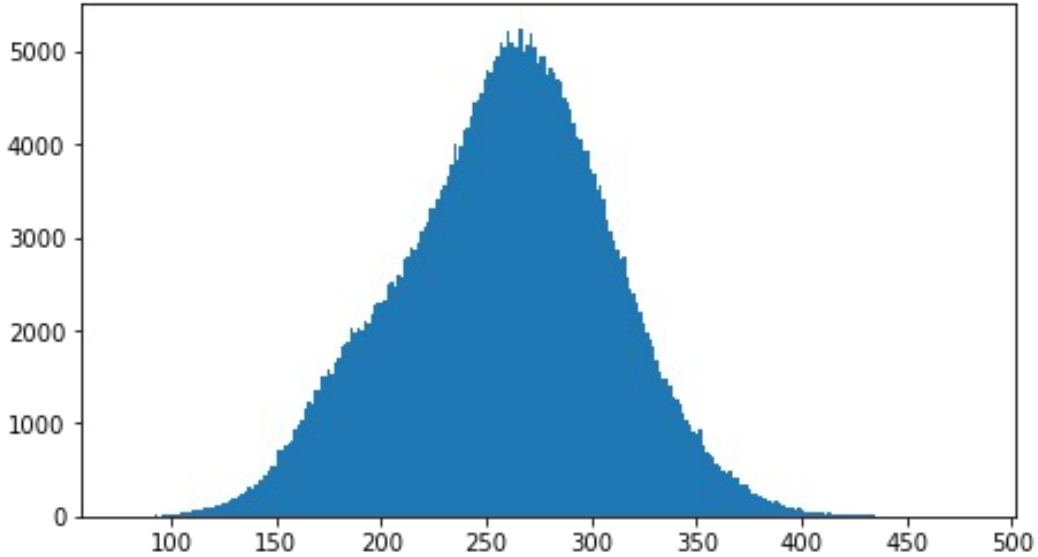
Hitogram



Ảnh nhiễu Rayleigh



Histogram

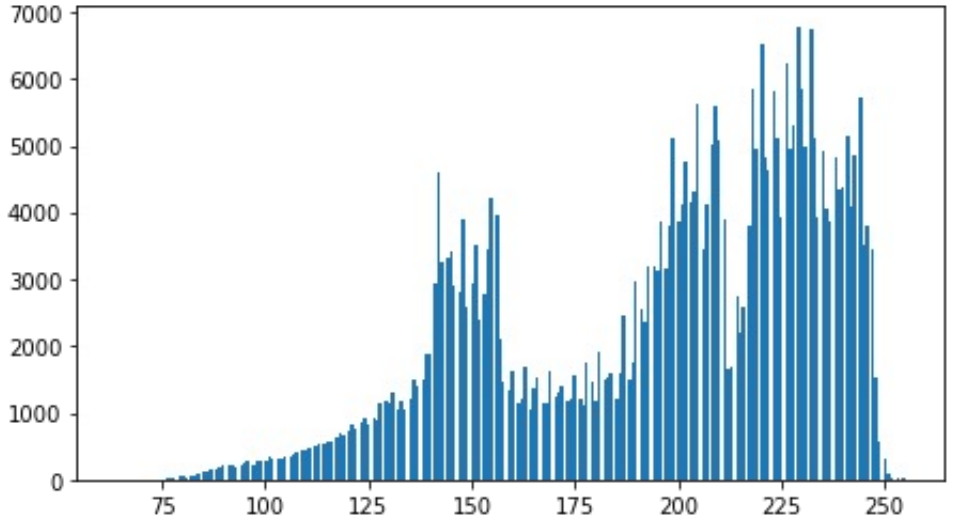


Một số loại nhiễu trong ảnh

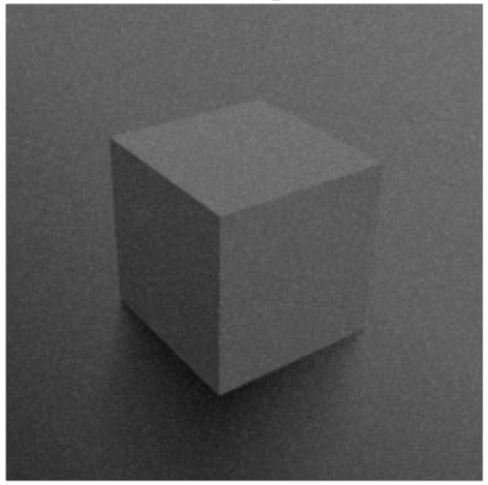
3. Nhiễu Erlang (Gammar)

Ảnh gốc

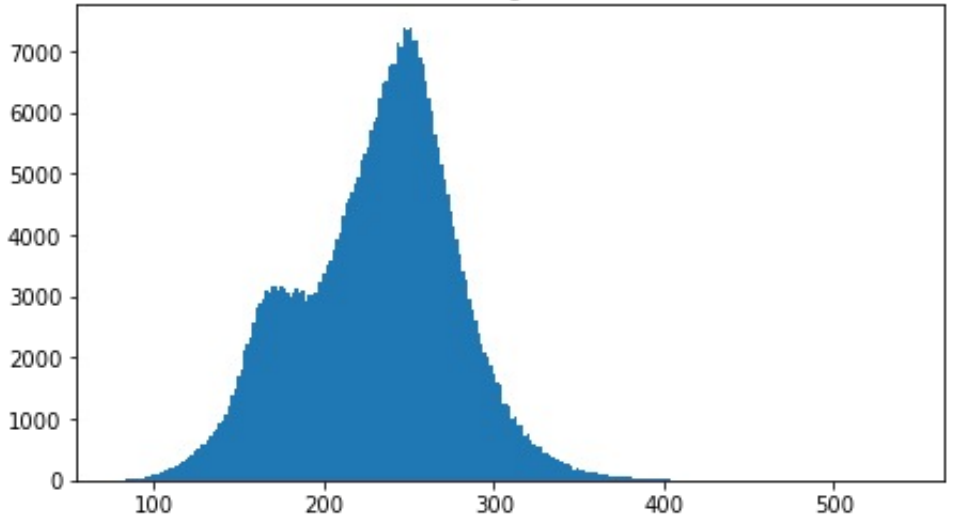
Hitogram



Ảnh nhiễu Erlang (Gammar)



Histogram

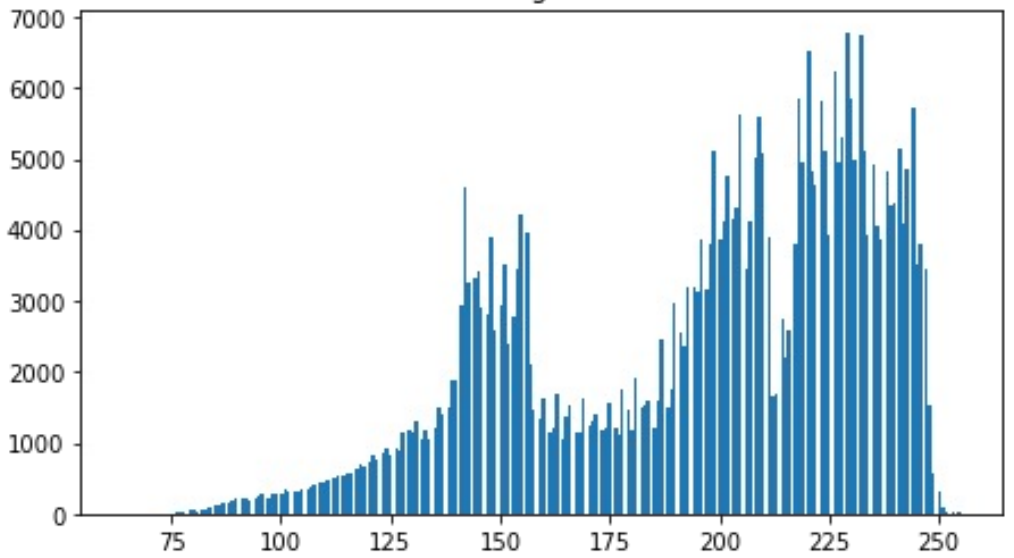


Một số loại nhiễu trong ảnh

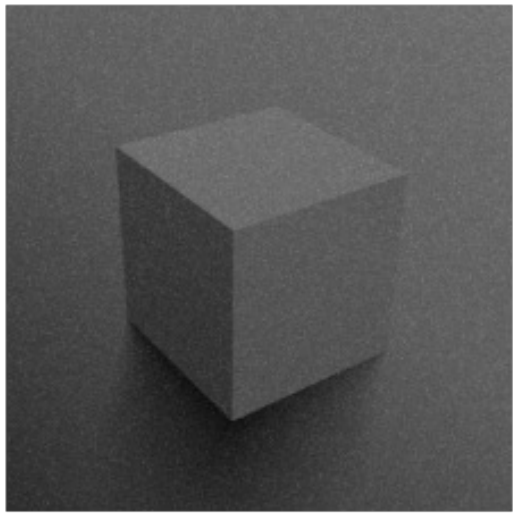
4. Nhiễu hàm mũ

Ảnh gốc

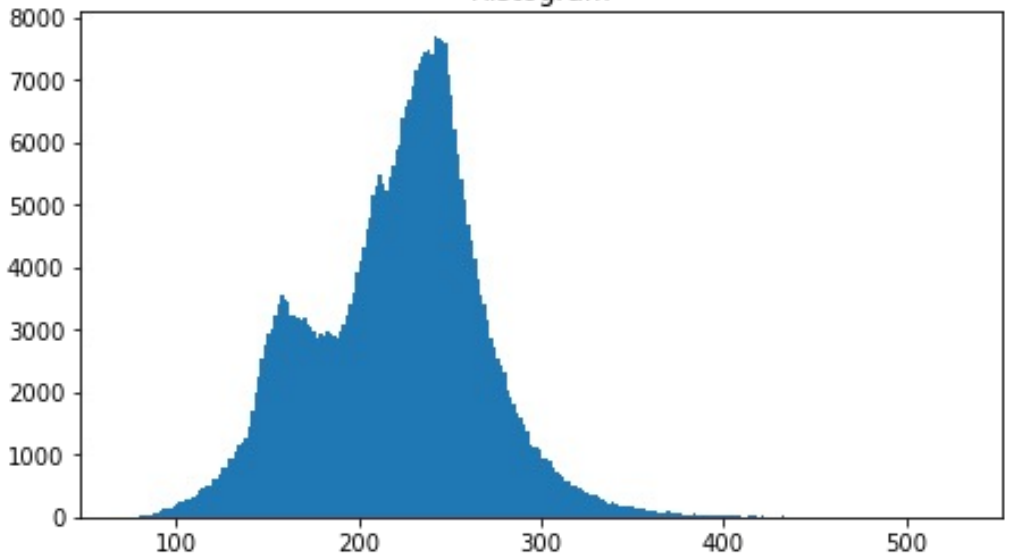
Hitogram



Ảnh nhiễu hàm mũ



Histogram

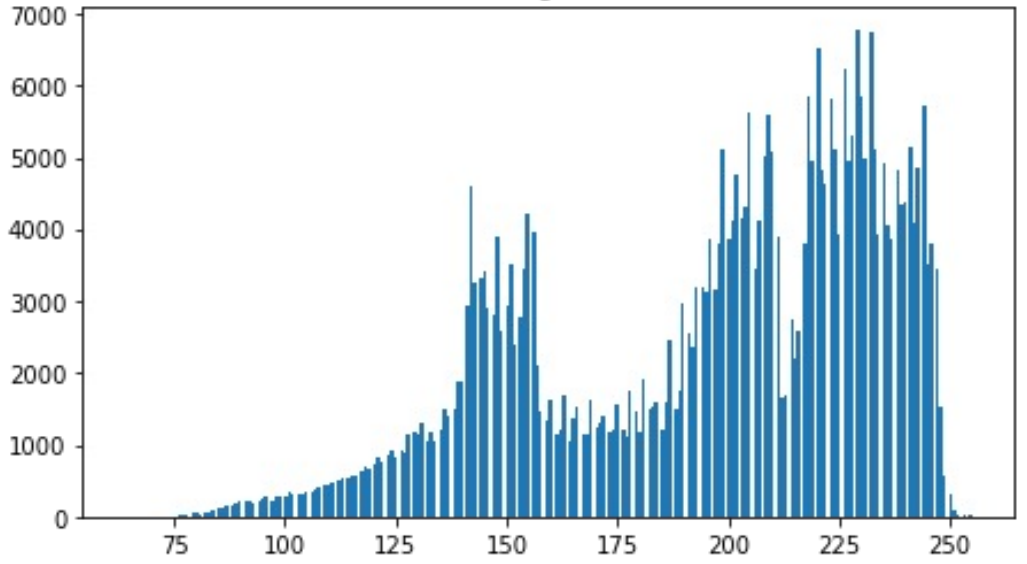


Một số loại nhiễu trong ảnh

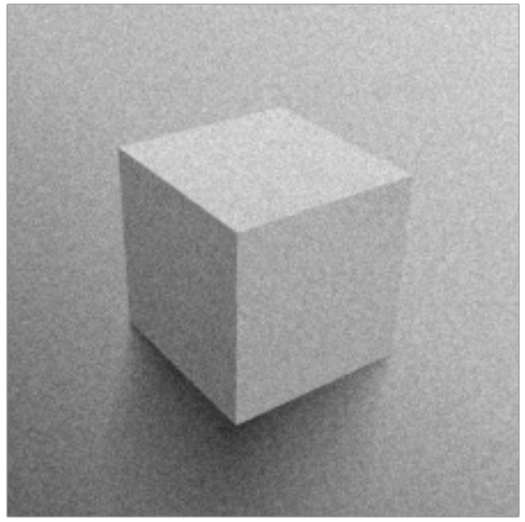
5. Nhiễu đồng nhất (Uniform)

Ảnh gốc

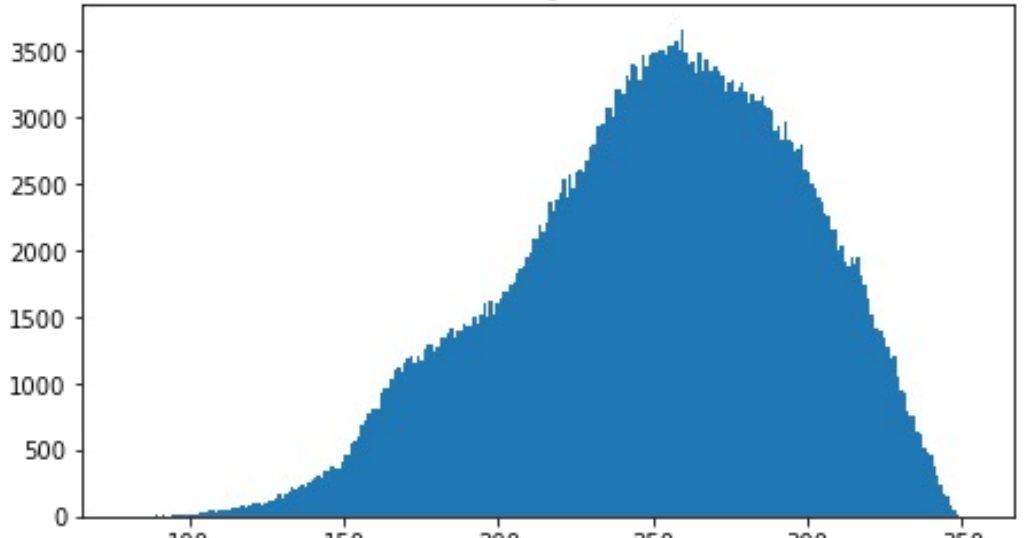
Hitogram



Ảnh nhiễu Uniform



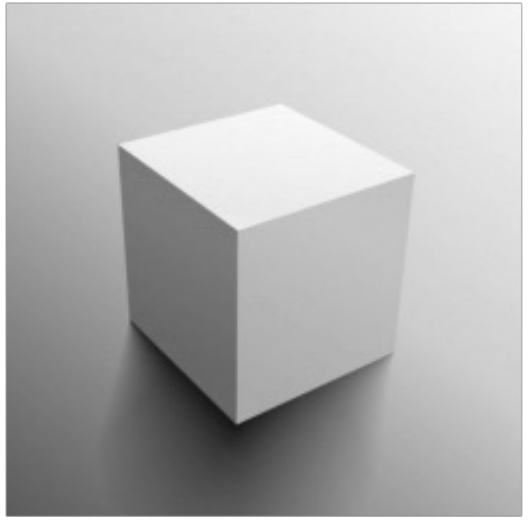
Histogram



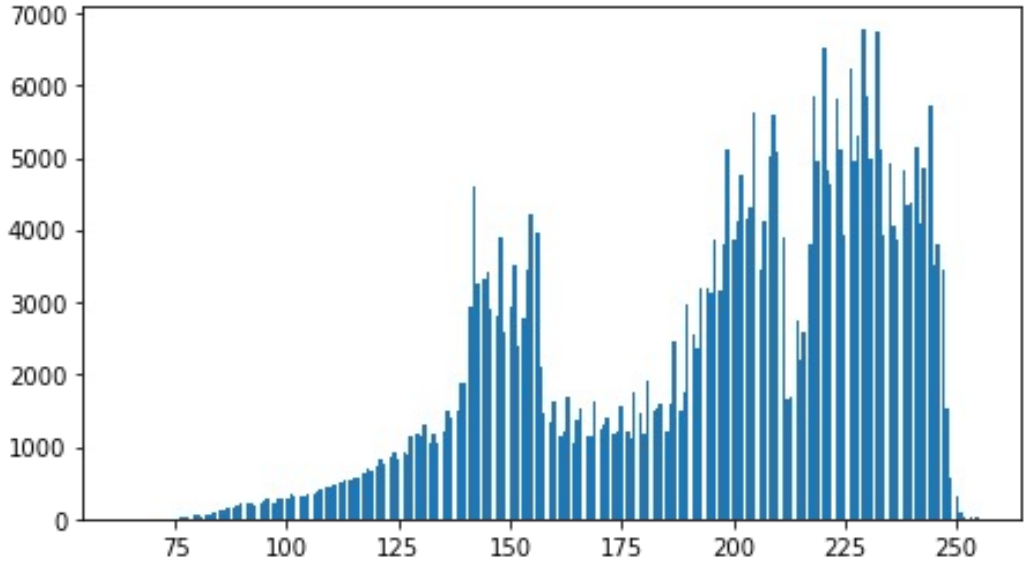
Một số loại nhiễu trong ảnh

6. Nhiễu muối tiêu

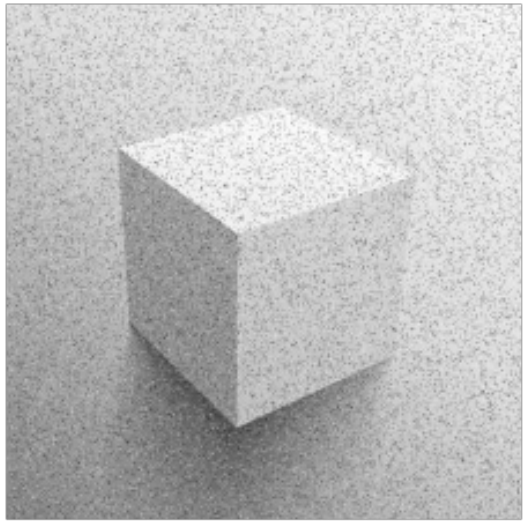
Ảnh gốc



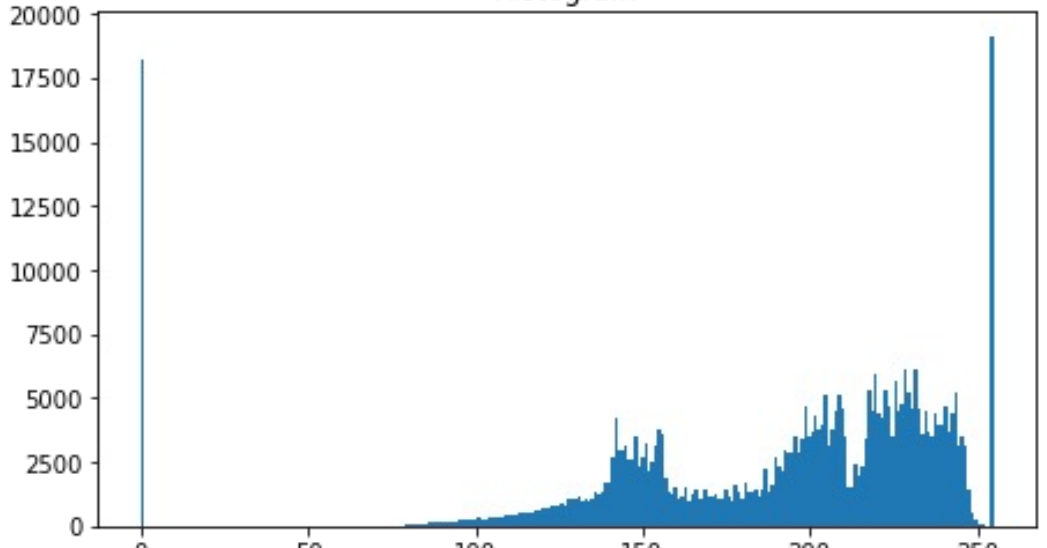
Hitogram



Ảnh nhiễu muối tiêu



Histogram



Khử nhiễu trong ảnh (khôi phục ảnh)

Mục đích là để thu nhận được ảnh càng gần với ảnh gốc càng tốt thông qua việc loại bỏ nhiễu.

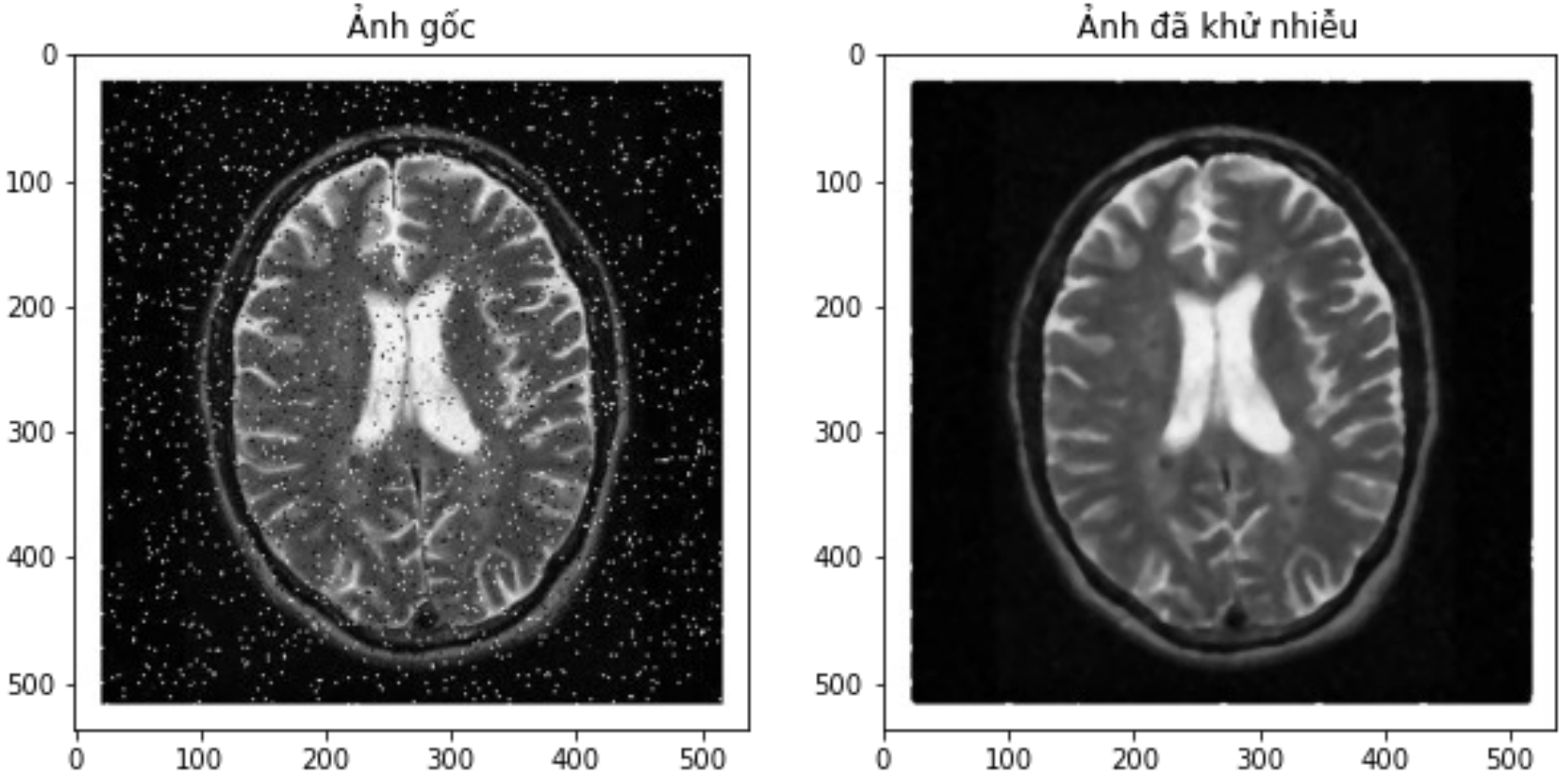
Một số phương pháp loại bỏ nhiễu:

- ❖ Lọc trung bình: Trung bình số học, hình học, Harmonic, Contra harmonic
- ❖ Lọc thứ tự thống kê: Trung vị, Max, Min, Điểm giữa max-min, trung bình cắt alpha.
- ❖ Lọc thích nghi

Thực hành số 2.9

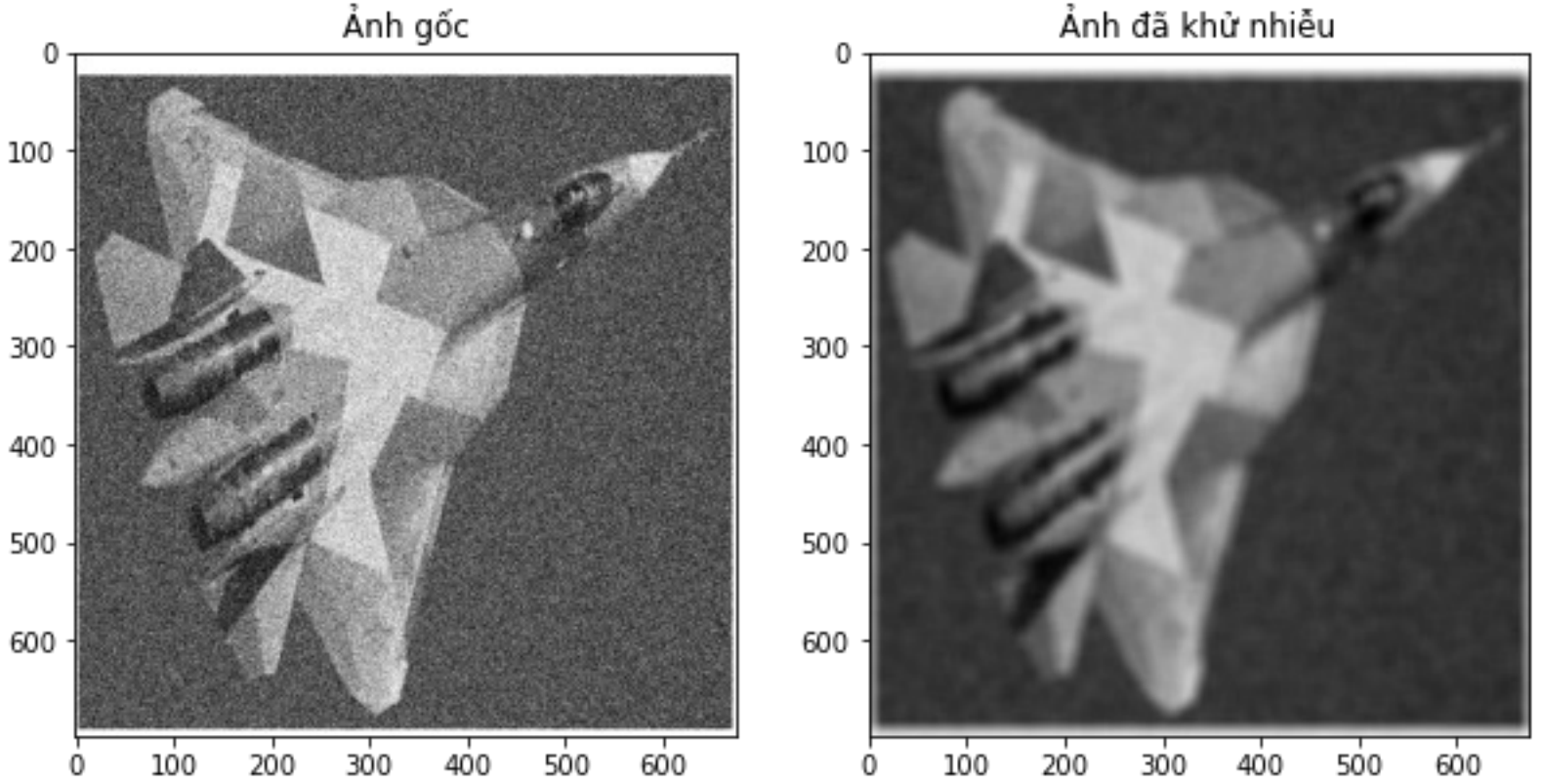
Thực hành 2.9

Yêu cầu 1: Sinh viên đọc ảnh images/Thuchanh2_9_1.jpeg. Lựa chọn phương pháp khử nhiễu và tham số phù hợp để thu được ảnh khử nhiễu tốt nhất. Lưu ảnh đã xử lý vào thư mục images/Saves/MSV_brain.png



Thực hành 2.9

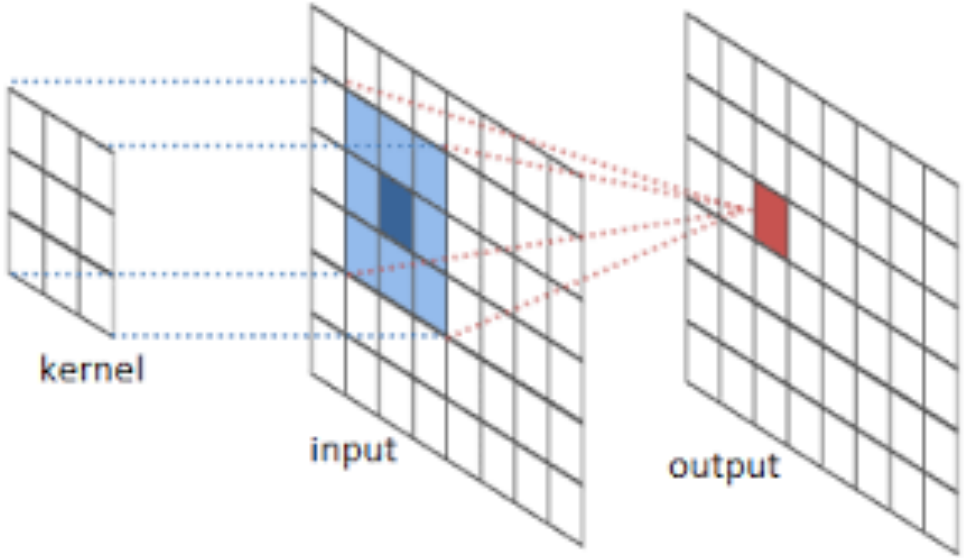
Yêu cầu 2: Sinh viên đọc ảnh images/Thuchanh2_9_2.png. Lựa chọn phương pháp khử nhiễu và các tham số phù hợp để thu được ảnh khử nhiễu tốt nhất. Lưu ảnh đã xử lý vào thư mục images/Saves/MSV_plane.png

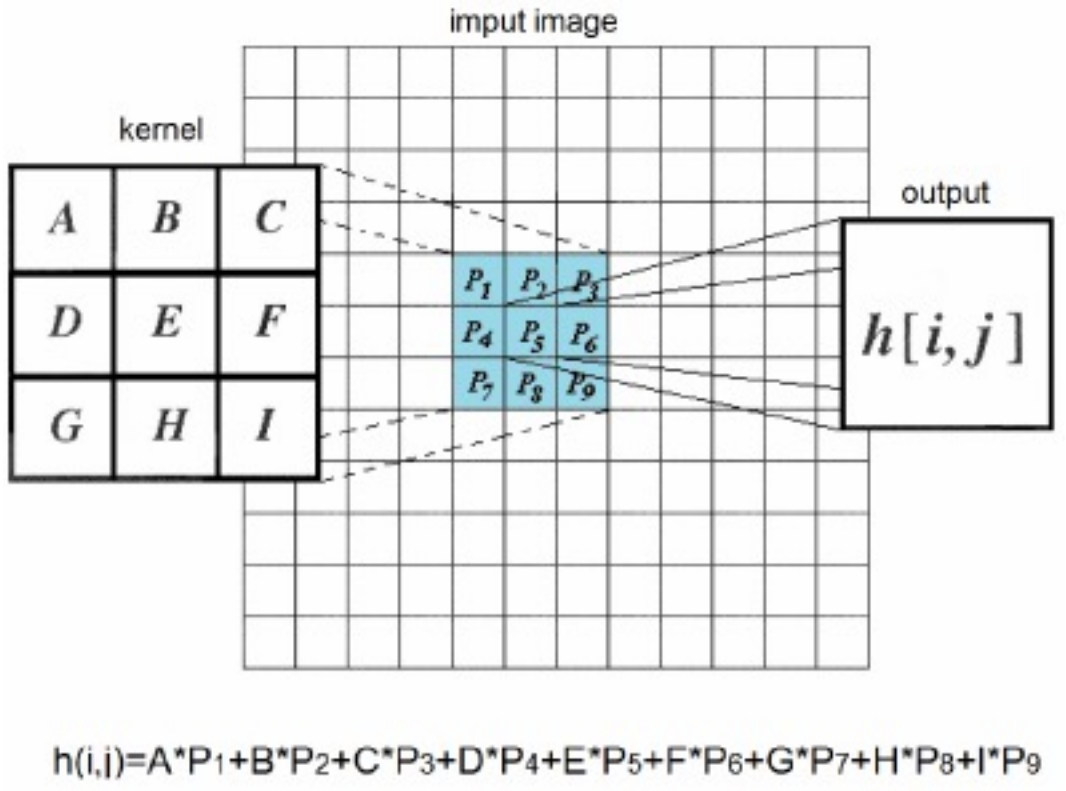


3. Giới thiệu Nhân tích chập

Mặt nạ lọc:

- Mặt nạ lọc còn được gọi là kernel, fifter, mask, là một ma trận vuông có kích thước 3x3, 4x4, 5x5, 7x7…
- Mặt nạ lọc di chuyển trên ảnh và thao tác lân cận với điểm ảnh.

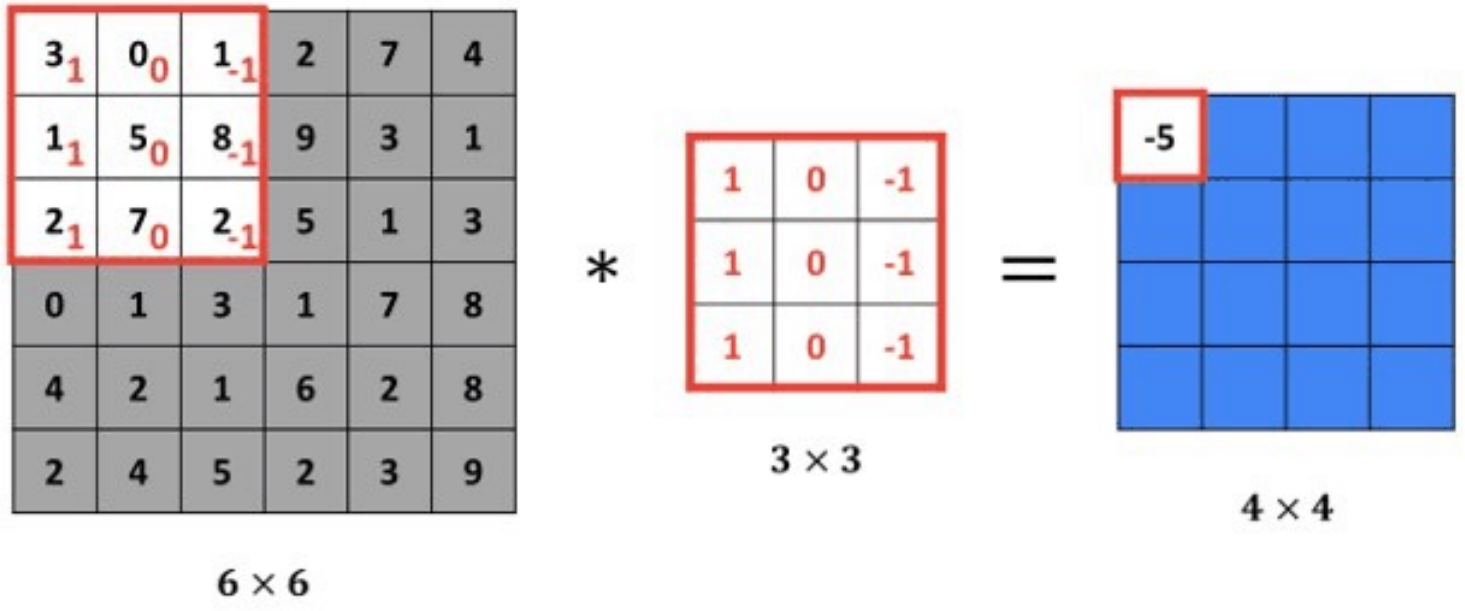




$h(i,j)=A*P_1+B*P_2+C*P_3+D*P_4+E*P_5+F*P_6+G*P_7+H*P_8+I*P_9$

Nhân tích chập

- Nhân tích chập là cho ma trận ảnh nhân với một ma trận lọc (Kernel). Ma trận lọc lọc (Kernel) còn có thể được gọi là cửa sổ chập (trong phép nhân chập), cửa sổ lọc, mặt nạ,…Việc nhân ảnh với ma trận lọc giống như việc trượt ma trận lọc theo hàng trên ảnh và nhân với từng vùng của ảnh, cộng các kết quả lại tạo thành kết quả của điểm ảnh trung tâm.



$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

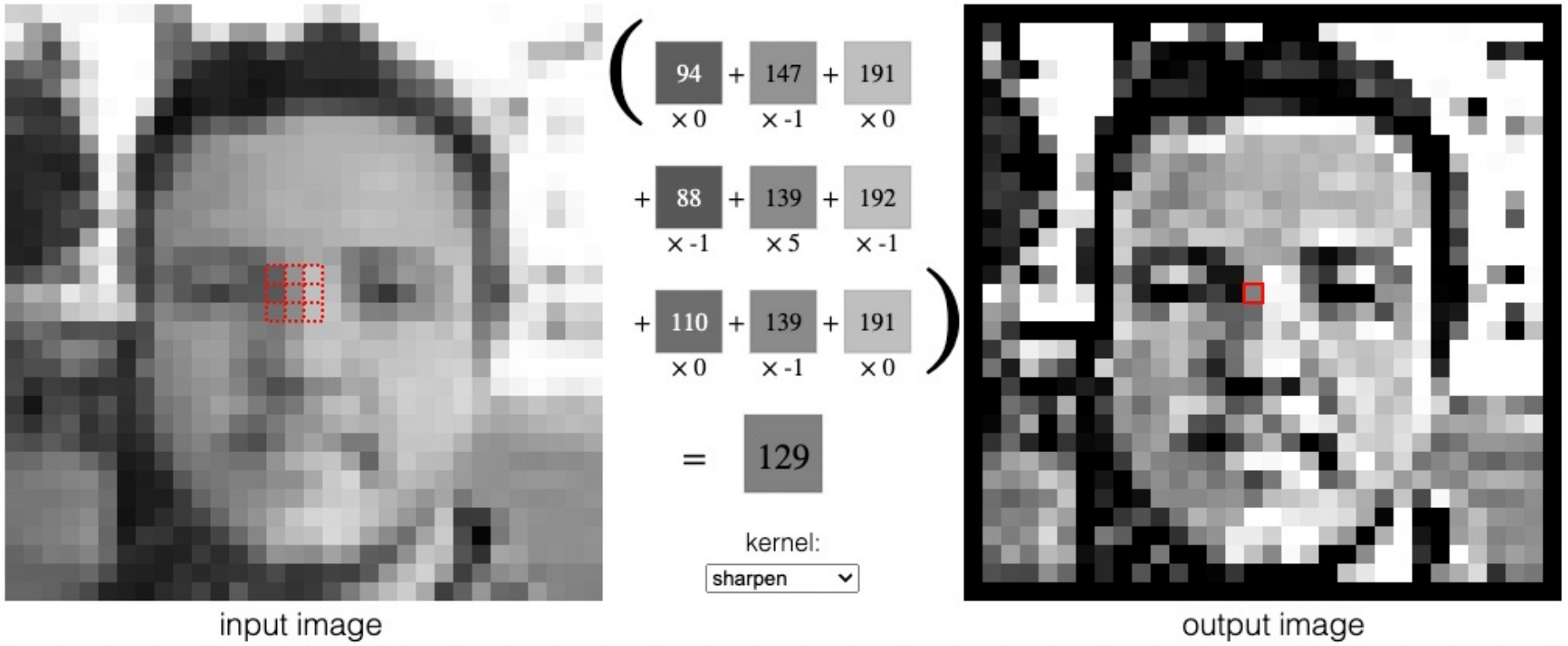
Nhân tích chập

- **Mặt nạ lọc**

sharpen

0	-1	0
-1	5	-1
0	-1	0

Nhân tích chập

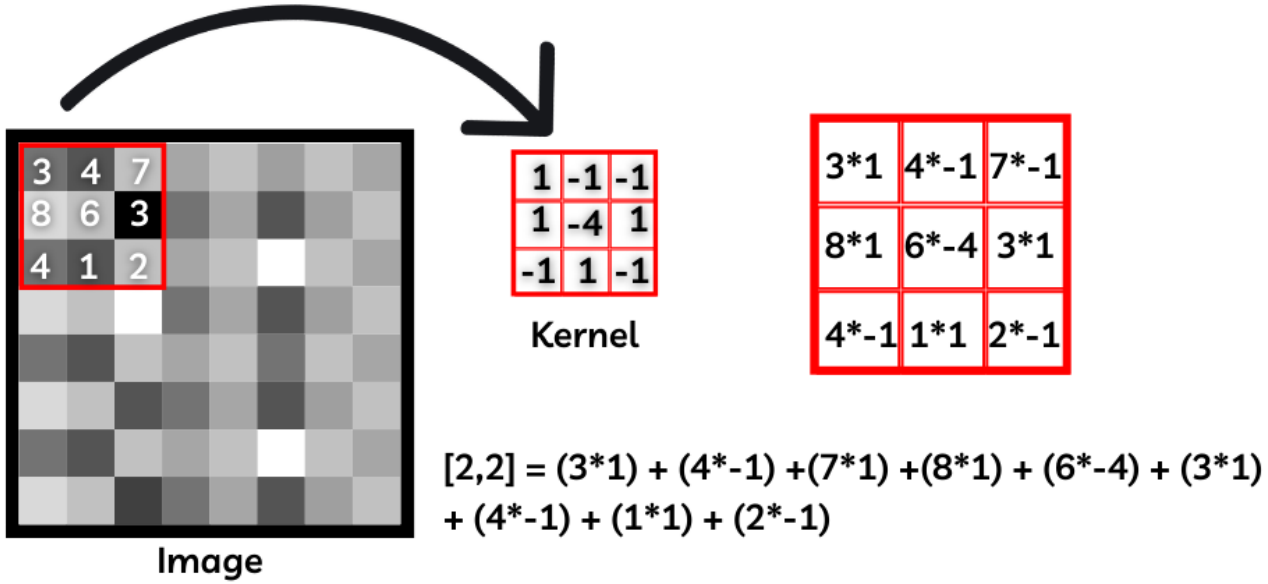


https://setosa.io/ev/image-kernels/

4. Nhân tích chap 2D với OpenCV

Nhân tích chập 2D với OpenCV

- Nguyên tắc chung của các phương pháp lọc là cho ma trận ảnh nhân với một ma trận lọc (Kernel). Ma trận lọc lọc (Kernel) còn có thể được gọi là cửa sổ chập (trong phép nhân chập), cửa sổ lọc, mặt nạ,…Việc nhân ảnh với ma trận lọc giống như việc trượt ma trận lọc theo hàng trên ảnh và nhân với từng vùng của ảnh, cộng các kết quả lại tạo thành kết quả của điểm ảnh trung tâm.
- Với mỗi phép lọc ta có những ma trận lọc (Kernel) khác nhau, không có một quy định cụ thể nào cho việc xác định M. Kích thước ma trận M là một số lẻ. Ví dụ: 3x3, 5x5.



Nhân tích chập 2D với OpenCV

OpenCV hỗ trợ phương thức lọc ảnh 2D:

cv2.filter2D(img, ddepth, kernel)

Trong đó:

- img: ảnh gốc muốn áp dụng bộ lọc
- ddepth: Độ sâu của ảnh đầu ra; -1: độ sâu của ảnh đầu ra bằng độ sâu của ảnh đầu vào.
- kernel: ma trận lọc

Một số bộ lọc thông dụng:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Identity kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen kernel

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Box blur

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian blurr kernel

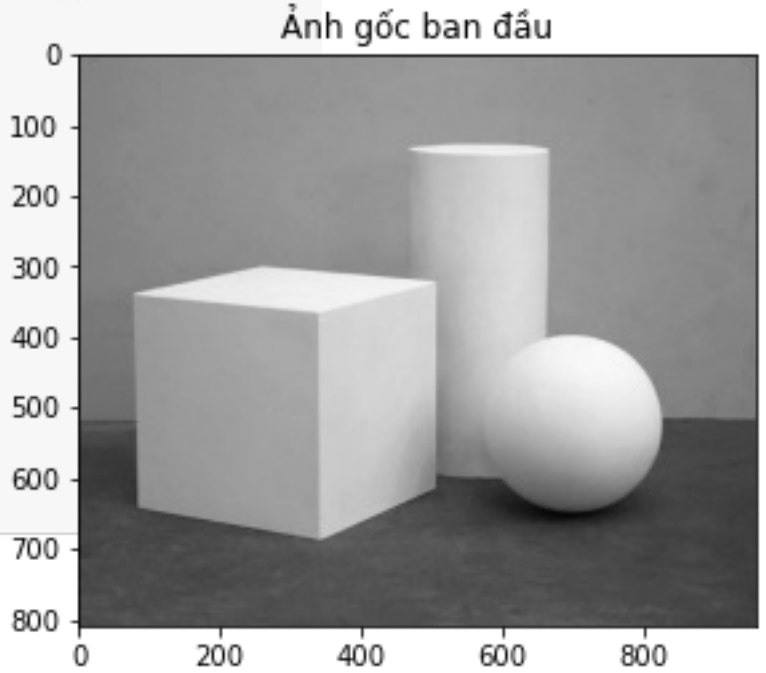
Một số bộ lọc thông dụng:

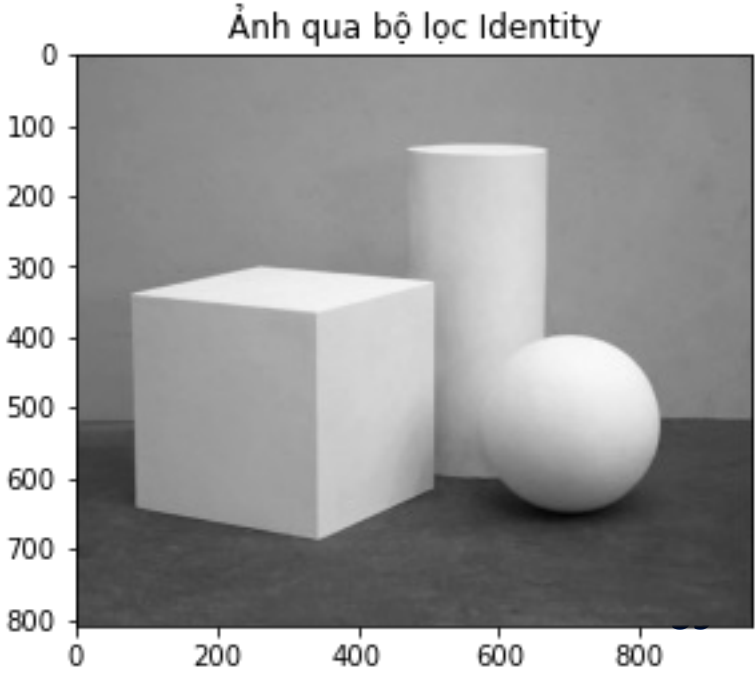
Bộ lọc identity:

```
#Khởi tạo kernel:
kernel_identity = np.array([[0,0,0],
                            [0,1,0],
                            [0,0,0]])
kernel_identity
```

```
array([[0, 0, 0],
       [0, 1, 0],
       [0, 0, 0]])
```

```
#Thực hiện lọc ảnh với kernel ở trên:
img_identity = cv2.filter2D(img, -1, kernel_identity)

#Hiển thị ảnh đã xử lý qua bộ lọc:
plt.figure(figsize=(10,8))
plt.subplot(1,2,1)
plt.imshow(img,cmap='gray')
plt.title('Ảnh gốc ban đầu')


plt.subplot(1,2,2)
plt.imshow(img_identity,cmap='gray')
plt.title('Ảnh qua bộ lọc Identity')
plt.show()
```



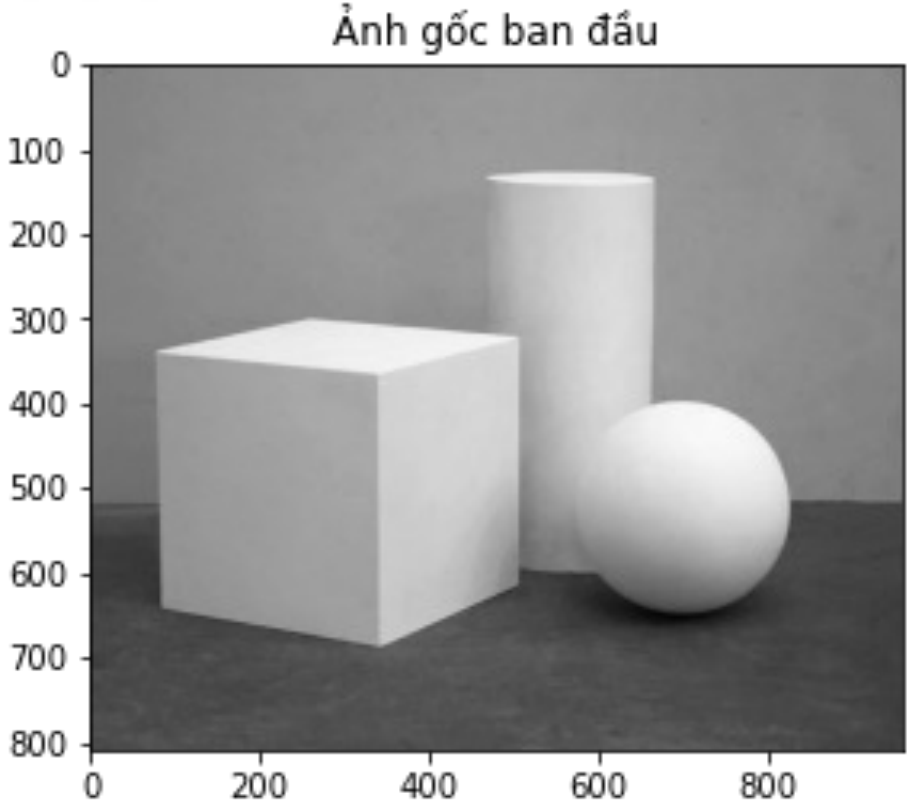


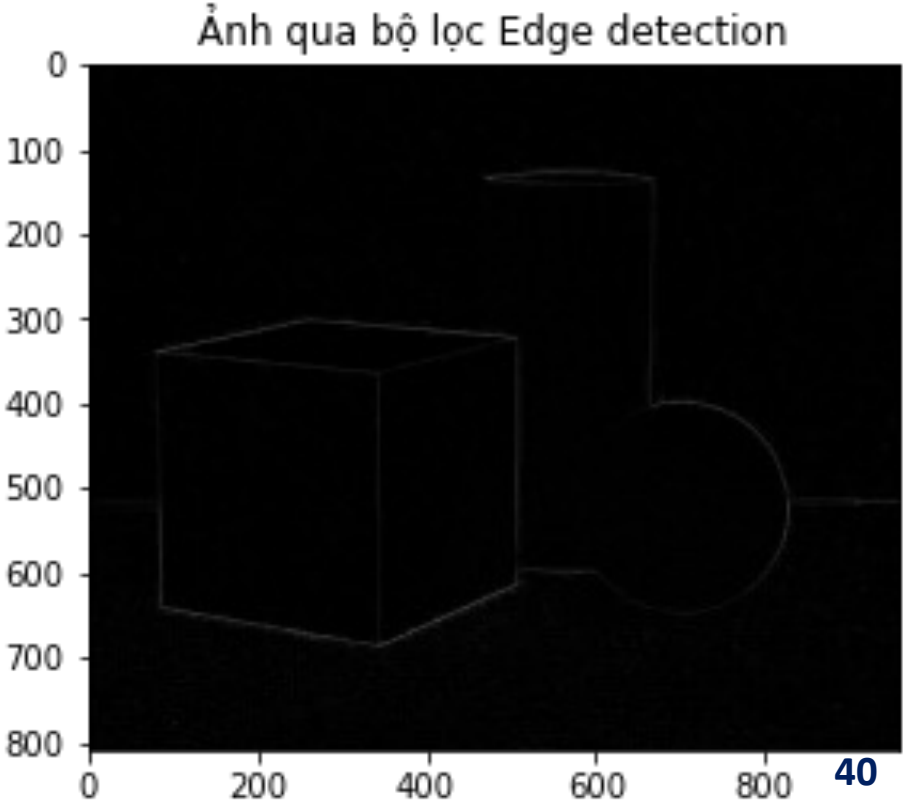
Một số bộ lọc thông dụng:

Bộ lọc Edge:

```
#Khởi tạo kernel:
kernel_edge = np.array([[-1,-1,-1],
                        [-1, 8,-1],
                        [-1,-1,-1]])
kernel_edge
```

```
array([[-1, -1, -1],
       [-1,  8, -1],
       [-1, -1, -1]])
```



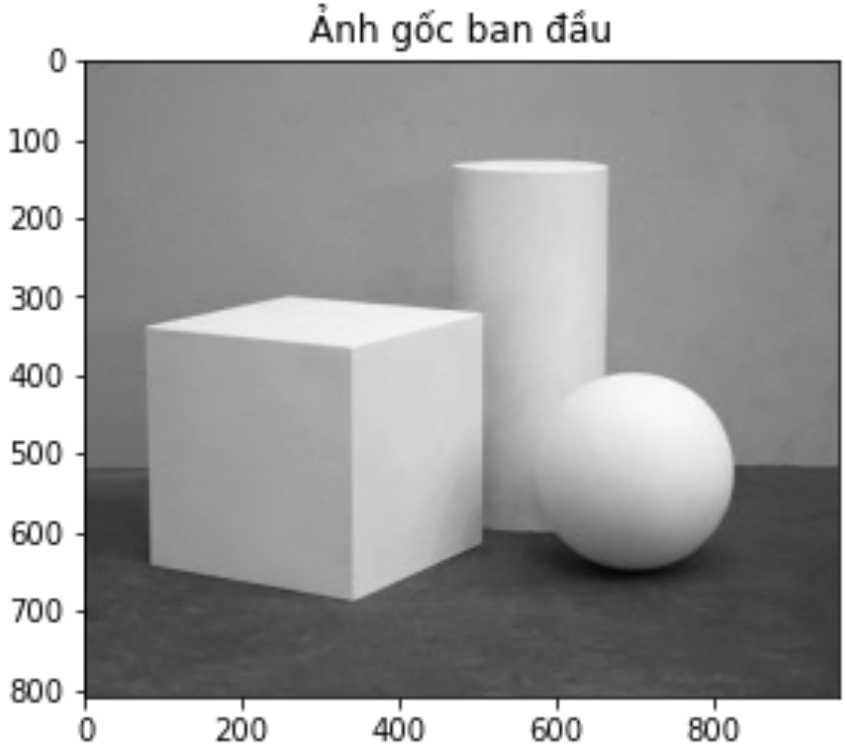


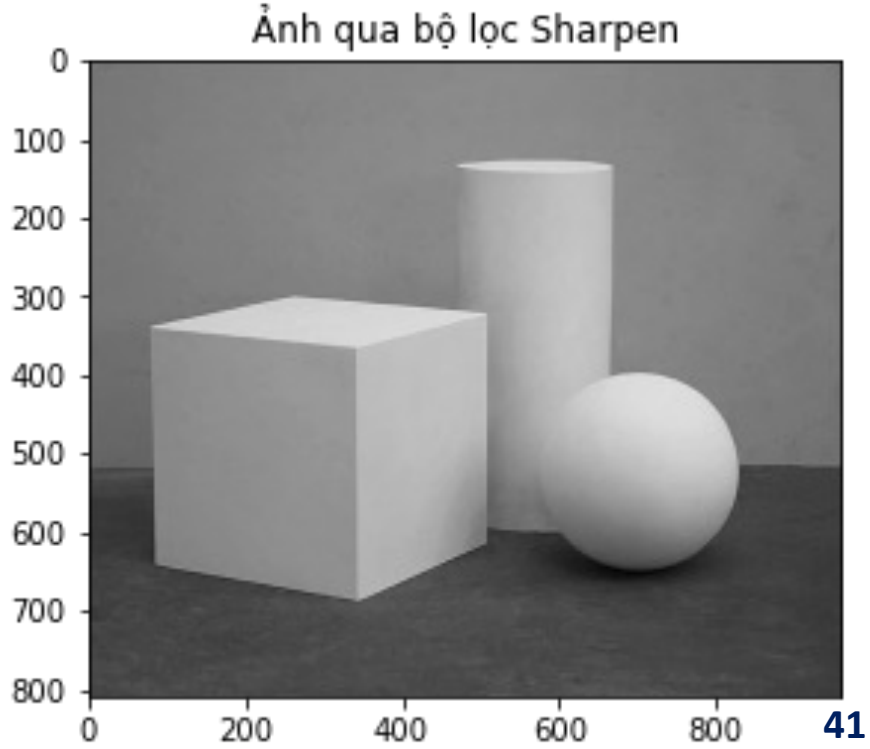
Một số bộ lọc thông dụng:

Bộ lọc sharpen:

```
#Khởi tạo kernel:
kernel_sharpen = np.array([[ 0,-1, 0],
                           [-1, 5,-1],
                           [ 0,-1, 0]])
kernel_sharpen
```

```
array([[ 0, -1,  0],
       [-1,  5, -1],
       [ 0, -1,  0]])
```





Một số bộ lọc thông dụng:

Bộ lọc mịn:

```
#Khởi tạo kernel:
kernel_blur = np.array([[1/9,1/9, 1/9],
                        [1/9, 1/9,1/9],
                        [1/9, 1/9,1/9]])

print(kernel_blur)
```

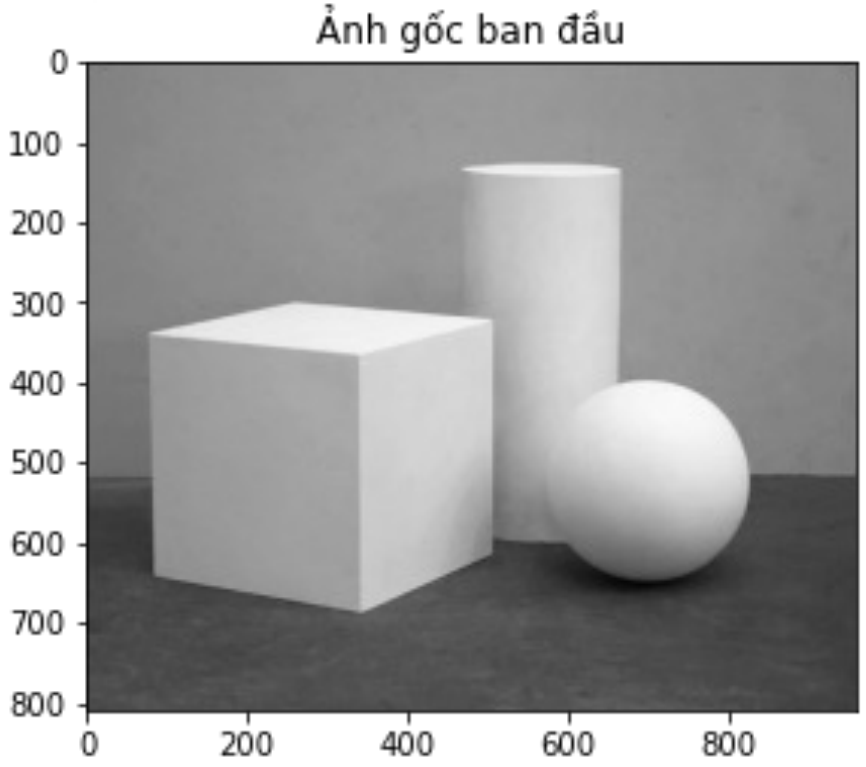
```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

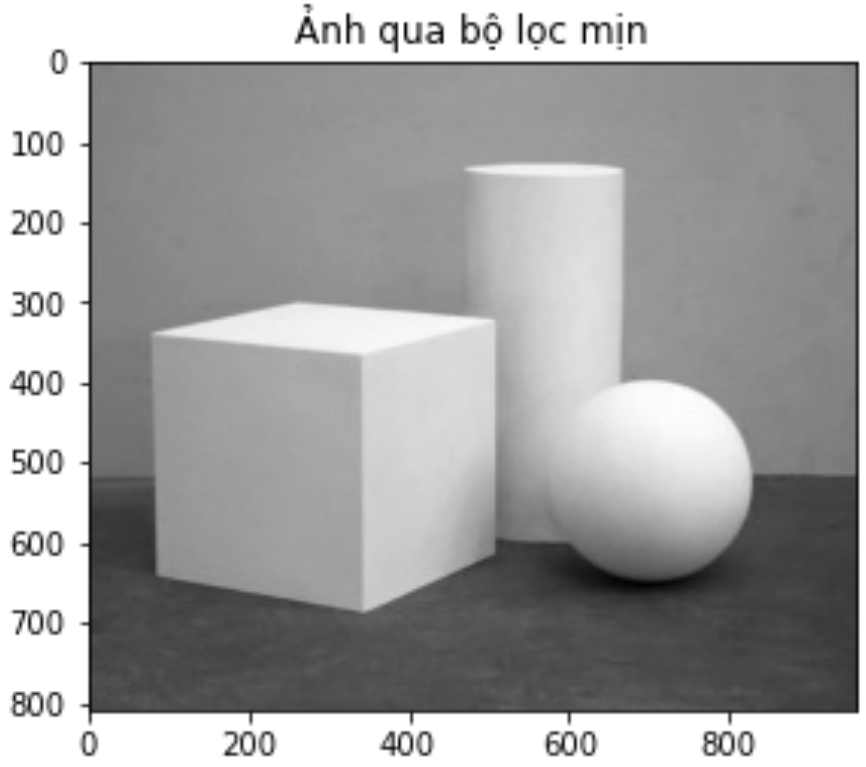
Averaging Filter 3x3

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Averaging Filter 5x5

$$\frac{1}{25} * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

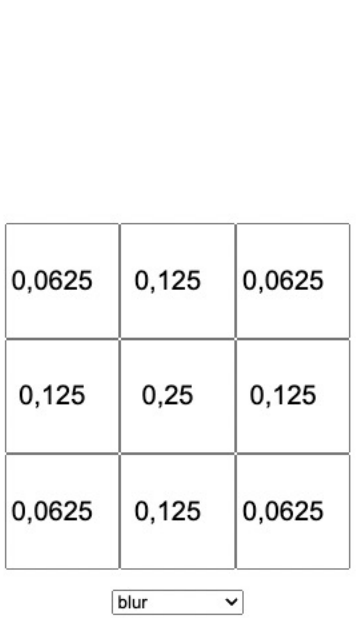


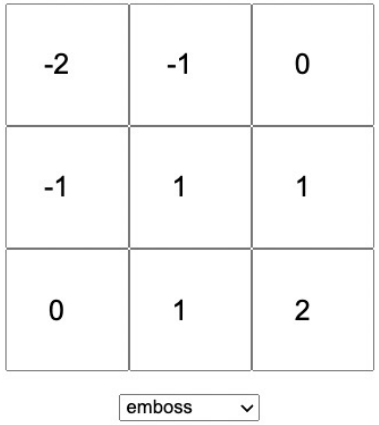


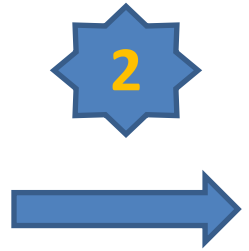
Thực hành số 2.10

Thực hành 2.10

- Sinh viên đọc và hiển thị ảnh images/Thuchanh2_10.png ở chế độ ảnh xám.
- Áp dụng 4 bộ lọc sau trên ảnh. Lưu và hiển thị kết quả sau khi lọc.







Thực hành 2.10

- Áp dụng các bộ lọc sau trên ảnh. Lưu và hiển thị kết quả sau khi lọc.

-1	-2	-1
0	0	0
1	2	1

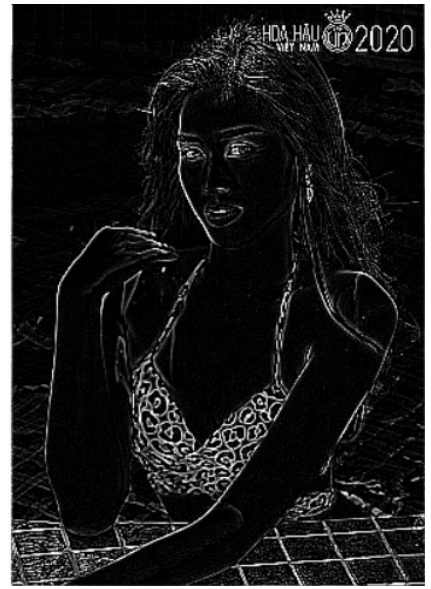
bottom sobel

3

-1	-1	-1
-1	8	-1
-1	-1	-1

outline

4




QUESTIONS
Q&A
ANSWERS