VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF SCIENCE
COMPUTER VISION

DIGITAL IMAGE & VIDEO PROCESSING

LAB 01 REPORT

Teacher: Ly Quoc Ngoc
Nguyen Manh Hung

Student: Duong Minh Anh Khoi – 20127212

Ho Chi Minh city - 2022

Contents

Color transformation

1 Linear mapping

- Brightness modification.

$$g(x, y) = f(x, y) + b$$

  - Function:

```
#Điều chỉnh độ sáng
brightness = img_gray + 25 #Công thức điều chỉnh độ sáng g(x,y) = f(x,y)
plt.imshow(brightness, cmap="gray");
plt.title("Brightness")
```

  - Result:

- Contrast modification.

$$g(x, y) = a.f(x, y)$$

  - Function:

```
#Điều chỉnh độ tương phản
contrast =  img_gray * ((255-19)/255) #Công thức điều chỉnh độ tương phản g(x,y) = a.f(x,y)
plt.imshow(contrast, cmap="gray");
plt.title("Contrast")
```

  - Result:

2 Linear mapping

- Brightness modification.

$$g(x,y) = clogf(x,y)$$

  - Function:

```
#Logarithmic mapping
plt.subplot(1,3,2)
img_log = np.log(img_gray)*45 #g(x,y)=clog(f(x,y))
plt.imshow(img_log, cmap="gray")
plt.title("Logarithmic")
plt.xticks([]), plt.yticks([])
```

  - Result:

Image blurring & image smoothing

3 Averaging filter

- Algorithm:
  - Perform convolutional math: $g(x,y) = \sum_i \sum_j f(x-i, y-j).h(i,j), (i,j) \in O$ with 2D kernel:

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

- Function:

```
plt.subplot(1, 2, 2)
#Averaging
kernel = np.ones((5,5),np.float32)/25 #Create kernel
avg = cv2.filter2D(img,-1,kernel)

plt.imshow(avg)
plt.title('Averaging')
plt.xticks([]), plt.yticks([])
```

4 Gaussian Filter

- Algorithm:
  - Perform convolutional math: $g(x,y) = \sum_i \sum_j f(x-i, y-j).h(i,j), (i,j) \in O$ with 2D Gaussian kernel:

$$h(i,j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- Function:

```
#Gaussian
gaussian = cv2.GaussianBlur(img , (5, 5), 0)
plt.imshow(gaussian);
plt.title("Gaussian")
plt.xticks([]), plt.yticks([])
```

5 Median Filter

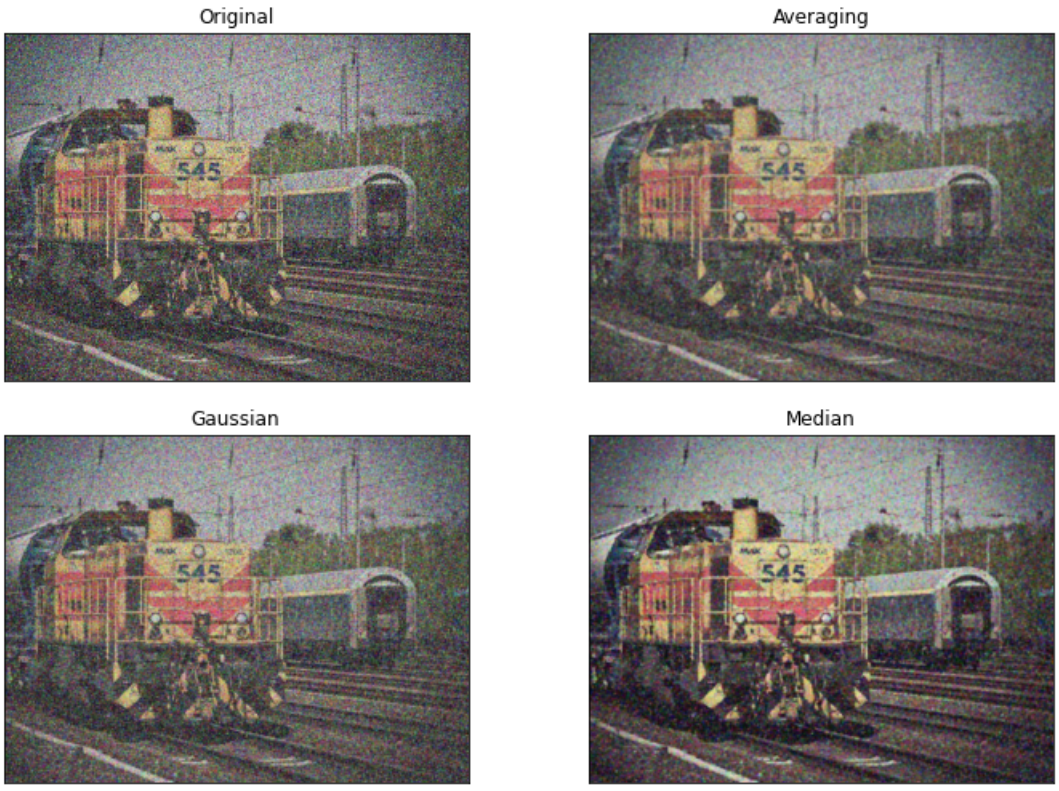
- Algorithm:
  - Get the pixel median values of the image corresponding to kernel of filter:

$$g(x, y) = med\{f(x + i, y + j), (i, j) \in)\}$$

- Function:

```
#Median
median = cv2.medianBlur(img,5)
plt.imshow(median)
plt.title("Median")
plt.xticks([]), plt.yticks([])
plt.show()
```

6 Result



Edge detection

7 Gradient operator

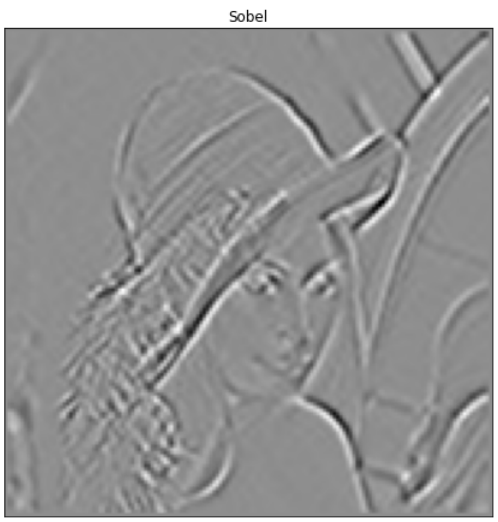
- Algorithm:
  - To determine where there is a large variation in brightness, derivative with respect to the gradient vector and take the vector perpendicular to the gradient vector to represent the edge through the extreme points.
- Sobel edge detection

```
plt.figure(figsize=(12,4))
sobel = cv2.Sobel(src=img_gray, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) #Sobel Edge Detection
plt.subplot(1, 2, 1)
plt.imshow(sobel, cmap="gray")
plt.title("Sobel")
plt.xticks([]), plt.yticks([])
```

8 Canny edge detection

```
edges = cv2.Canny(image=img_gray, threshold1=100, threshold2=200) # Canny Edge Detection
plt.imshow(edges, cmap="gray")
plt.title("Canny")
plt.xticks([]), plt.yticks([])
plt.show()
```

9 Result



Rating table

Requirement	**Rate**
Transform color	100%
Transform geometry	50 %
Image blurring, image smoothing)	100 %
Edge detection	60 %