DIGITAL IMAGE PROCESSING

Chapter 2. Image Enhancement in Spatial Domain

PGS.TS. Hoàng Văn Dũng
Faculty of Information Technology – HCMUTE

1. Convolutional function and application:

Convolution is a basic mathematical operation that several image processing operators use. Convolution is a method of multiplying two arrays of integers, often of different sizes but of the same dimensionality, to produce a third array of the same dimensionality.

Convolution is applied to image filtering : Image filtering is changing the pixel value of a specific image to blur, sharpen, emboss, or make edges more clear. It changes the appearance of the original image.

```
import cv2
import numpy as np
from matplotlib import pyplot
import matplotlib.pyplot as plt
#%% Xây dựng hàm tích chập
def conv(A,k,b=0):
    kh,kw=k.shape
    if b>0:
        h,w=A.shape
        B=np.ones((h+kh-1,w+kw-1))
        th=int(kh/2)
        tw=int(kw/2)
        B[th:h+th,tw:w+tw]=A
        A=B
    h,w=A.shape
    C=np.ones((h,w))
    for i in range(0,h-kh+1):
        for j in range(0,w-kw+1):
            sA=A[i:i+kh,j:j+kw]
            C[i,j]=np.sum(k*sA)
    C=C[0:h-kh+1,0:w-kw+1]
    return C
#%% Đọc file ảnh và hiển thị
img = cv2.imread('images/cat.png')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
img=cv2.resize(img,(200, 200))
plt.imshow(img)
plt.show()
#%% Tích chập ảnh với kernel 5x5=> lọc trung bình
k=np.ones((5,5))/25
r,g,b = cv2.split(img)
B=conv(b,k,1)
G=conv(g,k,1)
R=conv(r,k,1)
imgN=cv2.merge((R,G,B))
imgN=np.array(imgN,dtype='uint8')
```

```
plt.imshow(imgN)
plt.show()
```

Thay vì tự code hàm covolution, ta có thể sử dụng hàm `convolve`

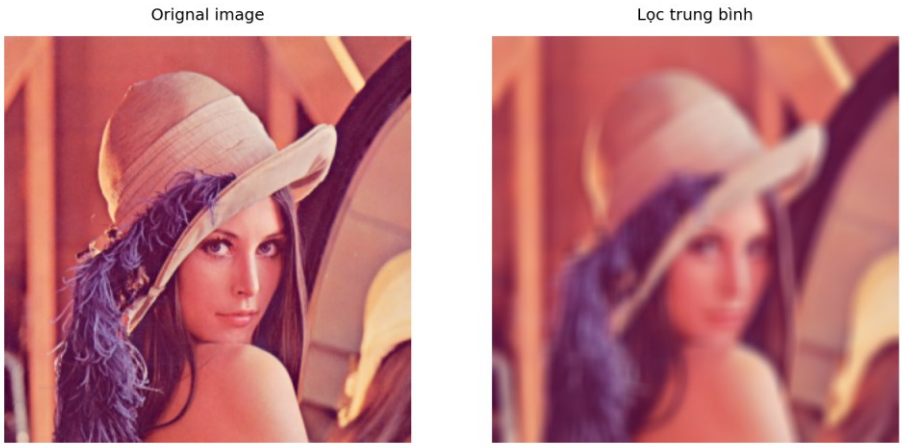
```
#%% SỬ DỤNG HÀM convolve trong thư viện scipy
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve
#% Đọc file ảnh và hiển thị
img = cv2.imread('images/cat.png')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
img=cv2.resize(img,(200, 200))
plt.imshow(img)
plt.show()
#% Tích chập ảnh với kernel 5x5=> lọc trung bình
k=np.ones((5,5))/25
r,g,b = cv2.split(img)
R=convolve(r,k)
G=convolve(g,k)
B=convolve(b,k)
imgN=cv2.merge((R,G,B))
imgN=np.array(imgN,dtype='uint8')
plt.imshow(imgN)
plt.show()
  #%%
```

2. Lowpass Filtering (Blurring).

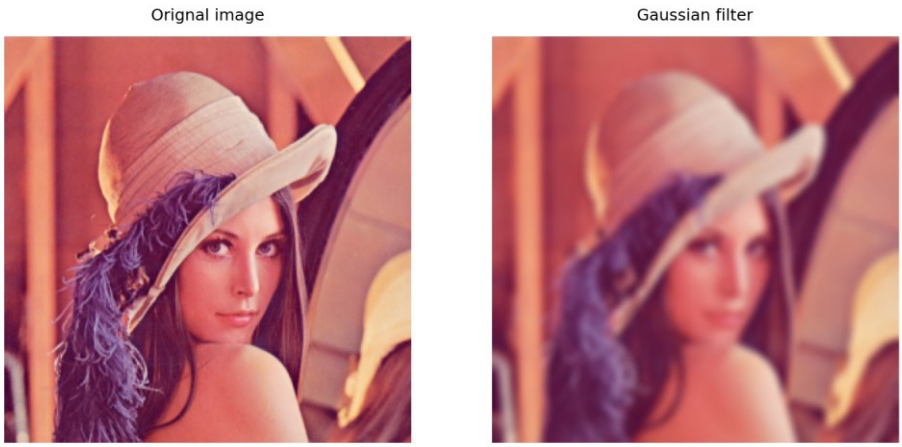
Lowpass filtering (smoothing), is employed to remove high spatial frequency noise from a digital image. The low-pass filters usually employ moving window operator which affects one pixel of the image at a time, changing its value by some function of a local region (window) of pixels. The operator moves over the image to affect all the pixels in the image.

Để làm cơ sở cho các phép lọc thông thấp và lọc thông cao, đầu tiên cần xây dựng convolution 2D cho ảnh gray hoặc ảnh màu 3 lớp.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve
plt.rcParams.update({'font.size': 5})
#Hàm convolution cho ảnh không gian với ảnh gray hoặc ảnh màu
3 lớp (RGB)
def Conv(img,k):
    Out=np.zeros_like(img)
    if img.ndim >2:
        for i in range(3):
           Out[:,:,i]=convolve(img[:,:,i],k)
    else:
        Out=convolve(img,k)
    return Out
#%% GAUSSIAN Kernel
def Gausskernel(l=5, sig=1.5):
    s=round((l - 1)/2)
    ax = np.linspace(-s, s, l)
    gauss = np.exp(-np.square(ax) / (2*(sig**2)))
    kernel = np.outer(gauss, gauss)
     #tính tích the outer product of two vectors.
    return kernel / np.sum(kernel)
#%% LỌC TRUNG BÌNH với kernel 11x11
img = cv2.imread('Images/lenna.jpg', cv2.IMREAD_COLOR)
img= cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.figure(dpi=300)
subf=plt.subplot(1,2,1); plt.imshow(img)
plt.axis('off'); subf.set_title("Orignal image")

k=np.ones((11,11))/(11*11)
imgOut=Conv(img,k)
subf=plt.subplot(1,2,2); plt.imshow(imgOut)
plt.axis('off'); subf.title.set_text('Mean filter')
plt.show()
```




```
#%% LỌC Gaussian với kernel 11x11 và sig=3
img = cv2.imread('Images/lenna.jpg', cv2.IMREAD_COLOR)
img= cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.figure(dpi=300)
subf=plt.subplot(1,2,1); plt.imshow(img)
plt.axis('off'); subf.set_title("Orignal image")
s=11;sig=3;
k=Gausskernel(s, sig)
imgOut=Conv(img,k)
subf=plt.subplot(1,2,2); plt.imshow(imgOut)
plt.axis('off'); subf.title.set_text('Gaussian filter')
plt.show()
```



```
#%% LỌC Gaussian với kernel kích thước khác nhau và sigmal khác nhau
img = cv2.imread('Images/lenna.jpg', cv2.IMREAD_COLOR)
img= cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
GauSig= np.linspace(1,4,5,endpoint=True)
ks= np.linspace(1,10,5,endpoint=True,dtype=int)*2+1
plt.figure(dpi=300)
subf=plt.subplot(2,3,1)
subf.imshow(img)
plt.axis('off'); subf.set_title("Orignal image")
for i,sig in enumerate(GauSig):
    k=Gausskernel(ks[i], sig)
    imgG=Conv(img,k)
    subf=plt.subplot(2,3,i+2)
    subf.imshow(imgG)
    subf.set_title('Gaussian(s='+str(ks[i])+'; sig='+
str(sig)[0:3]+')');
    subf.axis('off')
```

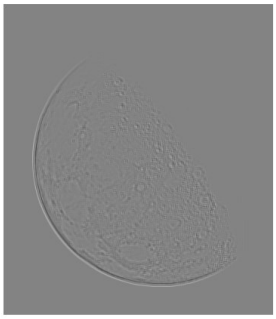
3. High pass filters (sharpening, edge detection)

A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image - the opposite of the low-pass filter. High-pass filtering works in the same way as low-pass filtering; it just uses a different convolution kernel.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve
plt.rcParams.update({'font.size': 5})
def Conv(img,k):
    Input=np.array(img,dtype='single')
    if Input.ndim >2:
        Out=np.zeros_like(Input)
        for i in range(3):
           Out[:,:,i]=np.convolve(Input[:,:,i],k)
    else:
        Out=convolve(Input,k)
    return Out
#%% Second Derivatives- Laplacian
img = cv2.imread('images/moon.jpg', cv2.IMREAD_GRAYSCALE)
img=resizeIm(img,0.3)
ksize=(11,11)
img = cv2.blur(img, ksize, cv2.BORDER_DEFAULT)
plt.figure(dpi=300)
subf=plt.subplot(2,2,1); subf.set_title("Orignal image")
plt.imshow(img,cmap='gray'); plt.axis('off') ;
# Tạo các kernal k1 4 hướng và k2 8 hướng
k1=np.array([[0,1,0],[1,-4,1],[0,1,0]])
k2=np.array([[1,1,1],[1,-8,1],[1,1,1]])

L1=Conv(img,k1)
subf=plt.subplot(2,2,2); subf.set_title("Laplacian with k1")
plt.imshow(L1,cmap='gray'); plt.axis('off')

L2=Conv(img,-k1)+img
subf=plt.subplot(2,2,3); subf.set_title("sharpening (Laplacian neg
kernel)")
plt.imshow(L2,cmap='gray'); plt.axis('off') ;

L3=Conv(img,k2)+img
subf=plt.subplot(2,2,4); subf.set_title("sharpening (Laplacian k2)")
plt.imshow(L3,cmap='gray'); plt.axis('off') ;
```

Orignal image

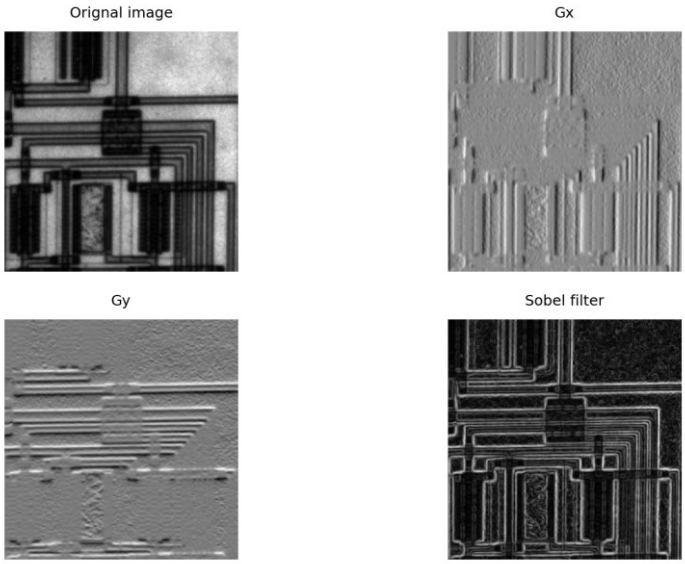
Laplacian with k1



sharpening (Laplacian neg kernel)

sharpening (Laplacian k2)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import convolve
plt.rcParams.update({'font.size': 5})
def Conv(img,k):
    Input=np.array(img,dtype='single')
    if Input.ndim >2:
        Out=np.zeros_like(Input)
        for i in range(3):
           Out[:,:,i]=np.convolve(Input[:,:,i],k)
    else:
        Out=convolve(Input,k)
    return Out

#%%Lọc Sobel theo 2 hướng x và y sau đó tính tổng độ lớn theo x và y
img = cv2.imread('images/circuit.tif', cv2.IMREAD_GRAYSCALE)
ky=np.array([[-1.0,-2,-1],[0,0,0],[1,2,1]])
kx=np.transpose(ky)
Gx=Conv(img,kx)
Gy=Conv(img,ky)
Gm=np.sqrt(Gx**2+Gy**2)

plt.figure(dpi=300)
subf=plt.subplot(2,2,1); subf.set_title("Orignal image")
plt.imshow(img,cmap='gray'); plt.axis('off') ;
subf=plt.subplot(2,2,2); subf.set_title("Gx")
plt.imshow(Gx,cmap='gray'); plt.axis('off')
subf=plt.subplot(2,2,3); subf.set_title("Gy")
plt.imshow(Gy,cmap='gray'); plt.axis('off')
subf=plt.subplot(2,2,4); subf.set_title("Sobel filter")
plt.imshow(Gm,cmap='gray'); plt.axis('off')
plt.show()

Out=Gm>130 # Thiết lập ngưỡng đơn lọc edge candidate
plt.imshow(Out,cmap='gray'); plt.axis('off')
  plt.title("Egde filter"); plt.show()
```



Egde filter