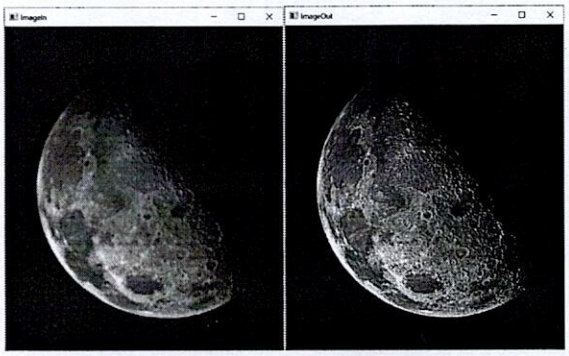
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

KHOA CƠ KHÍ CHẾ TẠO MÁY

BỘ MÔN CƠ ĐIỆN TỬ

ĐÁP ÁN CUỐI KỲ HK I NĂM HỌC 2020-2021
Môn: THỊ GIÁC MÁY
Mã môn học: MAVI332529
Đề số: 01 Đề thi có 02 trang
Ngày thi: 26/01/2021
Thời gian: 75 phút.
Được phép sử dụng tài liệu giấy

Câu 1: (2đ)
Trình bày đạo hàm cấp hai của ảnh. (1đ)
Xây dựng và cài đặt thuật toán làm tăng độ nét của ảnh dùng bộ lọc Laplace cho hình sau. (1đ)



Đáp án:

Using the Second Derivative for Image Sharpening—the Laplacian

In this section we discuss the implementation of 2-D, second-order derivatives and their use for image sharpening. The approach consists of defining a discrete formulation of the second-order derivative and then constructing a filter kernel based on that formulation. As in the case of Gaussian lowpass kernels in Section 3.5 , we are interested here in isotropic kernels, whose response is independent of the direction of intensity discontinuities in the image to which the filter is applied.

We will return to the second derivative in Chapter 10 , where we use it extensively for image segmentation.

It can be shown (Rosenfeld and Kak [1982]) that the simplest isotropic derivative operator (kernel) is the *Laplacian*, which, for a function (image) f(x, y) of two variables, is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3-59}$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator. To express this equation in discrete form, we use the definition in Eq. (3-58) , keeping in mind that we now have a second variable. In the *x*-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \tag{3-60}$$

and, similarly, in the *y*-direction, we have

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \tag{3-61}$$

It follows from the preceding three equations that the discrete Laplacian of two variables is

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{3-62}$$

This equation can be implemented using convolution with the kernel in Fig. 3.51(a) ; thus, the filtering mechanics for image sharpening are as described in Section 3.5 for lowpass filtering; we are simply using different coefficients here.

0	1	0	1	1	1	0	−1	0	−1	−1	−1
1	−4	1	1	−8	1	−1	4	−1	−1	8	−1
0	1	0	1	1	1	0	−1	0	−1	−1	−1

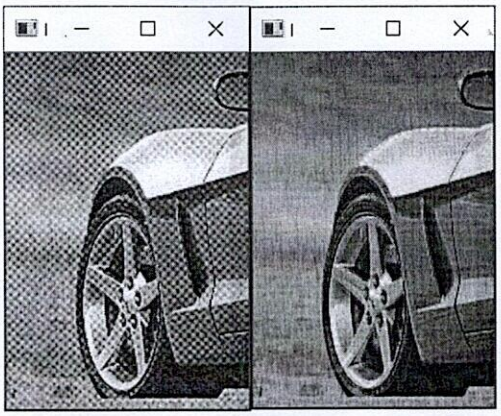
def MySharpen(imgin, imgout):

```
M, N = imgin.shape
m = 3
n = 3
a = m // 2
b = m // 2
w = np.zeros((m,n),np.int32)
w[0,0] = 1
w[0,1] = 1
w[0,2] = 1
w[1,0] = 1
w[1,1] = -8
w[1,2] = 1
w[2,0] = 1
w[2,1] = 1
w[2,2] = 1
for x in range(a, M-a):
    for y in range(b, N-b):
        r = 0
        for s in range(-a, a+1):
            for t in range(-b, b+1):
                r = r + w[s+a, t+b]*imgin[x+s, y+t]
        r = imgin[x,y] - r;
        if r < 0:
            r = 0
        if r > L-1:
            r = L-1
        imgout[x, y] = r


def Sharpen(imgin):
    w = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]], np.int32)
    temp = cv2.filter2D(imgin, cv2.CV_32FC1, w)
    result = imgin - temp
    result = np.clip(result, 0, L-1)
    imgout = result.astype(np.uint8)
    return imgout
```

Câu 2: (3đ)

Xây dựng và cài đặt thuật toán xóa nhiễu moiré dùng bộ lọc chặn khe Butterworth cho hình sau (trình bày vắn tắt):



Đáp án:

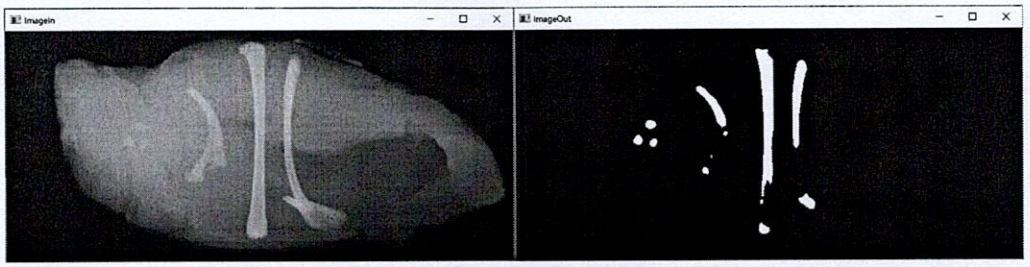
```
def RemoveMoire(imgin):
    # Buoc 1 va 2
    # Mo rong anh co kich thuoc PxQ
    # Them 0 vao phan mo rong
    M, N = imgin.shape
    P = cv2.getOptimalDFTSize(M)
    Q = cv2.getOptimalDFTSize(N)
    f = np.zeros((P,Q), np.float)
    f[0:M,0:N] = imgin

    # Buoc 2: Tinh DFT
    F = cv2.dft(f, flags = cv2.DFT_COMPLEX_OUTPUT)
    # Buoc 3: Shift vao the center of the image
    F = np.fft.fftshift(F)
    # Buoc 4: Tao bo loc H
    H = NotchRejectFilter(P, Q)
    # Buoc 5: G = FH
    G = cv2.mulSpectrums(F, H, flags = cv2.DFT_ROWS)
    # Buoc 6: Shift nguoc tro lai
    G = np.fft.ifftshift(G)
    # Buoc 7: IDFT
    g = cv2.idft(G, flags = cv2.DFT_SCALE)
    # Buoc 8: Get real image with kich thuoc ban dau
    g = g[0:M,0:N,0]

    g = np.clip(g, 0, L-1).astype(np.uint8)
    return g
```

Câu 3: (2đ)

Xây dựng và cài đặt thuật toán đếm và tính các điểm ảnh của các mảnh xương trong miếng thịt gà như hình sau:



Đáp án:

```
def MyConnectedComponent(imgin):
    ret, temp = cv2.threshold(imgin, 200, L-1, cv2.THRESH_BINARY)
    temp = cv2.medianBlur(temp, 7)
    M, N = temp.shape
    dem = 0
    color = 150
    for x in range(0, M):
        for y in range(0, N):
            if temp[x,y] == L-1:
                mask = np.zeros((M+2,N+2),np.uint8)
                cv2.floodFill(temp, mask, (y,x), (color,color,color))
                dem = dem + 1
                color = color + 1
    print('Co %d thanh phan lien thong' % dem)
    a = np.zeros(L, np.int)
    for x in range(0, M):
        for y in range(0, N):
            r = temp[x,y]
            if r > 0:
                a[r] = a[r] + 1
    dem = 1
    for r in range(0, L):
        if a[r] > 0:
            print('%4d   %5d' % (dem, a[r]))
            dem = dem + 1
    return temp

def ConnectedComponent(imgin):
    ret, temp = cv2.threshold(imgin, 200, L-1, cv2.THRESH_BINARY)
    temp = cv2.medianBlur(temp, 7)
    dem, label = cv2.connectedComponents(temp)
    print('Co %d thanh phan lien thong' % (dem-1))

    a = np.zeros(dem, np.int)
    M, N = label.shape
    color = 150
    for x in range(0, M):
        for y in range(0, N):
```

```
            r = label[x, y]
            a[r] = a[r] + 1
            if r > 0:
               label[x,y] = label[x,y] + color

    for r in range(1, dem):
       print('%4d %10d' % (r, a[r]))
    return label.astype(np.uint8)
```

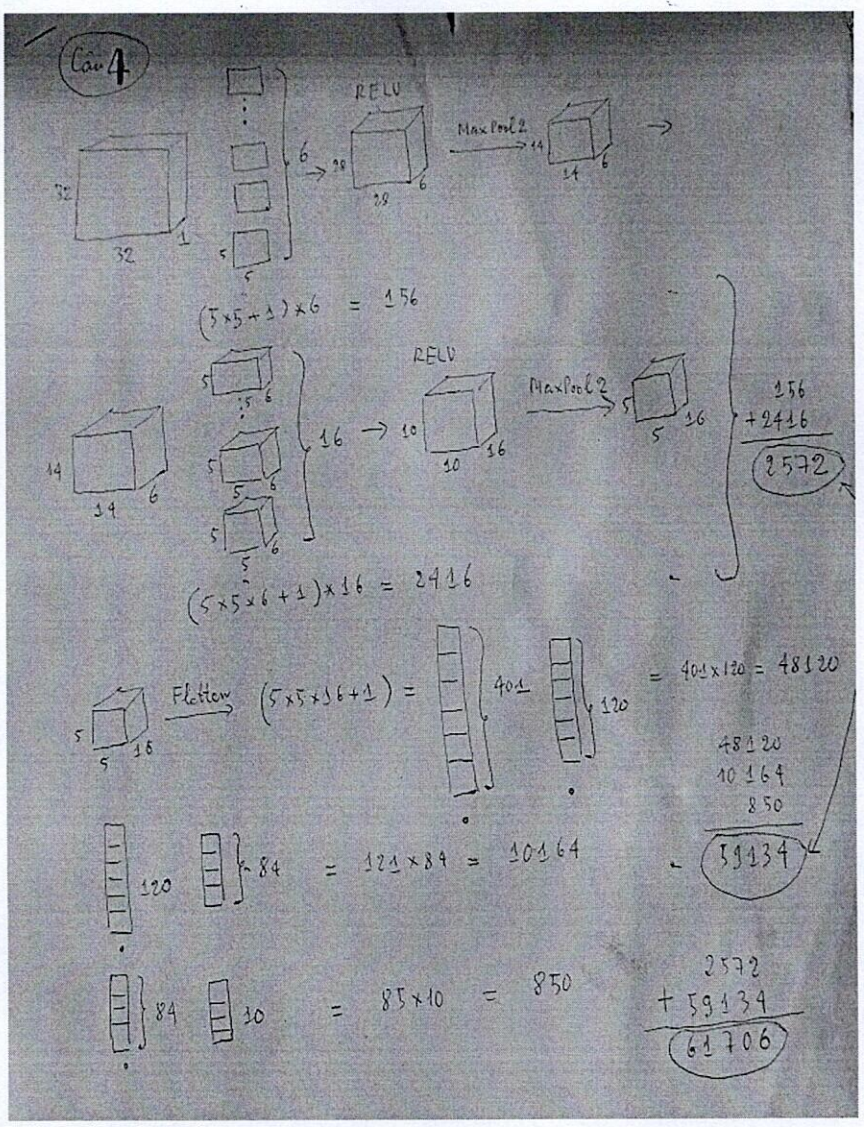
Câu 4: (3đ)

Implementing the LeNet-5 convolutional neural network architecture to classify MNIST Digits as following:

```
def LeNet_5():
    model = Sequential()
    model.add(Conv2D(filters = 6,
                     kernel_size = 5,
                     strides = 1,
                     activation = 'relu',
                     input_shape = (32,32,1)))
    model.add(MaxPooling2D(pool_size = 2, strides = 2))
    model.add(Conv2D(filters = 16,
                     kernel_size = 5,
                     strides = 1,
                     activation = 'relu',
                     input_shape = (14,14,6)))
    model.add(MaxPooling2D(pool_size = 2, strides = 2))
    model.add(Flatten())
    model.add(Dense(units = 120, activation = 'relu'))
    model.add(Dense(units = 84, activation = 'relu'))
    model.add(Dense(units = 10, activation = 'softmax'))
```

a. Draw the network architecture diagram. (1 point)
b. Determine the number of parameters of filters (include the bias node) in the convolutional layers. (1 point)
c. Determine the number of weights of the fully connected layer. (1 point)

Đáp án:



Using TensorFlow backend.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 6)	156
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 6)	0
conv2d_2 (Conv2D)	(None, 10, 10, 16)	2416
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 120)	48120
dense_2 (Dense)	(None, 84)	10164

dense_3 (Dense) (None, 10) 850

==

Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0

----------HẾT----------

Ngày 3 tháng 2 năm 2021

Thông qua bộ môn
(ký và ghi rõ họ tên)

Trần Tiến Đức