Image Processing Application – Spring 2021

Lecturer: Professor Lee Guee Sang

Student: Le Hoai Duy – 218493

**Homework4**

1. Choose a gray scale image.

1) Compute the Laplacian

2) Image - Laplacian --> sharpening

3) Get blurred image and unsharp masking

4) High boost filtering with different parameters

5) Compare the sharpening results and choose the best result. Explain why it results in the best result.

**Code:**

import sys

import cv2 as cv

from matplotlib import pyplot as plt

ddepth = cv.CV\_16S

kernel\_size = 3

src = cv.imread("./moon3.jpg")

# Compute the Laplacian and sharpening by Image - Laplacian

lpc = cv.Laplacian(src, ddepth, ksize=kernel\_size)

sharpening\_laplacian = src-lpc

# Get blurred image and unsharp masking

blurImg = cv.blur(src,(10,10))

unsharp\_masking = src - blurImg

sharpening\_unsharp\_masking = src + unsharp\_masking

# High boost filtering with different parameters

sharpening\_high\_boost2 = src + 2 \* unsharp\_masking

sharpening\_high\_boost3 = src + 3 \* unsharp\_masking

sharpening\_high\_boost4 = src + 4 \* unsharp\_masking

fig, axes = plt.subplots(ncols=3, nrows=2,figsize=(30, 20))

ax0, ax1, ax2, ax3, ax4, ax5 = axes.flat

ax0.imshow(src,cmap='gray')

ax0.set\_title('Original', fontsize=24)

ax0.axis('off')

ax1.imshow(sharpening\_laplacian,cmap='gray')

ax1.set\_title('Laplacian', fontsize=24)

ax1.axis('off')

ax2.imshow(sharpening\_unsharp\_masking,cmap='gray')

ax2.set\_title('Unsharp Masking', fontsize=24)

ax2.axis('off')

ax3.imshow(sharpening\_high\_boost2,cmap='gray')

ax3.set\_title('High boost filtering with A = 2', fontsize=24)

ax3.axis('off')

ax4.imshow(sharpening\_high\_boost3,cmap='gray')

ax4.set\_title('High boost filtering with A = 3', fontsize=24)

ax4.axis('off')

ax5.imshow(sharpening\_high\_boost4,cmap='gray')

ax5.set\_title('High boost filtering with A = 4', fontsize=24)

ax5.axis('off')

fig.savefig("D:/result\_HW4.png",bbox\_inches='tight')

**A picture containing crater, night sky

Description automatically generatedResult:**

**A picture containing outdoor object, crater, star, night sky

Description automatically generated**

All 3 methods Laplacian mask, unsharp mask and high boost filtering increase the contrast of the image as well as sharpen the edges. However, in the unsharp masking method, noise occurs in the background and in the boundary between the object and the background. The high boost filtering method is similar, with noise occurring and noise increasing the bigger the parameter. In conclude, in terms of the criteria of enhancing image contrast, sharpness of details and minimizing noise, the first method gives the best results among 3 methods.

2. For the wine bottle images in the last part of the ppt file.

1) try to sharpen the results (only use Sharpening Spatial Filters)

**Code:**

import sys

import cv2 as cv

from matplotlib import pyplot as plt

ddepth = cv.CV\_16S

kernel\_size = 3

for i in range(4):

    img = cv.imread("./img"+ str(i) +".jpg",1)

    src\_gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

    lpc = cv.Laplacian(src\_gray, cv.CV\_16S, ksize=kernel\_size)

    sharpening\_laplacian = src\_gray-2\*lpc

    # Get blurred image and unsharp masking

    blurImg = cv.blur(src\_gray,(3,3))

    unsharp\_masking = src\_gray - blurImg

    sharpening\_unsharp\_masking = src\_gray + unsharp\_masking

    # High boost filtering with different parameters

    sharpening\_high\_boost2 = src\_gray + 2 \* unsharp\_masking

    sharpening\_high\_boost3 = src\_gray + 3 \* unsharp\_masking

    sharpening\_high\_boost4 = src\_gray + 4 \* unsharp\_masking

    fig, axes = plt.subplots(ncols=3, nrows=1,figsize=(30, 10))

    ax0, ax1, ax2 = axes.flat

    ax0.imshow(src\_gray,cmap='gray')

    ax0.set\_title('Original\_grayscale', fontsize=24)

    ax0.axis('off')

    ax1.imshow(sharpening\_laplacian,cmap='gray')

    ax1.set\_title('Laplacian', fontsize=24)

    ax1.axis('off')

    ax2.imshow(sharpening\_unsharp\_masking,cmap='gray')

    ax2.set\_title('Unsharp Masking', fontsize=24)

    ax2.axis('off')

    fig.savefig("D:/result\_HW4\_1.png",bbox\_inches='tight')

    fig, axes = plt.subplots(ncols=3, nrows=1,figsize=(30, 10))

    ax3, ax4, ax5 = axes.flat

    ax3.imshow(sharpening\_high\_boost2,cmap='gray')

    ax3.set\_title('High boost filtering with A = 2', fontsize=24)

    ax3.axis('off')

    ax4.imshow(sharpening\_high\_boost3,cmap='gray')

    ax4.set\_title('High boost filtering with A = 3', fontsize=24)

    ax4.axis('off')

    ax5.imshow(sharpening\_high\_boost4,cmap='gray')

    ax5.set\_title('High boost filtering with A = 4', fontsize=24)

    ax5.axis('off')

    fig.savefig("D:/result\_HW4\_" + str(i+3) + ".jpg",bbox\_inches='tight')

**Results:**

**A picture containing text

Description automatically generated**

**A picture containing text

Description automatically generated**

**Text

Description automatically generated**

**A picture containing timeline

Description automatically generated**

2) do whatever you can do to improve the result.

Code:

import cv2 as cv

from matplotlib import pyplot as plt

ddepth = cv.CV\_16S

kernel\_size = 3

for i in range(4):

    img = cv.imread("./img"+ str(i) +".jpg",1)

    src\_gray\_origin = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

    img = cv.GaussianBlur(img, (3, 3), 0)

    src\_gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

    lpc = cv.Laplacian(src\_gray, cv.CV\_16S, ksize=kernel\_size)

    sharpening\_laplacian = src\_gray-lpc

    # Get blurred image and unsharp masking

    blurImg = cv.blur(src\_gray,(3,3))

    unsharp\_masking = src\_gray - blurImg

    sharpening\_unsharp\_masking = src\_gray + unsharp\_masking

    # High boost filtering with different parameters

    sharpening\_high\_boost4 = src\_gray + 4 \* unsharp\_masking

    fig, axes = plt.subplots(ncols=3, nrows=1,figsize=(30, 15))

    ax0, ax1, ax2= axes.flat

    ax0.imshow(img,cmap='viridis')

    ax0.set\_title('Original\_color', fontsize=24)

    ax0.axis('off')

#     ax1.imshow(sharpening\_laplacian,cmap='gray')

    ax1.imshow(src\_gray\_origin,cmap='gray')

    ax1.set\_title('Original\_grayscale', fontsize=24)

    ax1.axis('off')

    ax2.imshow(sharpening\_high\_boost4,cmap='gray')

    ax2.set\_title('Sharpen\_image', fontsize=24)

    ax2.axis('off')

    fig.savefig("D:/result\_HW4\_" + str(i+3) + ".jpg",bbox\_inches='tight')

Results:

Text, whiteboard

Description automatically generated

A picture containing text

Description automatically generated

A picture containing text

Description automatically generated

Text, whiteboard

Description automatically generated