S2 6201011631188 โสภณ สุขสมบูรณ์

26 มกราคม 2563

ปฏิบัติการที่ 2 ขนาดของสัญญาณดิจิทัลสองมิติ

การทดลองที่ 2.1 .

2.2.1 ให้นักศึกษาเขียนข้อมูลภาพใหม่จากรูปภาพใบหน้าของตนเองจากปฏิบัติการที่ 1 โดยตั้งชื่อไฟล์ว่า MyCompressedFace.jpeg โดยใช้มาตรฐานการบีบอัดรูปภาพดิจิทัลแบบ jpeg ให้เหลือคุณภาพ เพียง 33%

```
from PIL import Image
import os
import matplotlib.pyplot as plt
import cv2

file_name='MyCompressedFaceV2.jpeg'
picture =Image.open('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg')
picture.save(''+file_name,optimize=True,quality=33)
img2=cv2.imread('MyCompressedFaceV2.jpeg',cv2.IMREAD_GRAYSCALE)

plt.imshow(img2,cmap='gray')
plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

-import Libraryที่ต้องการใช้งาน ในที่นี้ได้แก่ PIL (Image os) , matplotlib.pyplot และ cv2

-ตั้งชื่อไฟล์ที่ต้องการสร้างเมื่อทำการลดqualityของรูปภาพ หรือก็คือตัวแปร file_nameนั่นเอง

-ตั้งชื่อตัวแปรว่า picture เพื่อทำการเปิดรูปภาพจากไฟล์รูปที่ต้องการเปิดด้วยคำสั่ง Image.open

-คำสั่งsaveเป็นการสั่งโปรแกรมให้ทำการบันทึกภาพที่เราได้ทำการลดคุณภาพของภาพนั่นเอง

-สำหรับคำสั่ง cv2.imread เนื่องจากเราต้องการนำเสนอภาพที่เราลดขนาดขึ้นได้ป้อนคำสั่งเพื่อให้โปรแกรมอ่านภาพที่เราทำการลดขนาดไปนั่นเอง ในส่วนนี้ไม่จำเป็นกรณีที่เราไม่ต้องการเช็คภาพดังกล่าว

-เนื่องจากภาพต้นเป็นภาพ จำเป็นต้องใช้คำสั่งcv2.imread_grayscaleเพื่อเปลี่ยนภาพให้เป็นภาพgrayscale

สรุปผล

-ภาพที่ได้จากการป้อนคำสั่งมีความแตกต่างกับต้นฉบับ

-เมื่อทำการรันคำสั่ง โปรแกรมทำการสร้างภาพขึ้นมาใหม่เป็นอีกไฟล์หนึ่งทันที

ด้วยวิธีการนี้ทำให้เราไม่เสียภาพต้นของเราไป และมีภาพใหม่สำหรับใช้งานได้ทันที!

2.2.2 จงนำเสนอรูปต้นแบบและรูปที่ถูกบีบอัดบนหน้าต่างเดียวกันพร้อม โดยกำหนดให้ title แสดงชื่อของ นักศึกษา พร้อมคุณลักษณะขนาดของแต่ละภาพ (size และ dimension) แสดงบน title ด้วย

```
from PIL import Image
import os
import matplotlib.pyplot as plt
import cv2

plt.suptitle('       Sopon Suksomboon', fontweight='bold')

img=cv2.imread('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg',cv2.IMREAD_GRAYSCALE)
Origin_size = (os.path.getsize('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg')/1000)
h1, w1 = img.shape
Orgtitle = ('Original Size = {} kB Height = {} px and Width = {} px '.format(Origin_size,h1,w1))
plt.subplot(121)
plt.imshow(img,cmap='gray')
plt.title(Orgtitle)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')

file_name='MyCompressedFaceV2.jpeg'
picture =Image.open('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg')
picture.save(''+file_name,optimize=True,quality=33)
img2=cv2.imread('MyCompressedFaceV2.jpeg',cv2.IMREAD_GRAYSCALE)
Compressed_size = (os.path.getsize('MyCompressedFaceV2.jpeg')/1000)
h2, w2 = img2.shape
Comptitle = ('Compressed Size = {} kB Height = {} px and Width = {} px '.format(Compressed_size,h1,w1))
plt.subplot(122)
plt.imshow(img2,cmap='gray')
plt.title(Comptitle)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')
plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

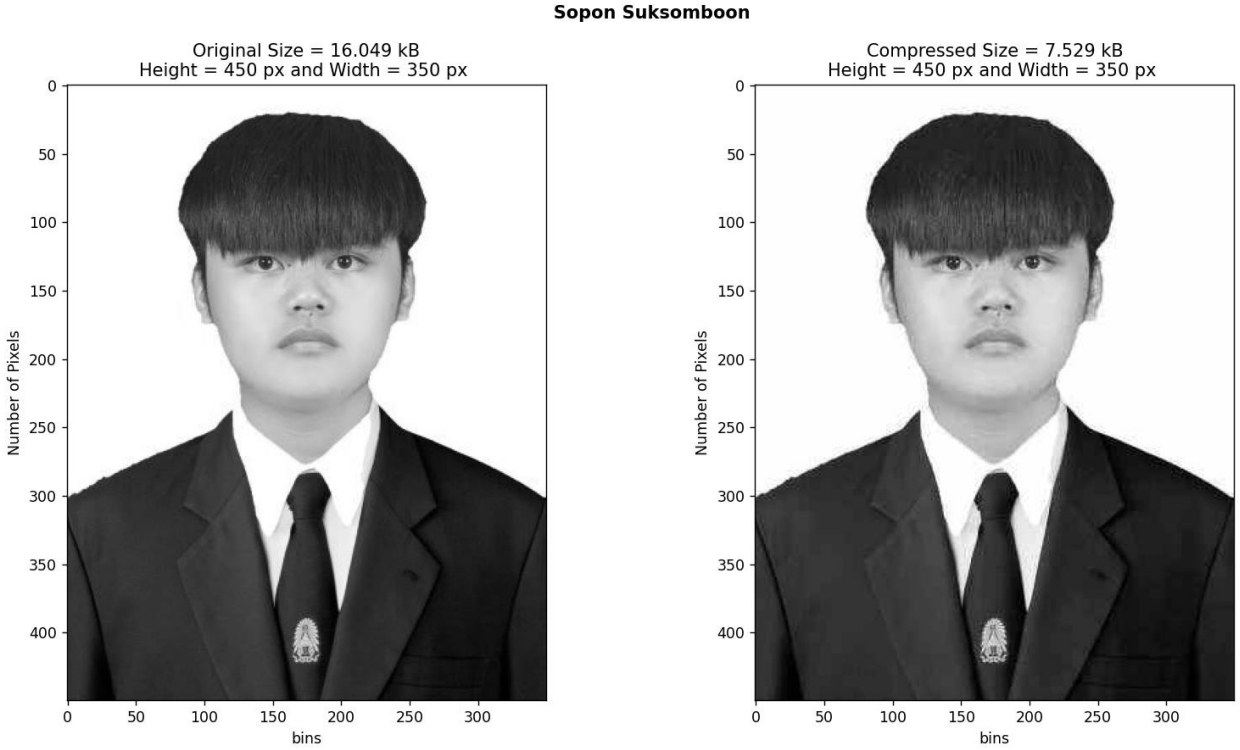
-สำหรับคำสั่ง plt.suptitle และ plt.title เป็นคำสั่งที่เขียนไว้สำหรับบอกชื่อหรือหมายเหตุหรือจั่วหัวของรูปภาพหรือกราฟต่างๆ เช่น ต้องการบอกผู้อ่านว่า รูปทั้งหมดนี้เป็นของตัวผู้เขียน ก็เจียนว่า plt.suptitle(‘Sopon Suksomboon’) หรือ ต้องการจะบอกว่าภาพด้านล่างนั้นเป็นภาพอะไร ก็เขียนคำสั่ง plt.title(‘compressed 33% myfacepic’) แปลว่า ภาพใบหน้านี้ถูกลดคุณภาพ33%

-คำสั่งos.path.getsizeเป็นคำสั่งที่ดึงข้อมูลขนาดของรูปภาพหรือข้อมูลต่างๆมาให้โปรแกรม การ/1000 เพื่อเป็นการเปลี่ยนขนาดจากหน่วย byteเป็นหน่วยkB

-ตัวแปร h1,w1,h2,w2 เป็นตัวแปรที่รับข้อมูลขนาด(ความยาว,ความกว้าง,ความหนา,ความสูง ,อื่นๆ)จากภาพต้นฉบับและภาพที่ผ่านการCompressแล้ว

-อื่นๆ เช่น Orgtitle , Comptitle เป็นตัวแปรที่ไว้ประกาศค่าความกว้าง ความสูง และขนาด รวมทั้ง plt.subplotเป็นคำสั่งที่สร้างขึ้นกรณีที่เราต้องการแสดงผลภาพหรือกราฟหลายรูป

การแสดงผล



สรุปผล

-จากการแสดงผลข้างต้นจะเห็นว่าคุณภาพของภาพหรือsizeมีขนาดลดลงจาก 16.049 kB เป็น 7.529 kB

-ความคมชัดของภาพที่ผ่านการCompressedแล้วนั้นลดลง

-ความกว้าง/ความยาว มีขนาดเท่าเดิมนั่นคือ 350x450

2.1.3 ให้นักศึกษาแสดงฮิสโตแกรมของรูป fullSizeImage และรูป MyCompressedFace.jpeg หน้าต่างเดียวกัน

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
plt.suptitle(Histogram, fontweight='bold')

prototype=cv2.imread('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg')
plt.subplot(121)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')
plt.hist(prototype.ravel(),254,[0,254])
plt.title(' fullSizeImage')

Compressed=cv2.imread('D:\Telecom_Lab\MyCompressedFaceV2.jpeg')
plt.subplot(122)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')
plt.hist(prototype.ravel(),253,[0,253])
plt.title('MyCompressedFace')
plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

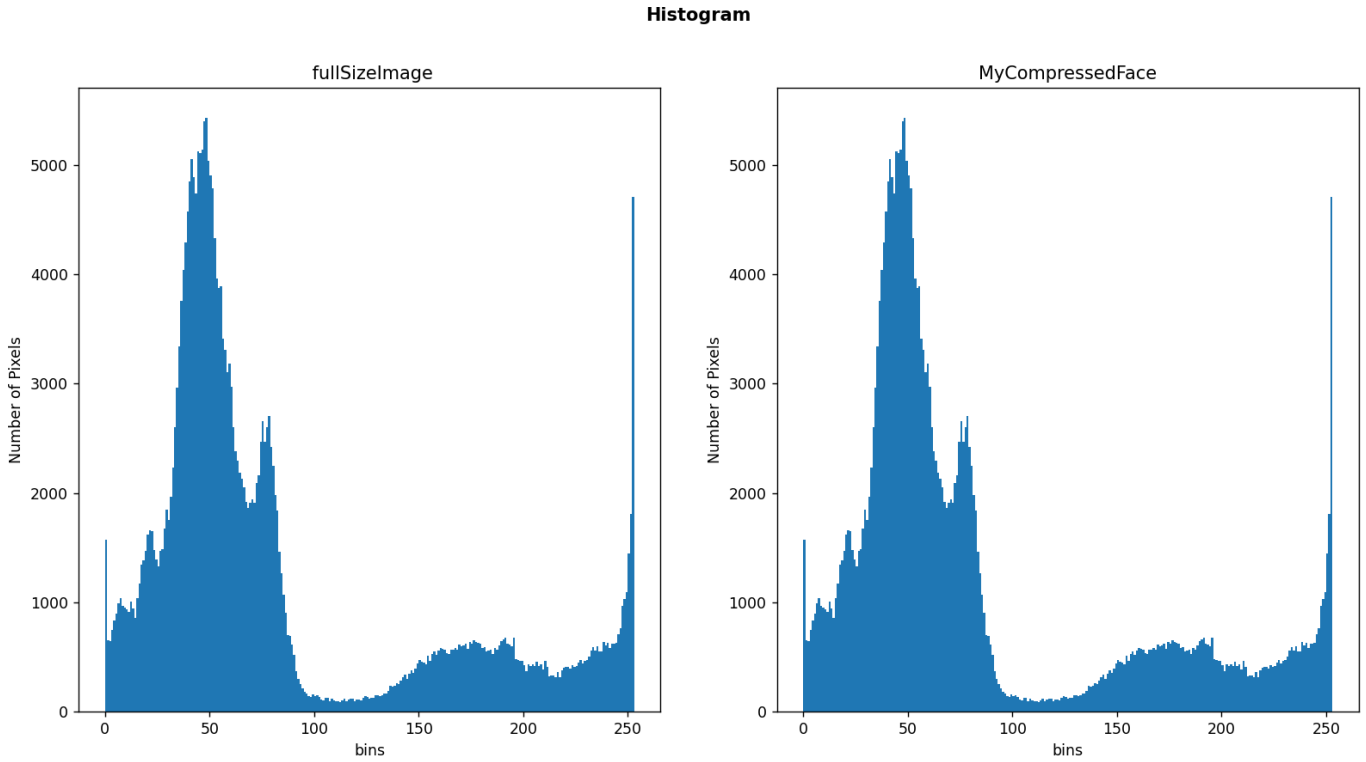
-ตัวแปรPrototypeและCompressedเป็นตัวแปรที่ประกาศมาเพื่อทำการอ่านภาพที่เราต้องการในที่นี้คือภาพต้นฉบับและภาพที่ทำการCompressedแล้วตามลำดับ

-สำหรับคำสั่งplt.subtitle,plt.title เป็นคำสั่งที่เขียนเพื่อกำกับชื่อหรือข้อมูลที่หัวรูปภาพเพื่อบ่งบอกที่มา หมายเหตุ หรือชื่อของภาพดังกล่าว

-สำหรับคำสั่งplt.histนั้นหากเราไม่ต้องการค่าpixelพื้นหลังให้ทำการเช็คค่าpixelจากข้อ2ว่ามีค่าเท่าใดแล้วทำการลดค่าจาก0,256เป็นค่าใหม่ที่ไม่นับพื้นหลัง ในที่นี้ภาพพื้นหลังของภาพต้นฉบับมีค่า255 และภาพที่ผ่านการCompressedมีค่า254 จึงทำการลดค่าจาก 256,[0,256] เป็น 255,[0,255]และ254,[0,254]ตามลำดับ

-ทำการแสดงหน้าต่างด้วยคำสั่ง plt.show()

การแสดงผล



สรุปผล

-มีความแตกต่างเล็กน้อยมองด้วยตาอาจจะเปรียบเทียบลำบาก อาจจะเป็นเพราะจำนานbinsมีจำนวนมากเกินไป หากต้องการเห็นความแตกต่างจริงๆ

จำเป็นต้องลดขนาดbinsเพื่อโฟกัสค่าเฉพาะจุดนั่นเอง

การทดลองที่2.2

2.2.1 ให้นักศึกษาทำการเปลี่ยนขนาดรูป 'cameraman.jpg' โดยให้มีขนาดเล็กลง 10 เท่า แล้วแสดงรูป fullSizeImage และรูป subsampledImage ทั้งสองรูป ใน graphic window เดียวกันโดยปรับ ขนาดให้เห็นผลการ subsample รูป

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

plt.suptitle('Cameraman')
caman=cv2.imread('D:\Telecom_Lab\Lab2\cameraman.jpg')
plt.subplot(1,2,1)
plt.title('fullSizeImage')
plt.imshow(caman,cmap='gray')

camanV2=cv2.resize(caman,None,None,0.1,0.1)
plt.subplot(1,2,2)
plt.title('subsampledImage')
plt.imshow(camanV2,cmap='gray')
plt.show()
#cv2.imwrite('camanV2.jpg',camanV2) #ถ้าอยากสร้างภาพใหม่ใช้คำสั่งนี้
```

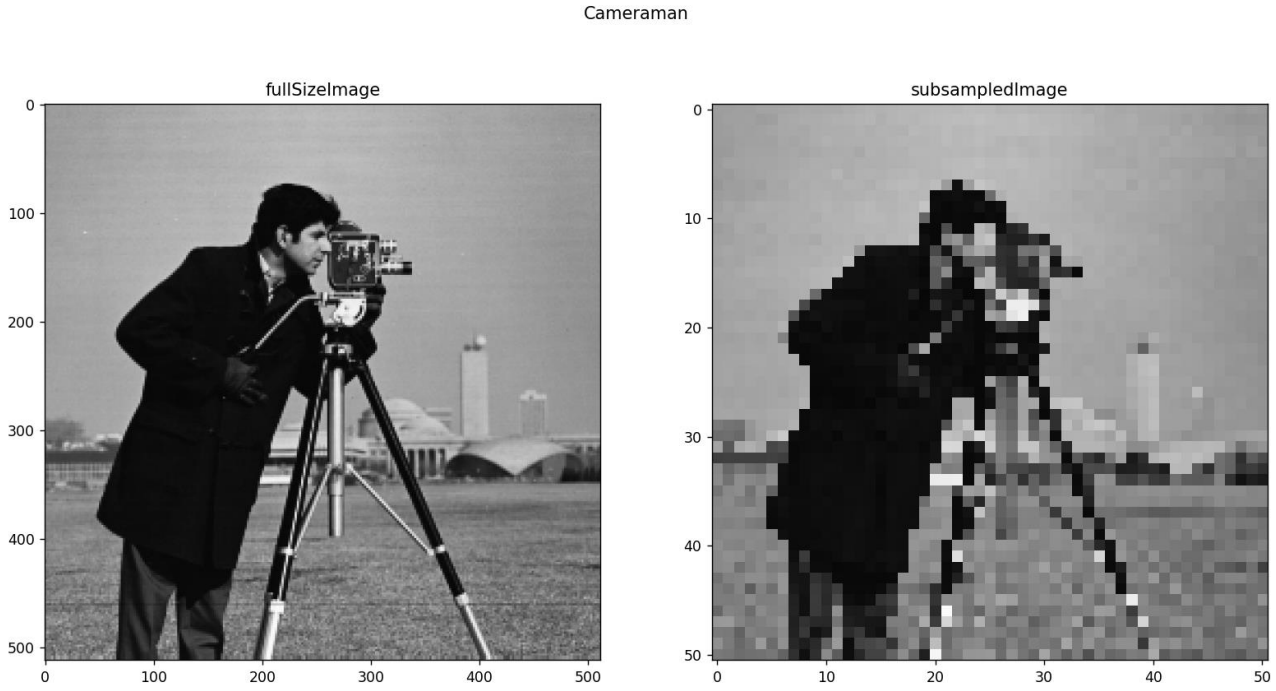
อธิบายชุดคำสั่งที่ใช้งาน

-ตัวแปร caman (ย่อมาจากcameraman) ประกาศขึ้นมาเพื่อรองรับคำสั่งcv2.imreadซึ่งเป็นคำสั่งอ่านภาพนั่นเอง

-ตัวแปร camanV2(ย่อมาจากcameraman Version 2) ประกาศขึ้นมาเพื่อรองรับคำสั่ง cv2.resizeซึ่งเป็นคำสั่งที่ไว้ย่อขนาดรูปโดยที่แต่งจากArgumentข้างในหมายถึง รูปภาพ,ขนาดที่ต้องการของภาพที่ส่งออก(ทั้ง2ช่อง),จำนวนเท่าที่จะลดในแกนx(1/n โดยที่nคือจำนวนเท่า),จำนวนเท่าที่จะลดในแกนy(1/n โดยที่nคือจำนวนเท่า) ตามลำดับ

-สำหรับคำสั่งที่commentไว้นั้นกรณีที่เราต้องการภาพที่เราลดขนาด ก็จะทำการCreateภาพใหม่ให้เรา

การแสดงผล



สรุปผล

-จากหน้าต่างที่แสดงผลนั้น จะเห็นว่าขนาดของภาพเดิมหรือfullSizeImageจะมีขนาด 500x500 เมื่อทำการลดขนาดเป็นจำนวน10เท่า จะเห็นว่าขนาดเหลือเพียง 50x50 ซึ่งแสดงให้เห็นในภาพsubsampleImageนั่นเอง

-จะเห็นว่าภาพsubsampleImageนั้นมีความชัดลดลง ภาพแตก เป็นเพราะขยายขนาดให้เท่ากับภาพต้นนั่นเอง แต่หากมองจากfolderภาพจะเล็กมากจนมองด้วยตาเปล่าไม่เห็นนั่นเอง

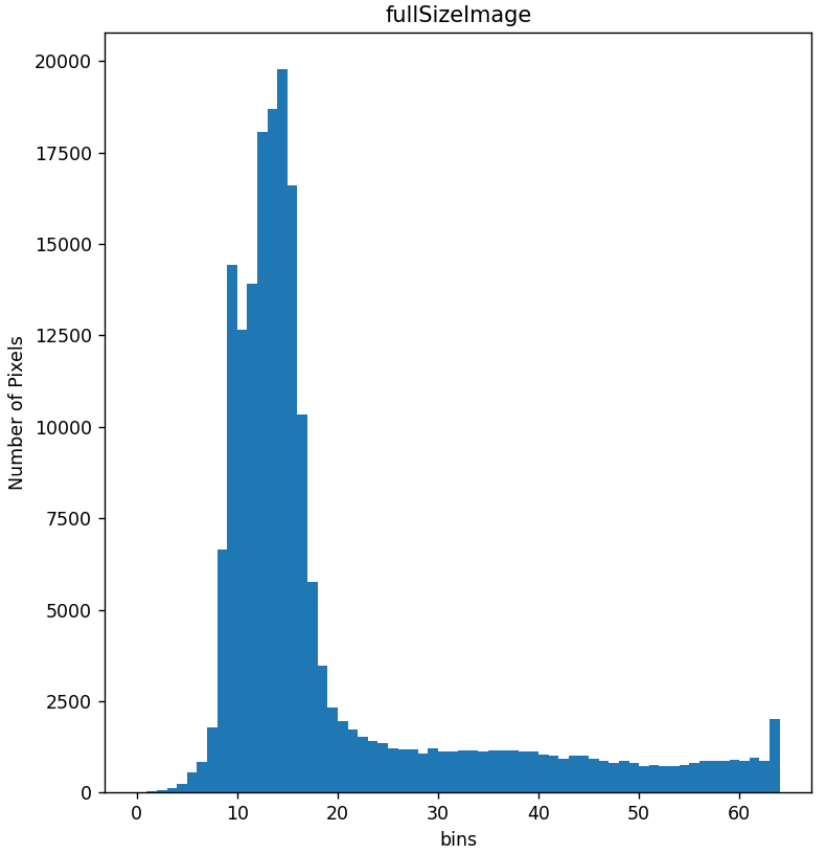
2.2.2 ให้นักศึกษาแสดงฮิสโตแกรมของรูป fullSizeImage และรูป subsampledImage โดยให้ความยาว ของ grayscale bar เท่ากับ 64 bins หน้าต่างเดียวกัน

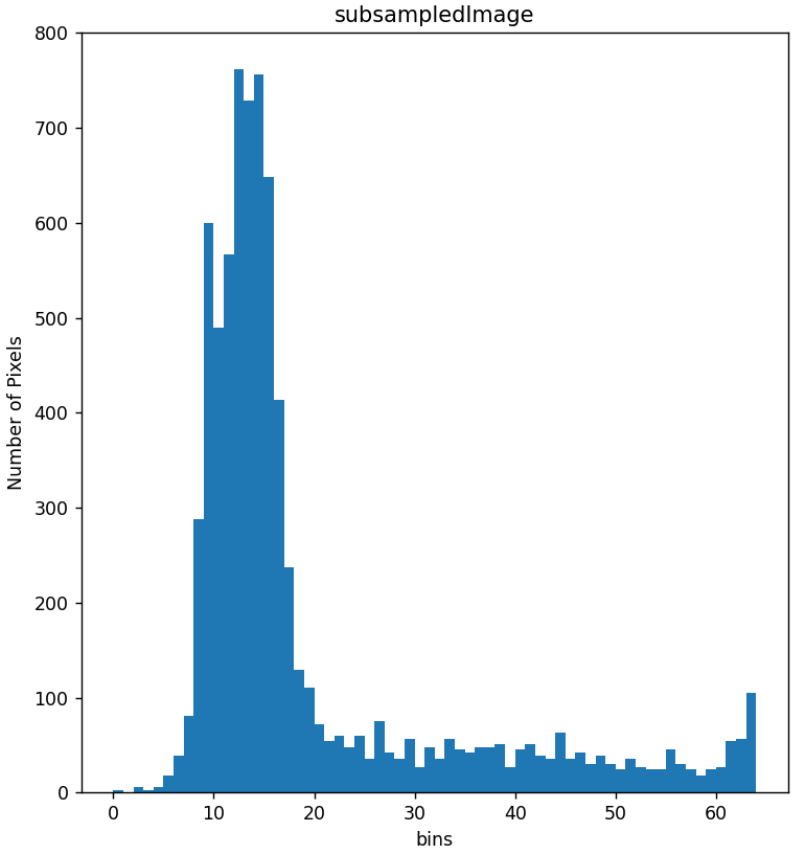
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

caman=cv2.imread('D:\Telecom_Lab\Lab2\cameraman.jpg')
plt.subplot(1,2,1)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')
plt.hist(caman.ravel(),64,[0,64])
plt.title('fullSizeImage')

camanV2=cv2.resize(caman,None,None,0.2,0.2)
plt.subplot(1,2,2)
plt.xlabel('bins')
plt.ylabel('Number of Pixels')
plt.hist(camanV2.ravel(),64,[0,64])
plt.title('subsampledImage')
plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

-คำสั่งในข้อ2.2.2เป็นการดึงคำสั่งของข้อ2.2.1มาแล้วทำการปรับเปลี่ยนบางส่วน โดยคำสั่งที่เพิ่มขึ้นมาคือ plt.hist(...)เป็นคำสั่งสำหรับการแสดงค่าHistogramของภาพในที่นี้เราต้องการค่าbinsแค่64ค่าตามที่โจทย์กำหนด

การแสดงผล





สรุปผล

-ฮิสโทแกรมของภาพต้นฉบับกราฟค่อนข้างSmooth กล่าวคือการแยกแยะค่าbinsและจำนวนpixelนั้นเป็นไปได้ยาก

-ฮิสโทแกรมของภาพที่ทำการresizeค่าbinsบางส่วนมีจำนวนลดลงหรือเห็นได้ชัดขึ้น กราฟไม่มีความsmooth สูงบ้างต่ำบ้าง สืบเนื่องจากจำนวนpixelในภาพที่resizeมีจำนวนลดลงจาก20k pixel เหลือเพียง800pixel นั่นเอง

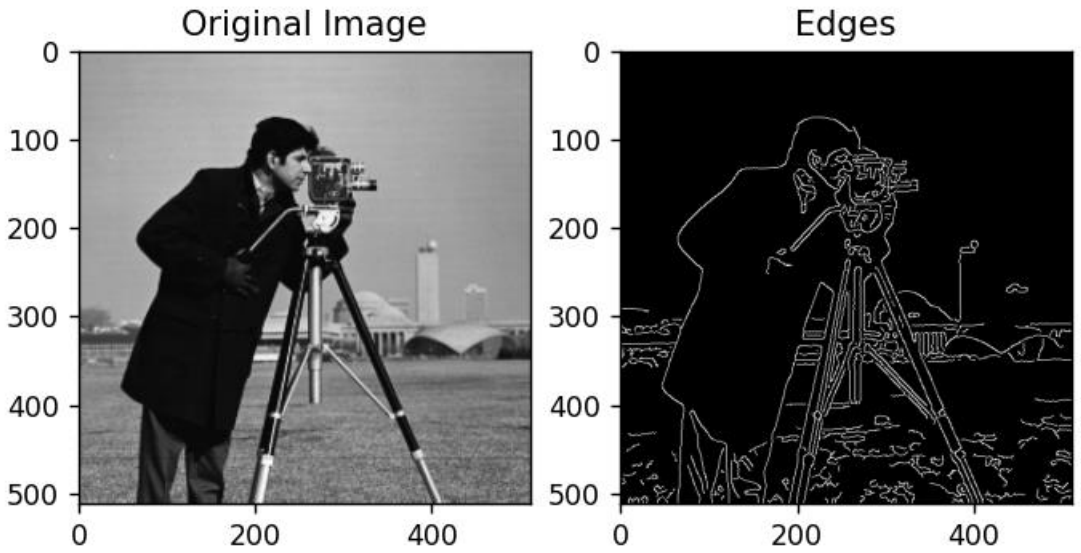
การทดลองที่2.3

2.3.1 ให้นักศึกษาดำเนินคำสั่งต่อไปนี้

```
import cv2
from matplotlib import pyplot as plt
import time

img = cv2.imread('D:\Telecom_Lab\Lab2\cameraman.jpg' , 0)
# set gaussian filter SD and factor of threshold
sigma = 2
factor = 0.75
# blur the image first with gaussian blur using GaussianBlur() function of cv2,
#with kernel size of (7,7) and set sigmaX to our sigma value
smoothedInput = cv2.GaussianBlur(img,(7,7),sigmaX=sigma)
# here use threshold() function of cv2 to find optimal threshold for image using
#Otsu’s binarization, thresholding algorithm can be specified by passing these flags
#to the function “cv.THRESH_BINARY + cv.THRESH_OTSU” for Otsu’s binarization, this
#function will return 2 values which are the calculated threshold and a thresholded
#input image
ret, otsu = cv2.threshold(smoothedInput, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
```

```
#Canny edge detection with Canny() function of cv2, setting lower and higher
#threshold, use lower threshold of 0.4 times
edges = cv2.Canny(smoothedInput, ret * 0.4 * factor, ret * factor)
plt.subplot(121)
plt.imshow(img , cmap = 'gray')
plt.title('Original Image')
plt.subplot(122)
plt.imshow(edges , cmap = 'gray')
plt.title('Edges')

plt.show()
```

การแสดงผล



สรุปผล

-จะเห็นว่าคำสั่ง cv2.canny เป็นคำสั่งที่ไว้สำหรับหาขอบของภาพ ซึ่งการที่จะทำให้การหาขอบชัดขึ้นหรือแยกขอบได้ง่ายขึ้น การเบลอภาพด้วยคำสั่ง cv2.gaussianBlur0จะเป็นตัวช่วยสำคัญในการหาขอบนั่นเอง

2.3.2 ให้นักศึกษาใช้คำสั่งที่เหมาะสมจับเวลาการประมวลผลการหาขอบในข้อ 2.3.1

```
import cv2
from matplotlib import pyplot as plt
import time

start=time.time()
img = cv2.imread('D:\Telecom_Lab\Lab2\cameraman.jpg' , 0)
sigma = 2
factor = 0.75
ret, otsu = cv2.threshold(smoothedInput, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
edges = cv2.Canny(smoothedInput, ret * 0.4 * factor, ret * factor)
end=time.time()
plt.subplot(121)
plt.imshow(img , cmap = 'gray')
plt.title('Original Image')
plt.subplot(122)
plt.imshow(edges , cmap = 'gray')
plt.title('Edges')
print('Operated Time = {} ms '.format((end-start)*1000))

plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

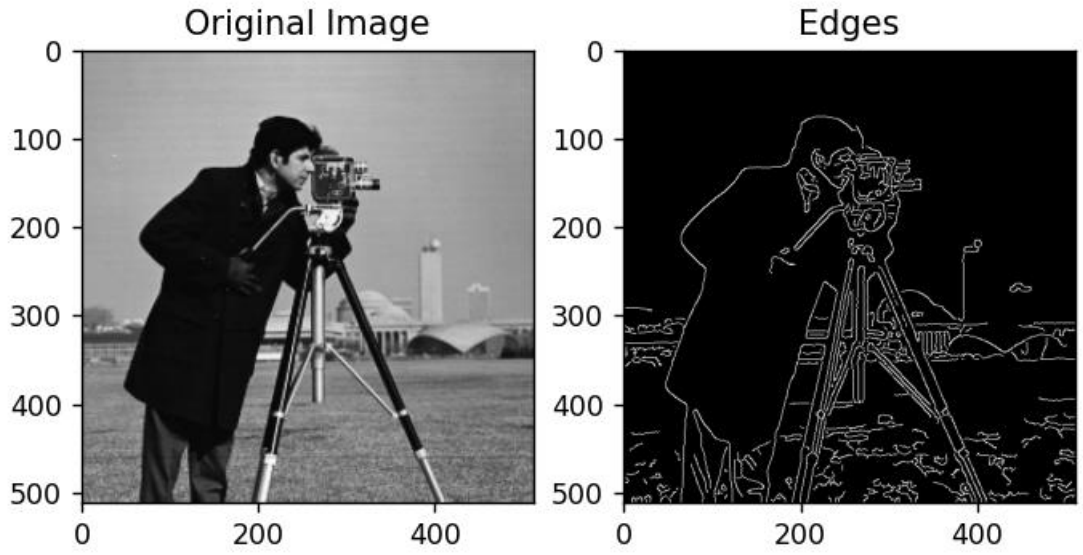
-เนื่องจากคำสั่งส่วนใหญ่ดึงมาจากข้อ2.3.1 สิ่งที่เพิ่มเข้ามาคือคำสั่งสำหรับการจับเวลาในการประมวลคำสั่ง นั่นคือ Library Time นั่นเอง

-สำหรับคำส่งtime จะทำการนับเวลาในการประมวลของคำสั่ง จึงมีตัวแปรตัวที่ประกาศ ซึ่งแต่ละตัวจะทำหน้าที่ดังนี้

-สำหรับตัวแปรstartแทนเวลาเริ่มต้นที่เริ่มทำงาน ตัวแปรendแทนเวลาที่จบการประมวลผลคำสั่ง

-สำหรับคำสั่งprint เป็นการสั่งให้โปรแกรมแสดงผลเวลาขึ้นที่หน้าTheminal นั่นเอง

การแสดงผล




```
Operated Time = 4.985809326171875 ms
```

2.3.3 ให้นักศึกษา subsample รูปภาพจากข้อ 1. แล้วหาขอบด้วยวิธีCanny เหมือนข้อ 1. โดยกำหนดให้ subsampleRate = 5 พร้อมจับเวลาการประมวลผลการหาขอบ

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
plt.suptitle('CameraMan')
start=time.time()
img=cv2.imread('D:\Telecom_Lab\Lab2\cameraman.jpg',0)
camanV2=cv2.resize(img,None,None,0.2,0.2)
canny=cv2.Canny(camanV2,51,51)

titles=('image','Edges by canny')
image=(camanV2,canny)
for i in range(2):
    plt.subplot(1,2,i+1),plt.imshow(image[i],'gray')
    plt.title(titles[i])

end=time.time()
print('Operated Time = {} ms '.format((end-start)*1000))
plt.show()
```

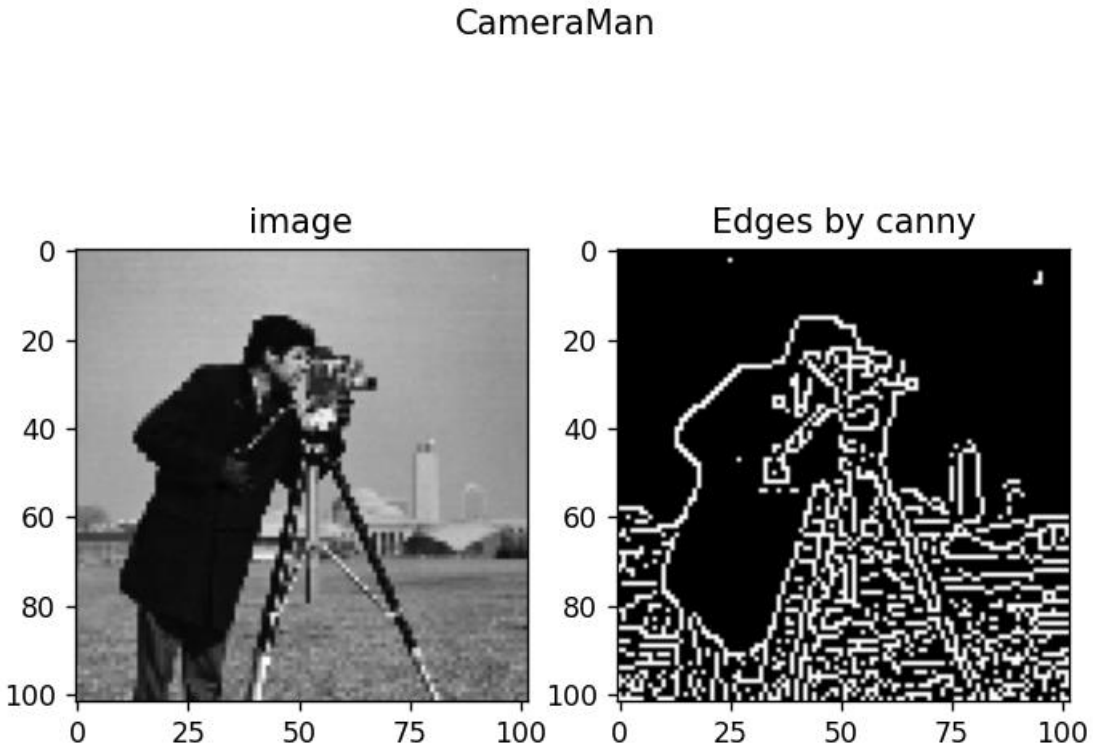
อธิบายชุดคำสั่งที่ใช้งาน

-สำหรับคำสั่ง cv2.canny(camanV2,51,51) ค่า51ทั้งสองแทนความกว้างและความยาวของภาพตามลำดับ

-สำหรับคำสั่ง for loop เป็นการรวมคำสั่งที่ซ้ำกันได้แต่ plt.subplot plt.subtitle plt.xtick และ plt.ytick ที่ซ้ำกัน2ครั้งนั่นเอง ประหยัดเวลาในการเขียนโปรแกรม

-สำหรับคำสั่งprint ที่ทำการ*1000 เพื่อไม่ให้เวลาที่ระบบแจงให้ทราบนั้นเป็นหน่วยวินาที เนื่องจากทศนิยมมีหลายตัวทำให้สับสนได้ จึงบอกในหน่วย ms แทน

การแสดงผล



สรุปผล

-จากการเปรียบเทียบเวลาจากข้อ2.3.2 เวลาที่ใช้ในการประมวลผลในการหาขอบของข้อ2.3.3 ใช้เวลานานกว่ามาก อาจเป็นเพราะภาพมีความละเอียดลดลง หรืออาจจะเป็นเพราะปัจจัยอื่นๆ

4. ให้นักศึกษานำรูปใบหน้าของตนเองจากการทดลองที่ 2.1 ทั้งสองรูปมาหาขอบและจับเวลา

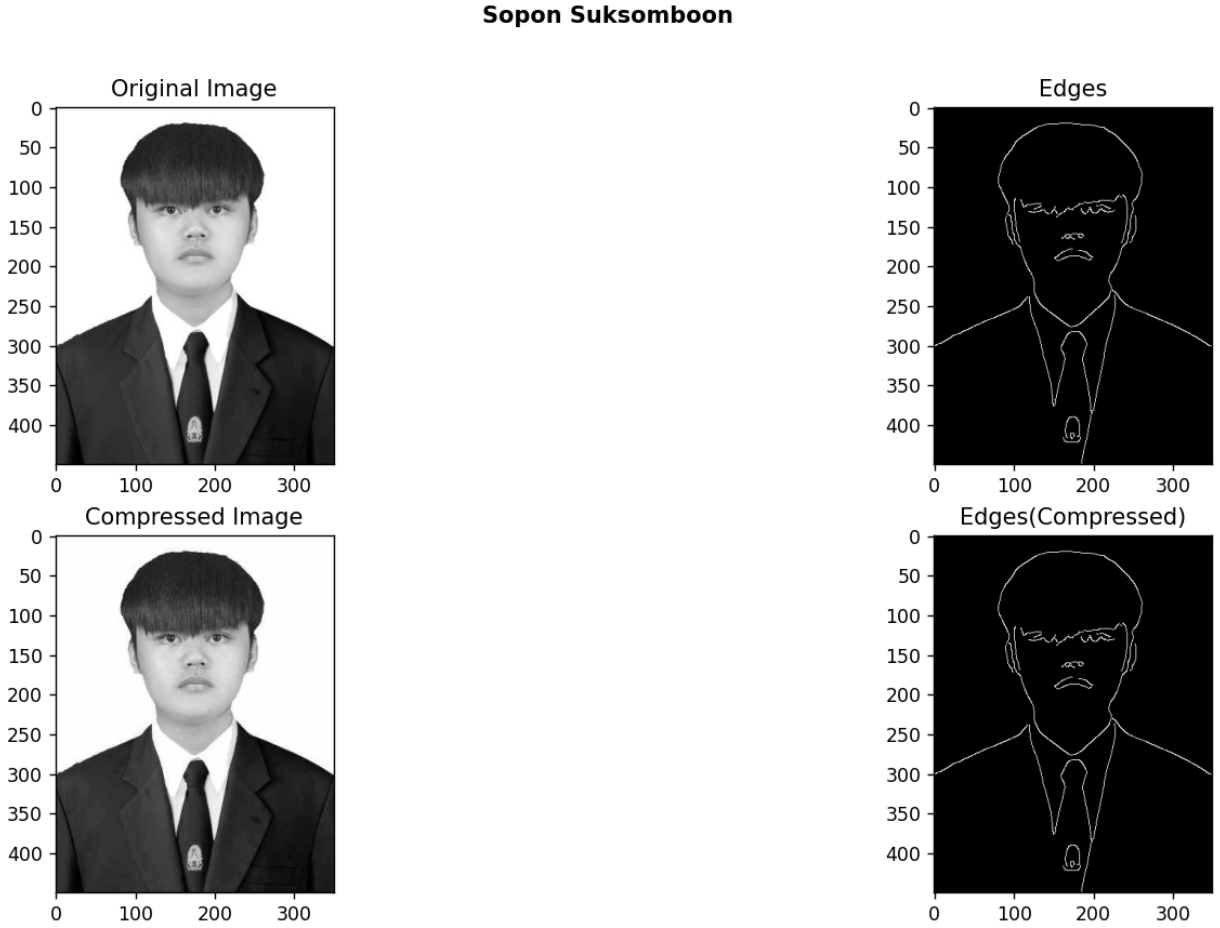
```
import cv2
from matplotlib import pyplot as plt
import time

start=time.time()
plt.suptitle('Sopon Suksomboon', fontweight='bold')
img = cv2.imread('D:\Telecom_Lab\Lab2\MyCompressedFace.jpeg' ,cv2.IMREAD_GRAYSCALE)
img2=cv2.imread('MyCompressedFaceV2.jpeg',cv2.IMREAD_GRAYSCALE)

sigma = 2
factor = 0.75

smoothedInput_Origin = cv2.GaussianBlur(img,(7,7),sigmaX=sigma)
smoothedInput_Compressed = cv2.GaussianBlur(img2,(7,7),sigmaX=sigma)

ret, otsu = cv2.threshold(smoothedInput_Origin, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
ret, otsu = cv2.threshold(smoothedInput_Compressed, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
edges_Origin = cv2.Canny(smoothedInput_Origin, ret * 0.4 * factor, ret * factor)
edges_Compressed = cv2.Canny(smoothedInput_Compressed, ret * 0.4 * factor, ret * factor)
```

```
end=time.time()
plt.subplot(221)
plt.imshow(img , cmap = 'gray')
plt.title('Original Image')
plt.subplot(222)
plt.imshow(edges_Origin , cmap = 'gray')
plt.title('Edges')
plt.subplot(223)
plt.imshow(img2,cmap='gray')
plt.title('Compressed Image')
plt.subplot(224)
plt.imshow(edges_Origin,cmap='gray')
plt.title('Edges(Compressed)')
print('Operated Time = {} ms '.format((end-start)*1000))

plt.show()
```

อธิบายชุดคำสั่งที่ใช้งาน

-คำสั่งที่ใช้เป็นคำสั่งชุดเดียวกับข้อ2.3.1 แต่มีการเพิ่มจำนวนตัวแปรมากขึ้น เนื่องจากเราต้องทำการเปรียบเทียบทั้งภาพเดิม(CompressedMyFace.jpeg)และภาพที่ผ่านการCompressed(MyCompressedV2) นั่นเอง ทำให้ตัวแปรเพิ่มขึ้นนั่นเอง

การแสดงผล




```
Operated Time = 115.86403846740723 ms
```

สรุปผล

-ภาพตัดขอบของภาพทั้งสองแทบไม่แตกต่างกัน หรืออาจจะเกิดความแตกต่างแต่ดวงตามนุษย์ไม่สามารถมองเห็นได้ อาจจะต้องมีการปรับสเกลให้มองง่ายขึ้น หรือกำหนดค่าสัมประสิทธิ์ใหม่เพื่อให้มองเห็นความแตกต่างได้ชัดเจนมากขึ้น

-เวลาที่ใช้ในการประมวลผล ค่อนข้างนาน เมื่องเทียบกับภาพCameraman อาจเป็นเพราะมีความละเอียดมากกว่า หรือจำนวนภาพที่ใช้ประมวลผลมากกว่านั่นเอง หรือด้วยเหตุผลอื่นๆ เป็นต้น