S2 6201011631188 โสภณ สุขสมบูรณ์

16 กุมภาพันธ์ 2565

ปฏิบัติการที่ 5 Gaussian Mixture Model

การทดลองที่ 1

การทดลองที่1.1 พิจารณา histogram ของภาพ แล้วประมาณว่าจะต้องใช้จำนวนของ Gaussian distribution เท่าใด ใส่จำนวนที่ประมาณลงไปในไฟล์ Parameter_Estimation.py โดยใส่ค่าที่ตัวแปร C และ ปรับจำนวนของ command lines ให้สอดคล้องกับจำนวน Gaussian distribution ทุกที่ในไฟล์ Parameter_Estimation.py (ดู comment lines ประกอบ) หมายเหตุหากใส่ค่า C ที่มีค่ามาก (ใช้Gaussian distribution มากเกินความจำเป็น) อาจทำให้เกิดการประมวลผลที่ล่าช้า ทำการทดลองซ้ำ โดยปรับค่า epsilon ในไฟล์ Parameter_Estimation.py ให้มีค่าระหว่าง (1.0e-4,1.0e-3) ที่แตกต่างกันสามค่า แล้วเลือกค่าที่เหมาะสม ทำการทดลองซ้ำ โดยปรับเปลี่ยนค่า C ให้เหมาะสมกับค่า epsilon ที่ได้เลือกไว้ แล้วนำเสนอผลการทดลองพร้อมอภิปรายและให้เหตุผลข้อสรุป

```
from PIL import Image
import numpy as np
img = Image.open('D:\Telecom_Lab\Lab5\Figure\MyFacePic.jpg') # image extension *.png,*.jpg
new_width  = 270
new_height = 480
img = img.resize((new_width, new_height))
img.save('D:\Telecom_Lab\Lab5\Figure\MyFacePic_Resize.jpg') # format may what u want
,*.png,*jpg,*.gif
from skimage.io import imread
from skimage.color import rgb2gray
mountain_r = rgb2gray(imread('D:\Telecom_Lab\Lab5\Figure\MyFacePic_Resize.jpg'))
```

```
#Plot
import matplotlib.pyplot as plt
plt.figure(0)
plt.imshow(mountain_r,cmap="gray")
plt.show()
import cv2
img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\MyFacePic_Resize.jpg',0)
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
meandata = mean(data)
stddata = np.std(data)
x= meandata+(4*stddata)

c = np.arange(0, x, 1)

k = len(c)
i = len(data)
plt.figure(1)
hist,bin = np.histogram(data,c)
y = len(hist)
w = np.arange(0, x-1, 1)
r = hist/i
plt.bar(w,r)
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Probability density function')
plt.show()

plt.figure(2)
plt.hist(img.ravel(),256,[0,256])
```

```
plt.ylabel('Number of pixel')
plt.xlabel('Intensity value')
plt.plot([24,24],[0,3000],'-k')
plt.plot([58.22,58.22],[0,3000],'-k')
plt.title('Histogram')
plt.show()

img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\MyFacePic_Resize.jpg',0)
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]

m = len(data)
epsilon = 1.0e-3
difference = epsilon
counter = 0
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
c = 21

from numpy.random import seed
from numpy.random import rand

seed(1)

mu_est = 2*mean(data)*np.sort(rand(c,1))

sigma_est = np.ones(c)*np.std(data)

p_est = np.ones(c)/c

def gaussian_norm_density(x, mu, sig):
```

```
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))/(sig * np.sqrt(2 *
np.pi))
d = max(data)

xl = np.arange(0, d*3,0.1)


p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[3] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
p6_est = p_est[4] * gaussian_norm_density(xl, mu_est[5], sigma_est[5]);
p7_est = p_est[6] * gaussian_norm_density(xl, mu_est[6], sigma_est[6]);
p8_est = p_est[7] * gaussian_norm_density(xl, mu_est[7], sigma_est[7]);
p9_est = p_est[8] * gaussian_norm_density(xl, mu_est[8], sigma_est[8]);
p10_est = p_est[9] * gaussian_norm_density(xl, mu_est[9], sigma_est[9]);
p11_est = p_est[10] * gaussian_norm_density(xl, mu_est[10], sigma_est[10]);
p12_est = p_est[11] * gaussian_norm_density(xl, mu_est[11], sigma_est[11]);
p13_est = p_est[12] * gaussian_norm_density(xl, mu_est[12], sigma_est[12]);
p14_est = p_est[13] * gaussian_norm_density(xl, mu_est[13], sigma_est[13]);
p15_est = p_est[14] * gaussian_norm_density(xl, mu_est[14], sigma_est[14]);
p16_est = p_est[15] * gaussian_norm_density(xl, mu_est[15], sigma_est[15]);
p17_est = p_est[16] * gaussian_norm_density(xl, mu_est[16], sigma_est[16]);
p18_est = p_est[17] * gaussian_norm_density(xl, mu_est[17], sigma_est[17]);
p19_est = p_est[18] * gaussian_norm_density(xl, mu_est[18], sigma_est[18]);
p20_est = p_est[19] * gaussian_norm_density(xl, mu_est[19], sigma_est[19]);
p21_est = p_est[20] * gaussian_norm_density(xl, mu_est[20], sigma_est[20]);
plt.figure(3)
plt.plot(xl,p1_est+p2_est+p3_est+p4_est+p5_est+p6_est+p7_est+p8_est+p9_est+p10_est+p11_est
+p12_est+p13_est+p14_est+p15_est+p16_est+p17_est+p18_est+p19_est+p20_est+p21_est, 'r--
',linewidth=2.0)
```

```
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
plt.plot(xl, p6_est, 'g-.', linewidth=2.0);
plt.plot(xl, p7_est, 'g-.', linewidth=2.0);
plt.plot(xl, p8_est, 'g-.', linewidth=2.0);
plt.plot(xl, p9_est, 'g-.', linewidth=2.0);
plt.plot(xl, p10_est, 'g-.', linewidth=2.0);
plt.plot(xl, p11_est, 'g-.', linewidth=2.0);
plt.plot(xl, p12_est, 'g-.', linewidth=2.0);
plt.plot(xl, p13_est, 'g-.', linewidth=2.0);
plt.plot(xl, p14_est, 'g-.', linewidth=2.0);
plt.plot(xl, p15_est, 'g-.', linewidth=2.0);
plt.plot(xl, p16_est, 'g-.', linewidth=2.0);
plt.plot(xl, p17_est, 'g-.', linewidth=2.0);
plt.plot(xl, p18_est, 'g-.', linewidth=2.0);
plt.plot(xl, p19_est, 'g-.', linewidth=2.0);
plt.plot(xl, p20_est, 'g-.', linewidth=2.0);
plt.plot(xl, p21_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Gaussian Weight Distribution')
plt.show()

clas = []
ok = []
while np.any(difference >= epsilon) and (counter < 25000):
    for j in range(0, c):

        clas.insert(j, p_est[j] * gaussian_norm_density(data, mu_est[j], sigma_est[j]))
```

```
    ok =
clas[0]+clas[1]+clas[2]+clas[3]+clas[4]+clas[5]+clas[6]+clas[7]+clas[8]+clas[9]+clas[10]+c
las[11]+clas[12]+clas[13]+clas[14]+clas[15]+clas[16]+clas[17]+clas[18]+clas[19]+clas[20]

    for j in range(0, c):
        clas[j] = clas[j] / ok


    mu_est_old = mu_est
    sigma_est_old = sigma_est
    p_est_old = p_est
    mu_est = []
    sigma_est = []
    p_est = []

    for j in range(0, c):
        mu_est.insert(j, sum((clas[j]) * data) / sum(clas[j]))
        sigma_est.insert(j, np.sqrt(sum((clas[j]) * np.power((data - mu_est[j]), 2)) /
sum(clas[j])))
        p_est.insert(j, mean(clas[j]))

    difference =
sum(abs(np.subtract(mu_est_old,mu_est)))+sum(abs(np.subtract(sigma_est_old,sigma_est)))\
                +sum(abs(np.subtract(p_est_old,p_est)))

    print(difference)

    counter = counter + 800
    print('counter =',counter)
xl = np.arange(0, d, 0.1)
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
```

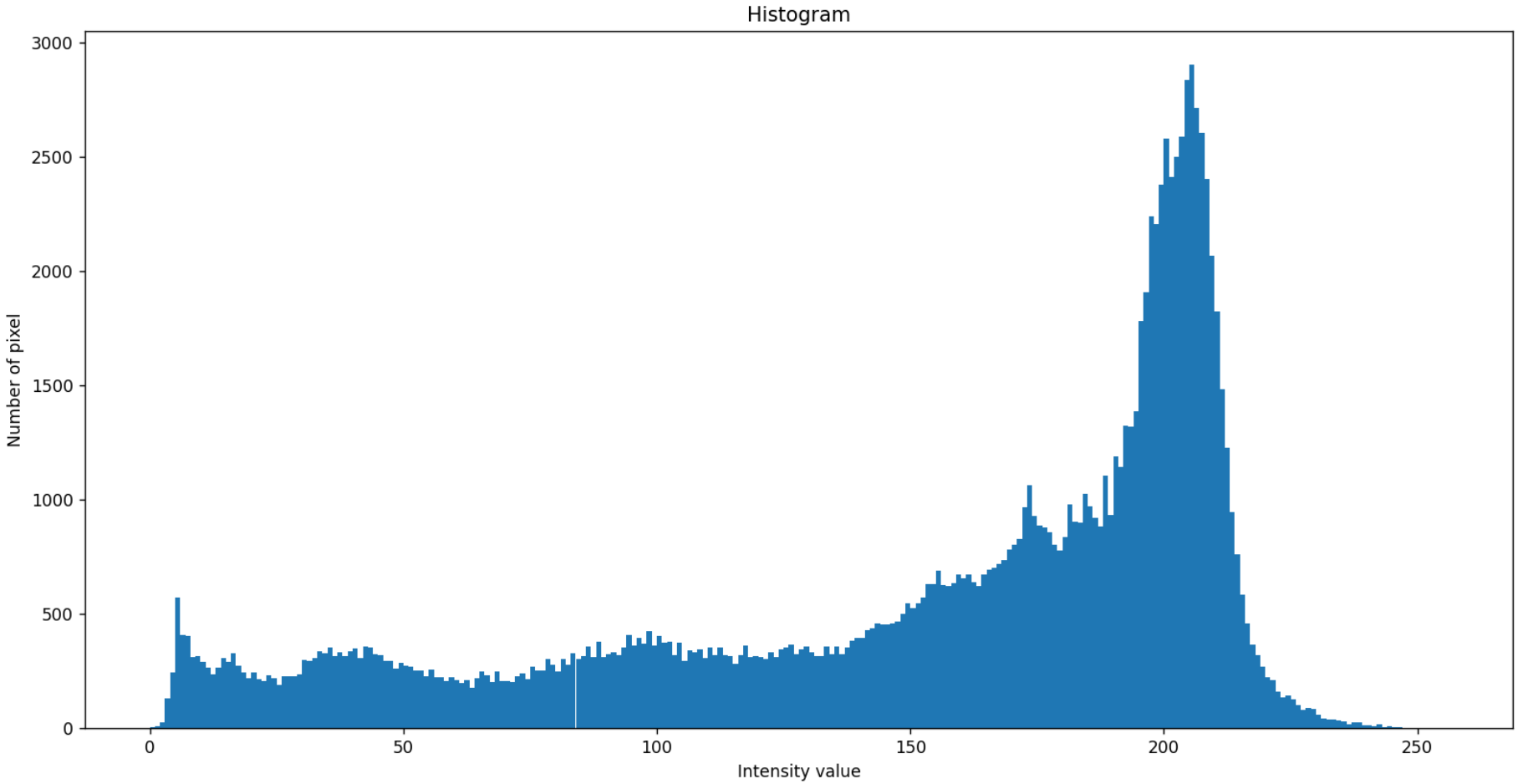
```
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
p6_est = p_est[5] * gaussian_norm_density(xl, mu_est[5], sigma_est[5]);
p7_est = p_est[6] * gaussian_norm_density(xl, mu_est[6], sigma_est[6]);
p8_est = p_est[7] * gaussian_norm_density(xl, mu_est[7], sigma_est[7]);
p9_est = p_est[8] * gaussian_norm_density(xl, mu_est[8], sigma_est[8]);
p10_est = p_est[9] * gaussian_norm_density(xl, mu_est[9], sigma_est[9]);
p11_est = p_est[10] * gaussian_norm_density(xl, mu_est[10], sigma_est[10]);
p12_est = p_est[11] * gaussian_norm_density(xl, mu_est[11], sigma_est[11]);
p13_est = p_est[12] * gaussian_norm_density(xl, mu_est[12], sigma_est[12]);
p14_est = p_est[13] * gaussian_norm_density(xl, mu_est[13], sigma_est[13]);
p15_est = p_est[14] * gaussian_norm_density(xl, mu_est[14], sigma_est[14]);
p16_est = p_est[15] * gaussian_norm_density(xl, mu_est[15], sigma_est[15]);
p17_est = p_est[16] * gaussian_norm_density(xl, mu_est[16], sigma_est[16]);
p18_est = p_est[17] * gaussian_norm_density(xl, mu_est[17], sigma_est[17]);
p19_est = p_est[18] * gaussian_norm_density(xl, mu_est[18], sigma_est[18]);
p20_est = p_est[19] * gaussian_norm_density(xl, mu_est[19], sigma_est[19]);
p21_est = p_est[20] * gaussian_norm_density(xl, mu_est[20], sigma_est[20]);
plt.figure(4)
plt.plot(xl, p1_est + p2_est + p3_est +
p4_est+p5_est+p6_est+p7_est+p8_est+p9_est+p10_est+p11_est+p12_est+p13_est+p14_est+p15_est+
p16_est+p17_est+p18_est+p19_est+p20_est+p21_est, 'r--', linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
plt.plot(xl, p6_est, 'g-.', linewidth=2.0);
plt.plot(xl, p7_est, 'g-.', linewidth=2.0);
plt.plot(xl, p8_est, 'g-.', linewidth=2.0);
```

```
plt.plot(xl, p9_est, 'g-.', linewidth=2.0);
plt.plot(xl, p10_est, 'g-.', linewidth=2.0);
plt.plot(xl, p11_est, 'g-.', linewidth=2.0);
plt.plot(xl, p12_est, 'g-.', linewidth=2.0);
plt.plot(xl, p13_est, 'g-.', linewidth=2.0);
plt.plot(xl, p14_est, 'g-.', linewidth=2.0);
plt.plot(xl, p15_est, 'g-.', linewidth=2.0);
plt.plot(xl, p16_est, 'g-.', linewidth=2.0);
plt.plot(xl, p17_est, 'g-.', linewidth=2.0);
plt.plot(xl, p18_est, 'g-.', linewidth=2.0);
plt.plot(xl, p19_est, 'g-.', linewidth=2.0);
plt.plot(xl, p20_est, 'g-.', linewidth=2.0);
plt.plot(xl, p21_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([24,24],[0,0.012],'-k')
plt.plot([58.22,58.22],[0,0.012],'-k')

plt.show()

p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
p6_est = p_est[5] * gaussian_norm_density(xl, mu_est[5], sigma_est[5]);
p7_est = p_est[6] * gaussian_norm_density(xl, mu_est[6], sigma_est[6]);
p8_est = p_est[7] * gaussian_norm_density(xl, mu_est[7], sigma_est[7]);
p9_est = p_est[8] * gaussian_norm_density(xl, mu_est[8], sigma_est[8]);
p10_est = p_est[9] * gaussian_norm_density(xl, mu_est[9], sigma_est[9]);
p11_est = p_est[10] * gaussian_norm_density(xl, mu_est[10], sigma_est[10]);
p12_est = p_est[11] * gaussian_norm_density(xl, mu_est[11], sigma_est[11]);
```

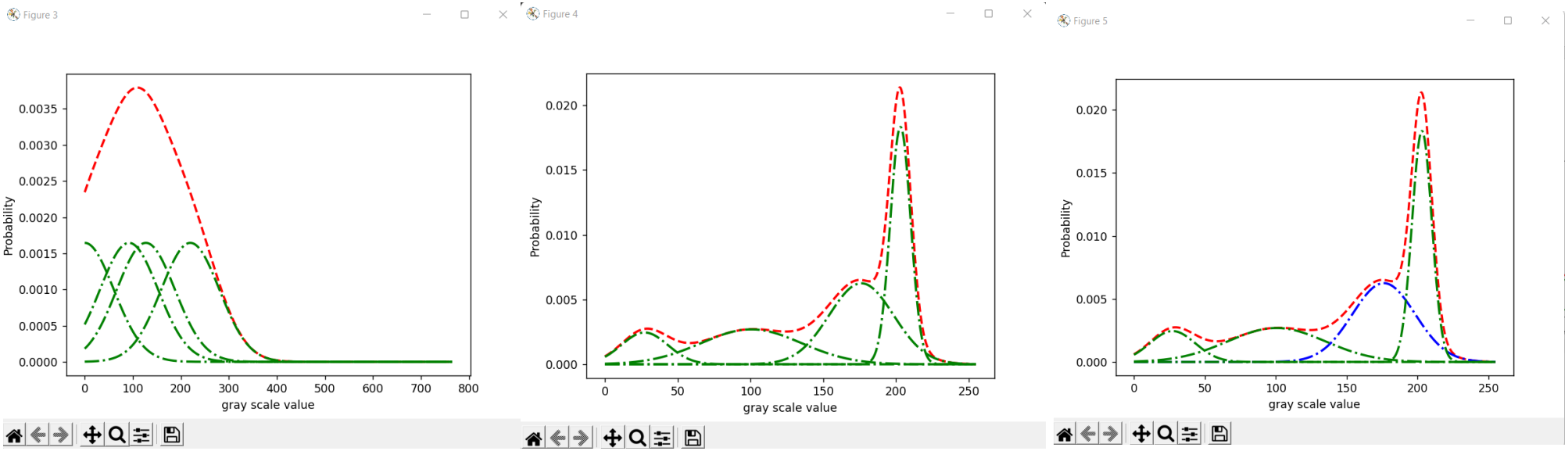
```
p13_est = p_est[12] * gaussian_norm_density(xl, mu_est[12], sigma_est[12]);
p14_est = p_est[13] * gaussian_norm_density(xl, mu_est[13], sigma_est[13]);
p15_est = p_est[14] * gaussian_norm_density(xl, mu_est[14], sigma_est[14]);
p16_est = p_est[15] * gaussian_norm_density(xl, mu_est[15], sigma_est[15]);
p17_est = p_est[16] * gaussian_norm_density(xl, mu_est[16], sigma_est[16]);
p18_est = p_est[17] * gaussian_norm_density(xl, mu_est[17], sigma_est[17]);
p19_est = p_est[18] * gaussian_norm_density(xl, mu_est[18], sigma_est[18]);
p20_est = p_est[19] * gaussian_norm_density(xl, mu_est[19], sigma_est[19]);
p21_est = p_est[20] * gaussian_norm_density(xl, mu_est[20], sigma_est[20]);
#sum_est = p1_est + p2_est + p3_est + p4_est+p5_est

plt.figure(5)

plt.plot(xl, p1_est + p2_est + p3_est + p4_est+p5_est+p6_est+p7_est+p8_est+p9_est+p10_est
+p11_est+p12_est+p13_est+p14_est+p15_est+p16_est+p17_est+p18_est+p19_est+p20_est+p21_est,
'r--', linewidth=2.0)

plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
plt.plot(xl, p6_est, 'g-.', linewidth=2.0);
plt.plot(xl, p7_est, 'g-.', linewidth=2.0);
plt.plot(xl, p8_est, 'g-.', linewidth=2.0);
plt.plot(xl, p9_est, 'g-.', linewidth=2.0);
plt.plot(xl, p10_est, 'g-.', linewidth=2.0);
plt.plot(xl, p11_est, 'g-.', linewidth=2.0);
plt.plot(xl, p12_est, 'g-.', linewidth=2.0);
plt.plot(xl, p13_est, 'g-.', linewidth=2.0);
plt.plot(xl, p14_est, 'g-.', linewidth=2.0);
plt.plot(xl, p15_est, 'g-.', linewidth=2.0);
```

```
plt.plot(xl, p16_est, 'g-.', linewidth=2.0);
plt.plot(xl, p17_est, 'g-.', linewidth=2.0);
plt.plot(xl, p18_est, 'g-.', linewidth=2.0);
plt.plot(xl, p19_est, 'g-.', linewidth=2.0);
plt.plot(xl, p20_est, 'g-.', linewidth=2.0);
plt.plot(xl, p21_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([24,24],[0,0.012],'-k')
plt.plot([58.22,58.22],[0,0.012],'-k')
plt.title('Gaussian Distribution')
plt.title('Gaussian Distribution')

plt.show()

plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=24,vmax=58.22)
plt.title('My Face without Background')

plt.show()
```

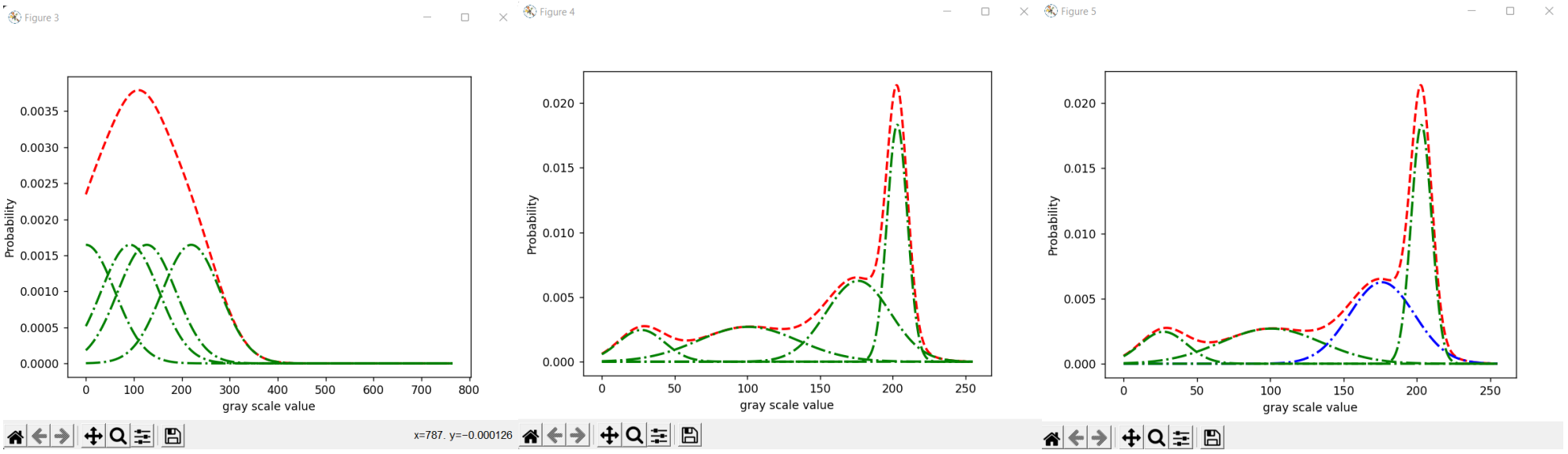
Histogtram


Histogram
Number of pixel
Intensity value
0
500
1000
1500
2000
2500
3000
0
50
100
150
200
250

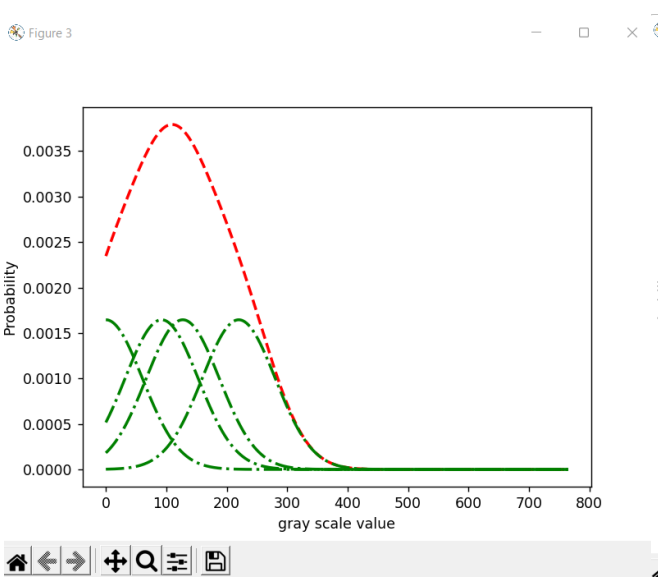
การแสดงผล (ค่า c=4 ; epsilon = 1.0e-3)

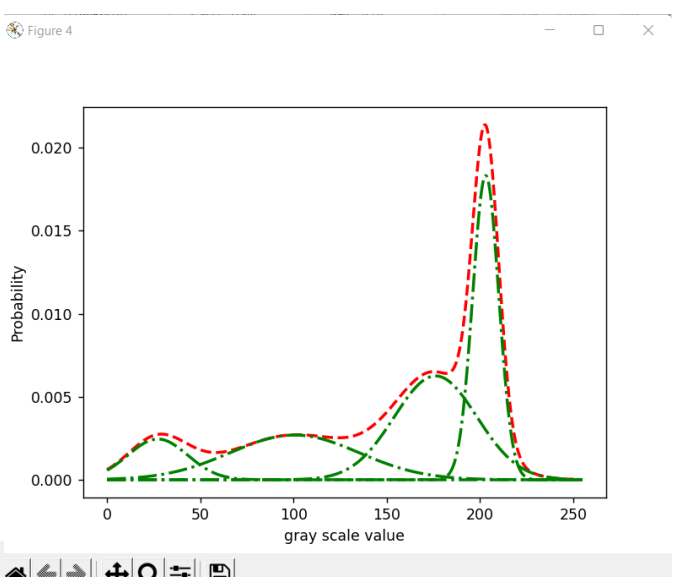


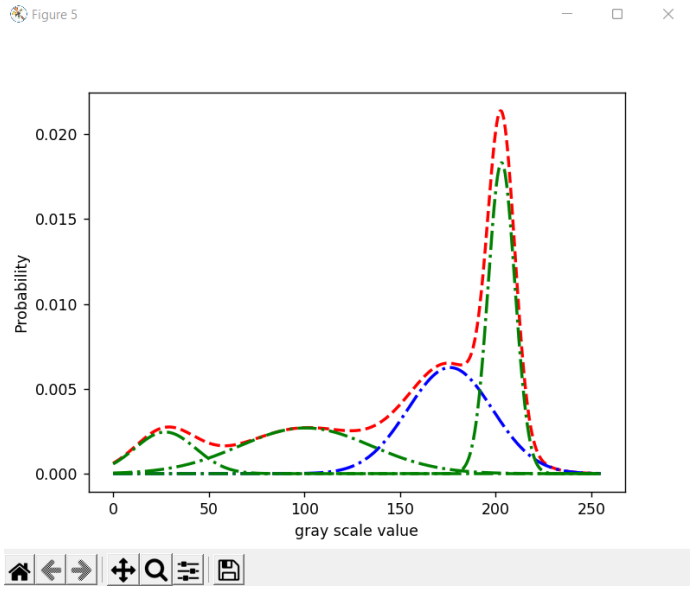
การแสดงผล (ค่า c=4 ; epsilon = 0.5e-3)



การแสดงผล (ค่าc=4 ; epsilon = 1.0e-4)





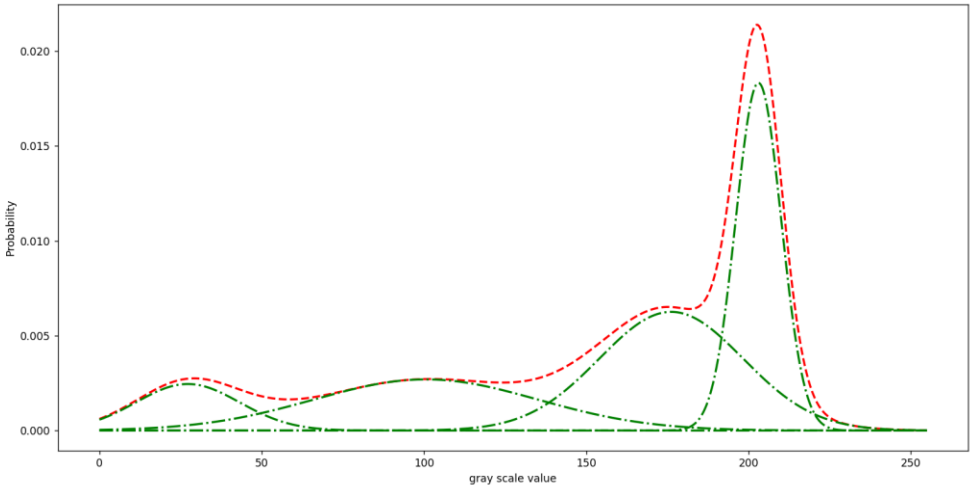


อภิปราย

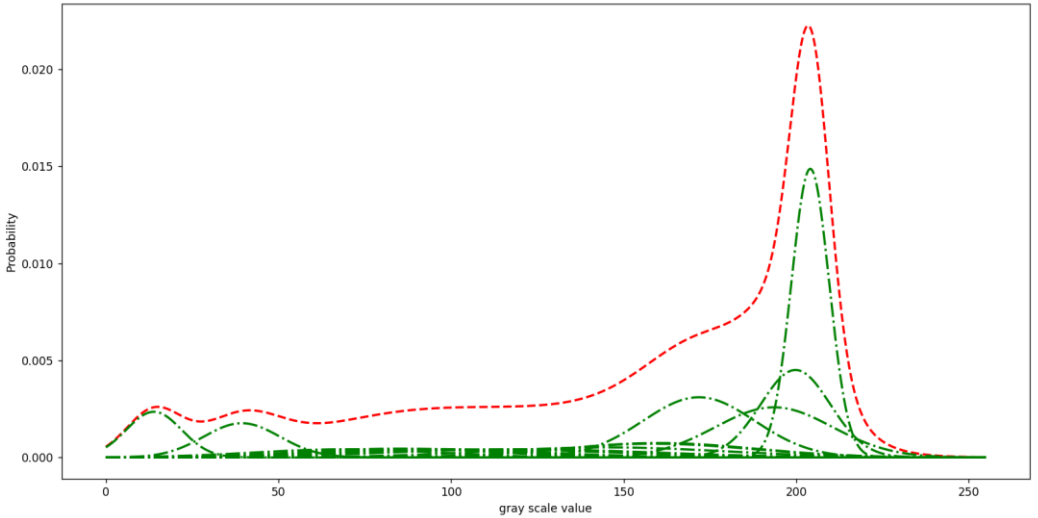
จากการทดลอง เปลี่ยนค่า epsilon ระหว่าง 1.0e-3 และ 1.0e-4 สังเกตจากภาพข้างต้น เราทำการ Fixed ค่า c ไว้ และทำการเลือกค่า epsilonแต่ละค่า ไม่เกิดการเปลี่ยนใดๆ เราจึงทำการเลือกค่าใดค่าหนึ่งเพื่อทำการทดลองขั้นตอนต่อไป

การเลือกค่า c ที่เหมาะสม

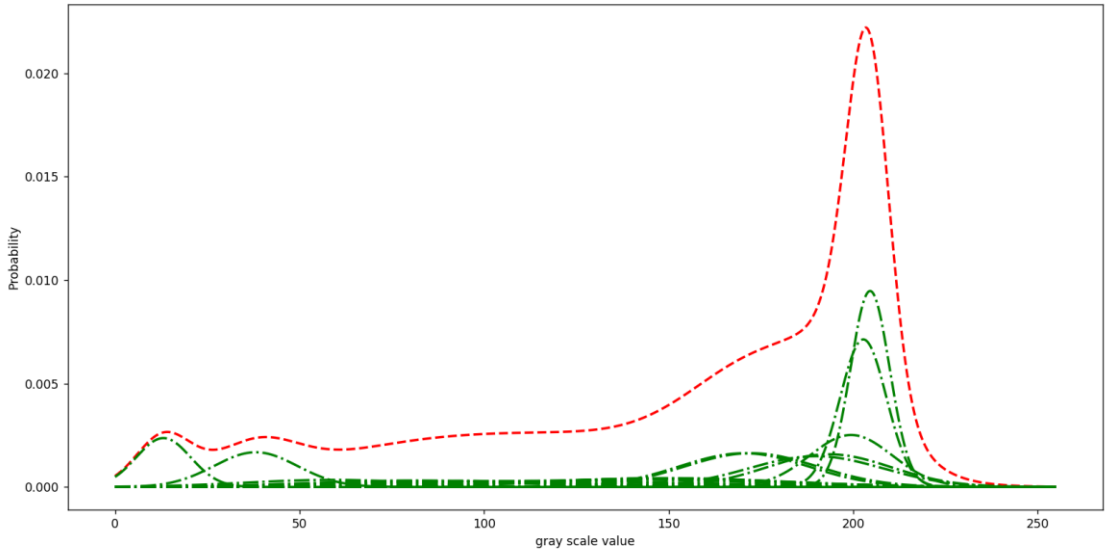
การแสดงผล กรณีที่ค่า c = 4



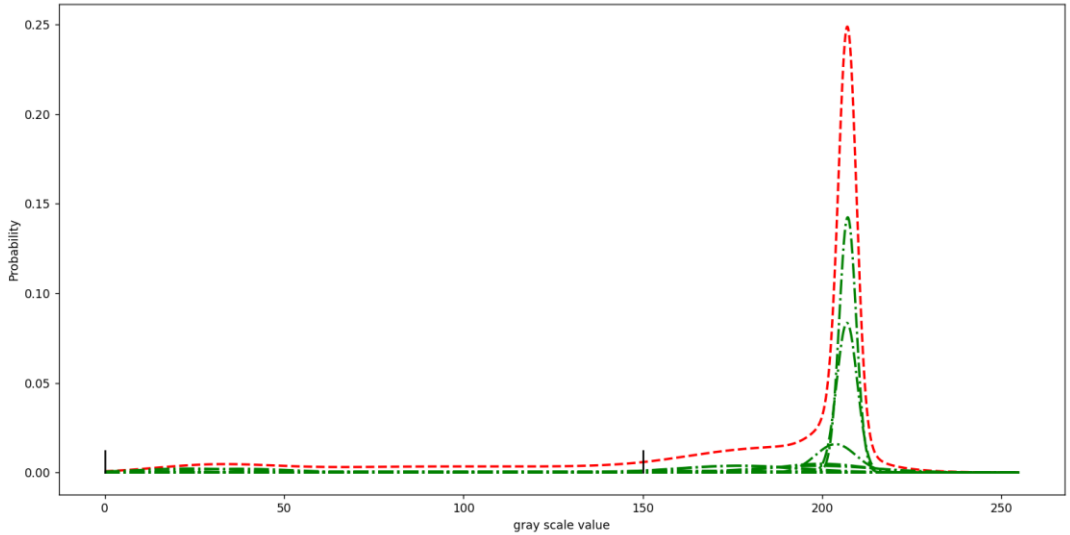
การแสดงผล กรณีที่ค่า c=15



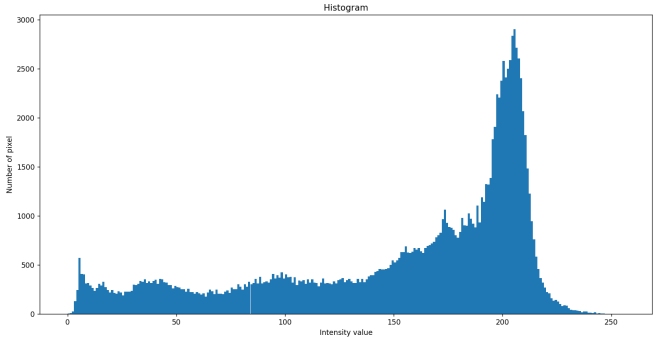
การแสดงผล กรณีที่ค่า c=21



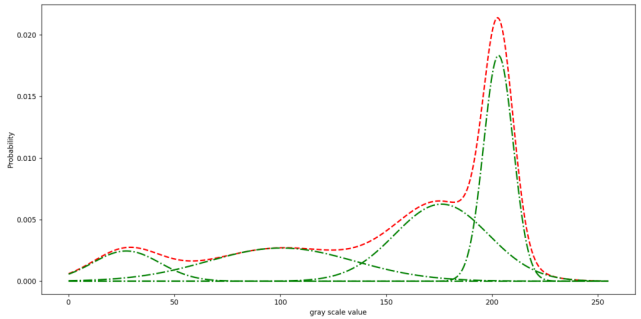
การแสดงผล กรณีที่ค่า c=42



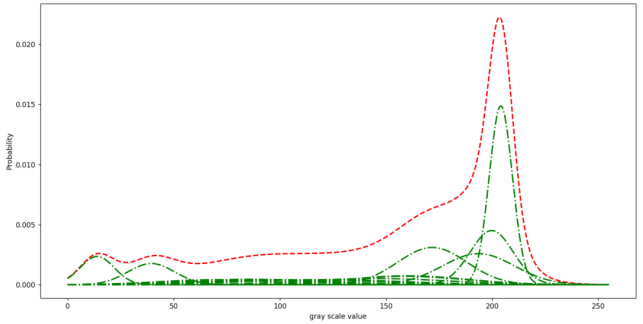
อภิปราย



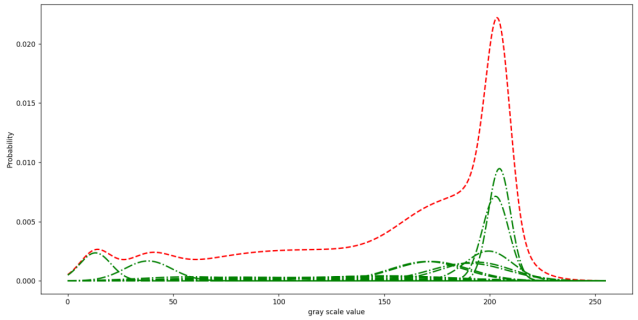
ภาพที่1 Histogram



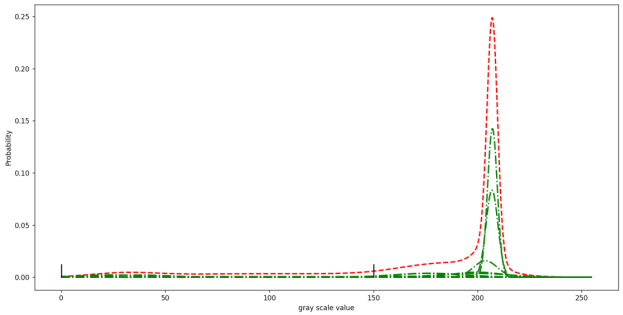
ภาพที่2 GMM (c=4)



ภาพที่3 GMM (c=15)



ภาพที่4 GMM (c=21)

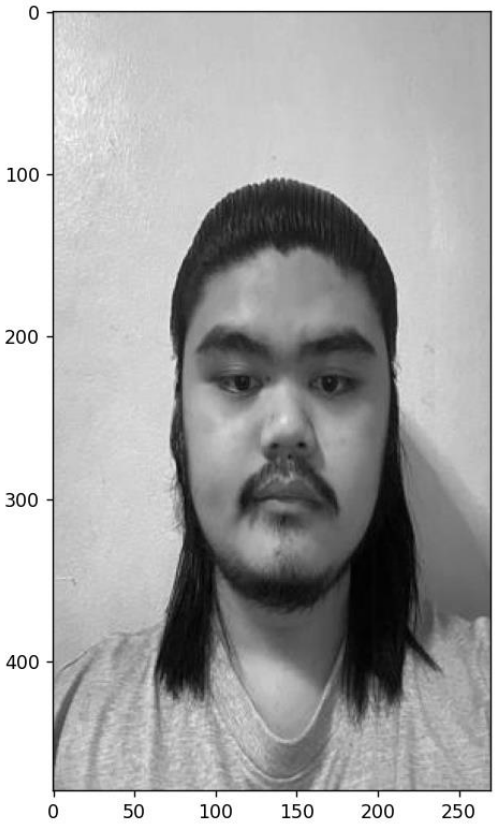


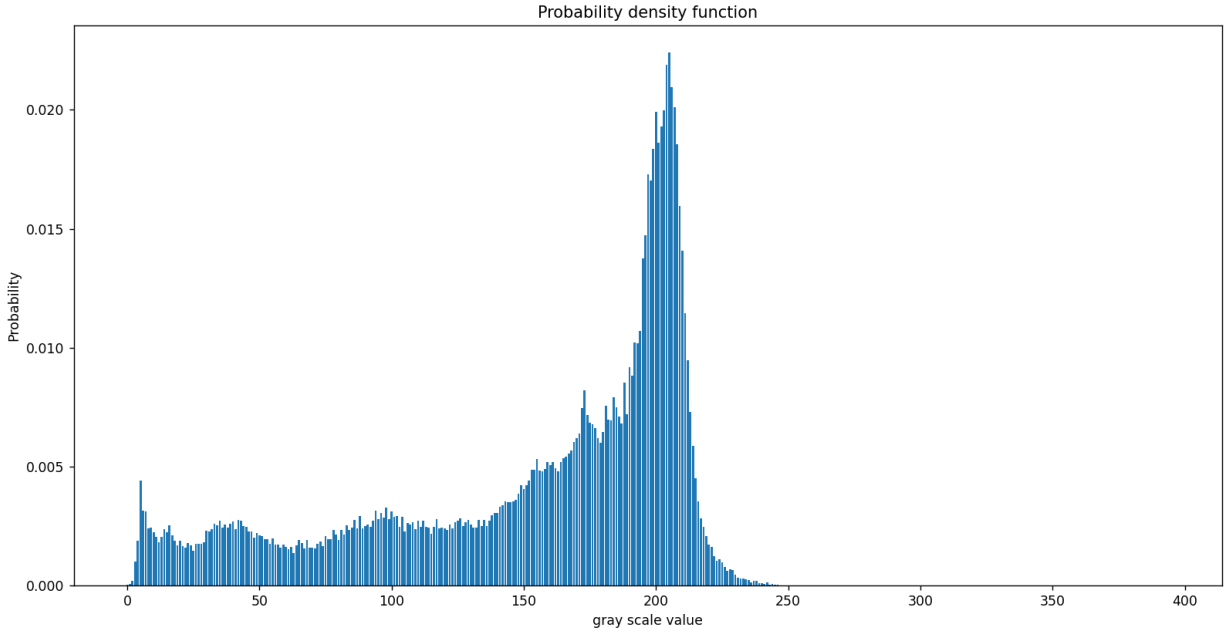
ภาพที่5 GMM (c=42)

จากภาพข้างต้น จะสังเกตว่า ภาพที่3และภาพที่4 มีลักษณะที่คล้ายกับ Histogram ดั้งเดิมมากที่สุด เนื่องจากค่า c เป็นตัวกำหนดการแบ่งกราฟที่แสดงถึง Gaussian Distribution แต่ละค่าจำนวน c ค่า ซึ่งผลรวมที่ได้ก็คือกราฟที่แสดงด้วยสีแดง ซึ่งจะต้องมีลักษณะคล้ายคลึงกับ Histogram ของรูปภาพมากที่สุด ดังนั้น จากภาพข้างต้น จะพบว่า ค่า c ที่เหมาะสมที่สุดคือ 15 เนื่องจากมีลักษณะใกล้เคียงกับHistogramมากที่สุด และแม้ว่า ที่ c=21 นั้นจะให้กราฟที่เหมือนกันกับHistogram แต่ก็ไม่ได้แตกต่างกับ c ที่ 15 มากนัก เพื่อเป็นการประหยัดเวลาในการประมวลผล และเวลาการในการเขียนโค้ด ค่าที่เหมาะสมที่สุดคือ c=15 นั่นเอง

การทดลองที่1.2 ทำการคัดเฉพาะส่วนที่เป็นใบหน้าของภาพ MyFacePic.jpg โดยใช้คำสั่ง plt.imshow(arr, cmap='gray', vmin = ค่าความเข้มต่ำสุด, vmax = ค่าความเข้มสูงสุด) โดยที่พิจารณาค่า vmin ที่เป็นความเข้มต่ำสุดที่ต้องการให้แสดง และค่า vmax ที่เป็นความเข้มสูงสุดที่ต้องการให้แสดง จากขอบเขตของการแจกแจงแบบ Gaussian ที่เหมาะสมสำหรับการคัดแยก

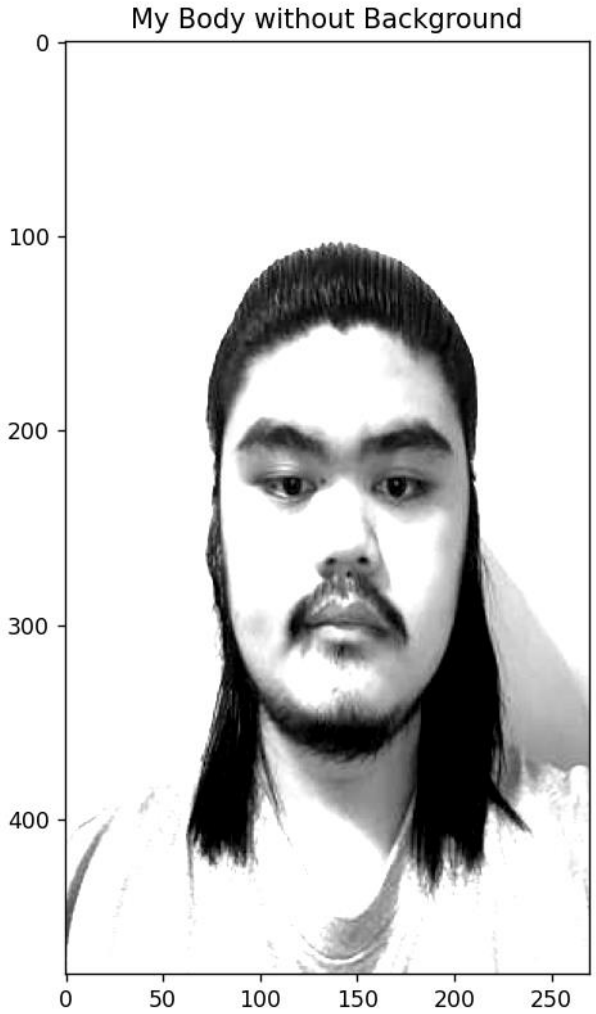
รายละเอียดของภาพที่ใช้ในการทดลอง (MyFacePic_Resize)



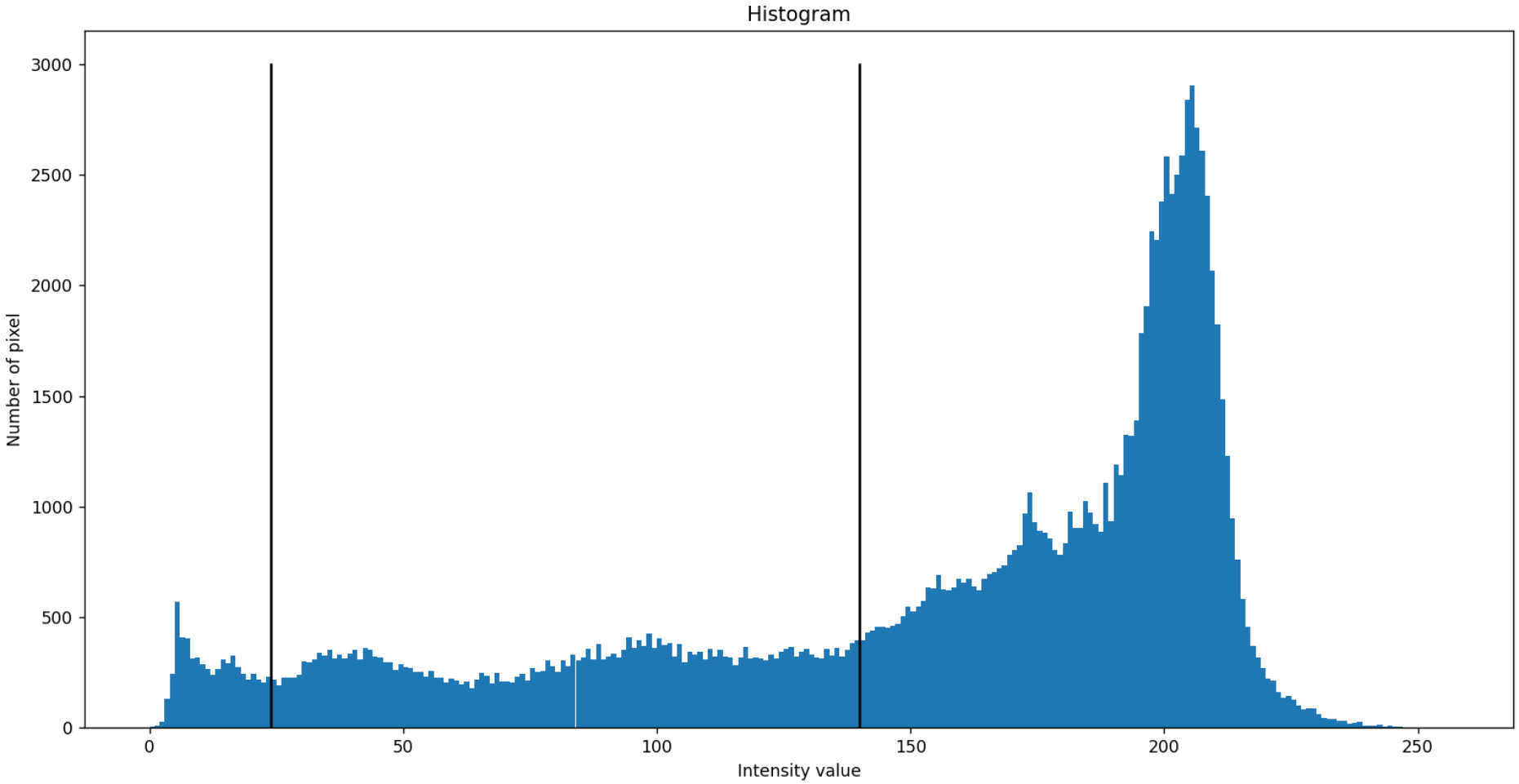


a. ใบหน้าและส่วนที่เป็นลำตัวพร้อมกัน

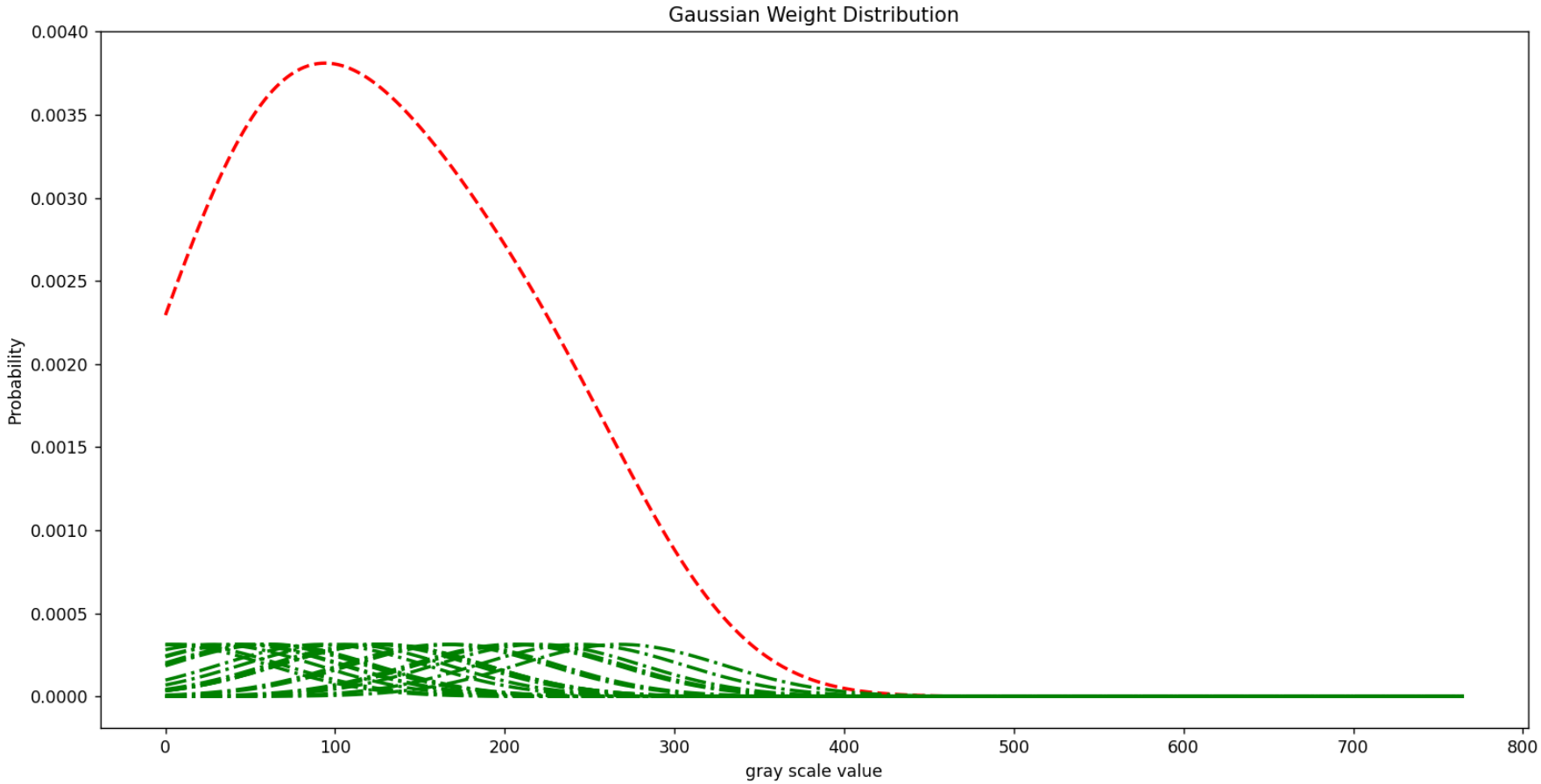
การแสดงผล(c=15 ; Vmin=24 ; Vmax=140)



Histogram (c=15 ; Vmin=24 ; Vmax=140)



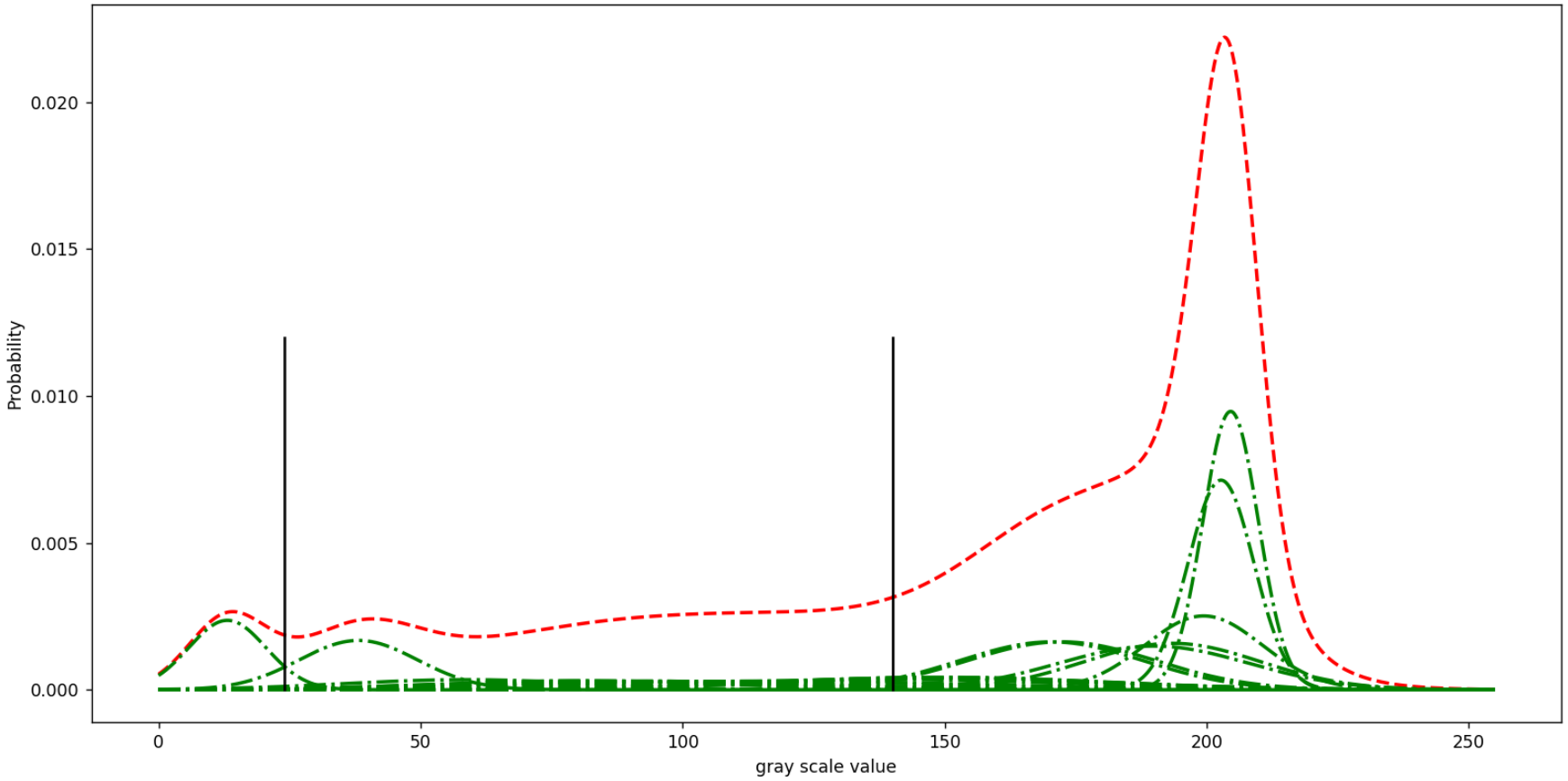
Gaussian Weight Distribution (c=15 ; Vmin=24 ; Vmax=140)



.หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

Gaussian Mixture Model(c=15 ; Vmin=24 ; Vmax=140)

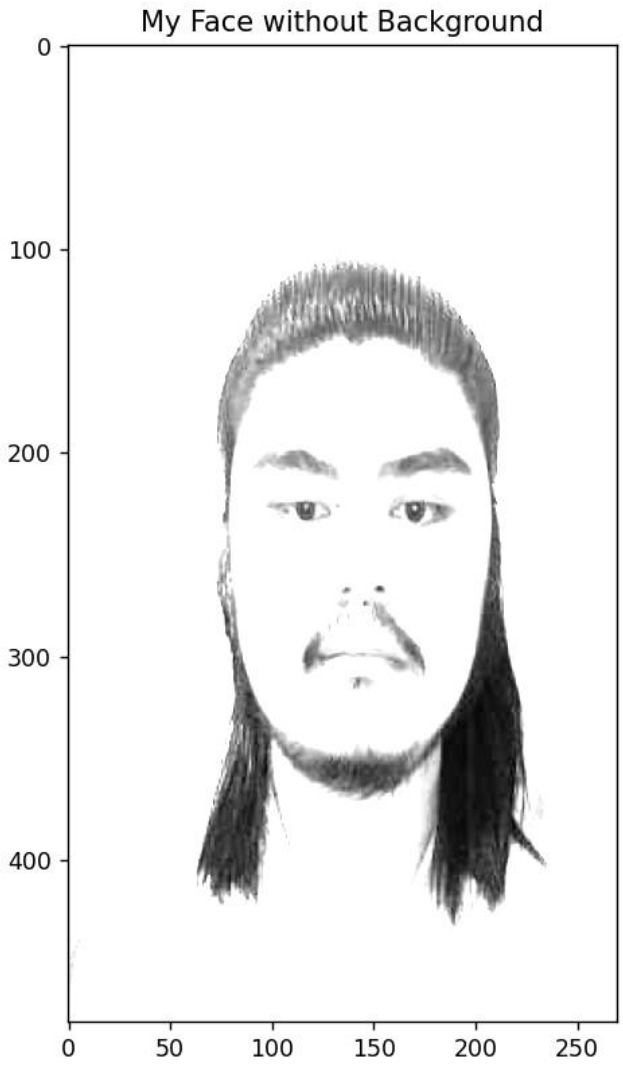


หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

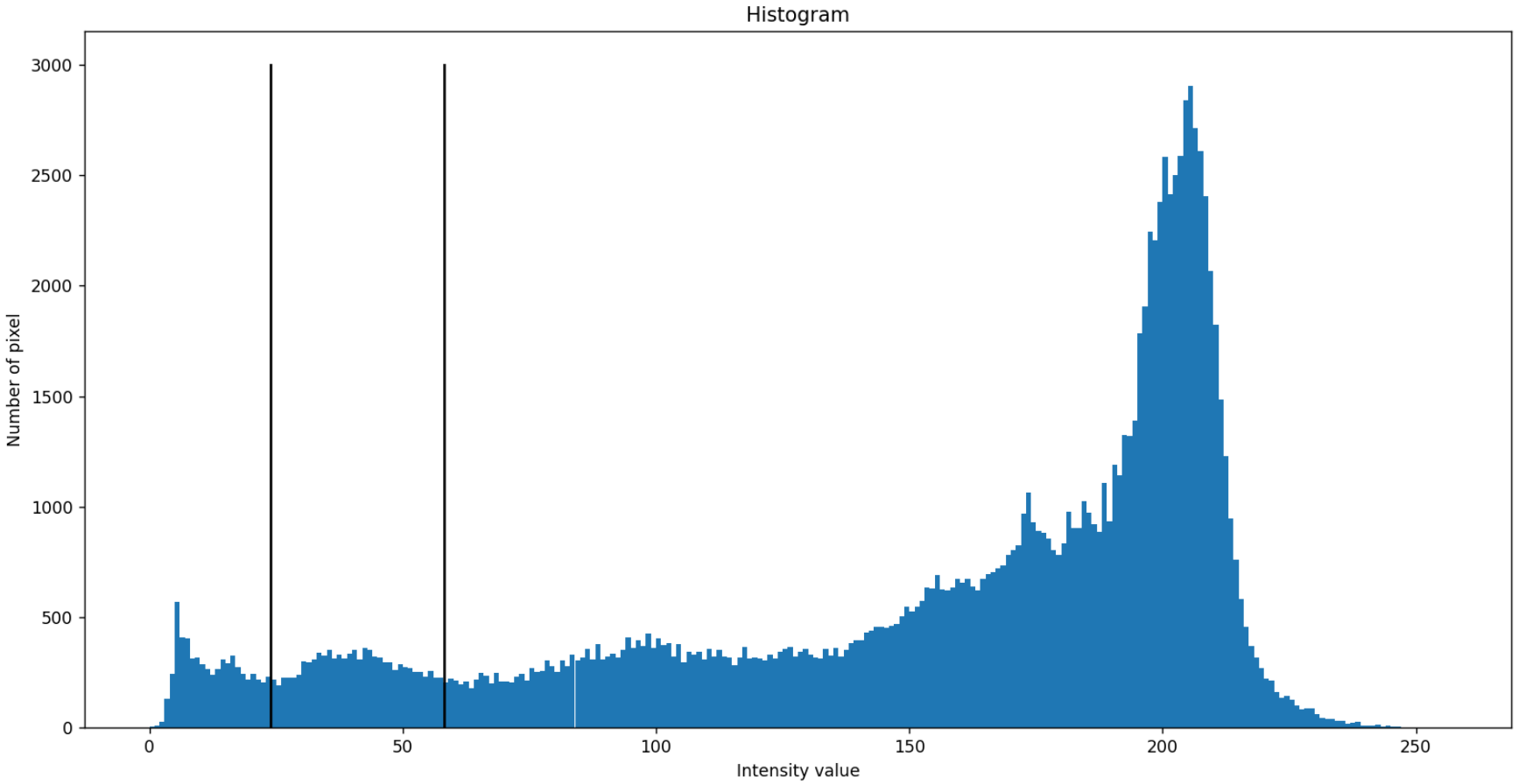
กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

b. เฉพาะส่วนที่เป็นใบหน้า

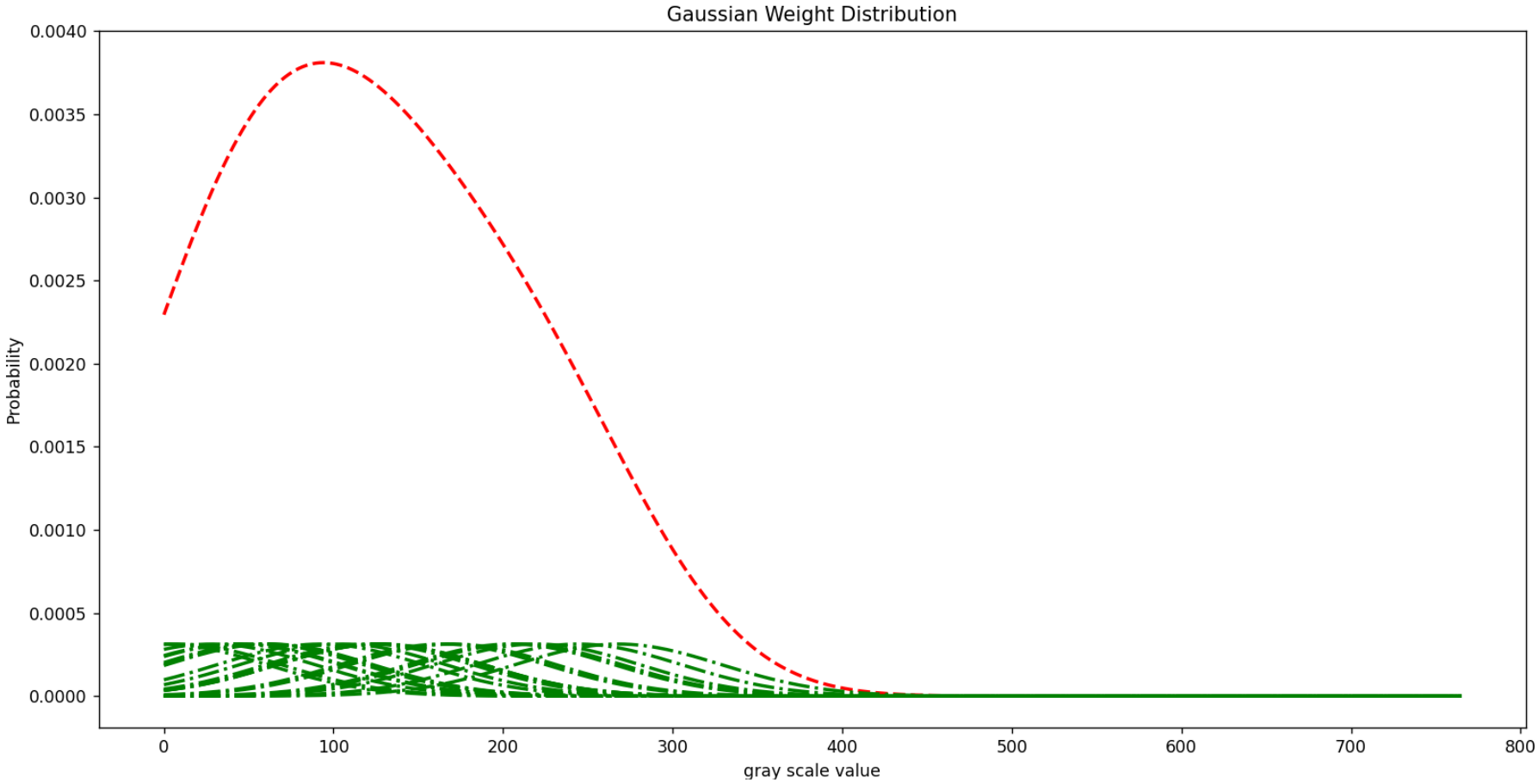
การแสดงผล (Vmin=0 ; Vmax=70)



Histogram (c=15 ; Vmin=24 ; Vmax=58.22)



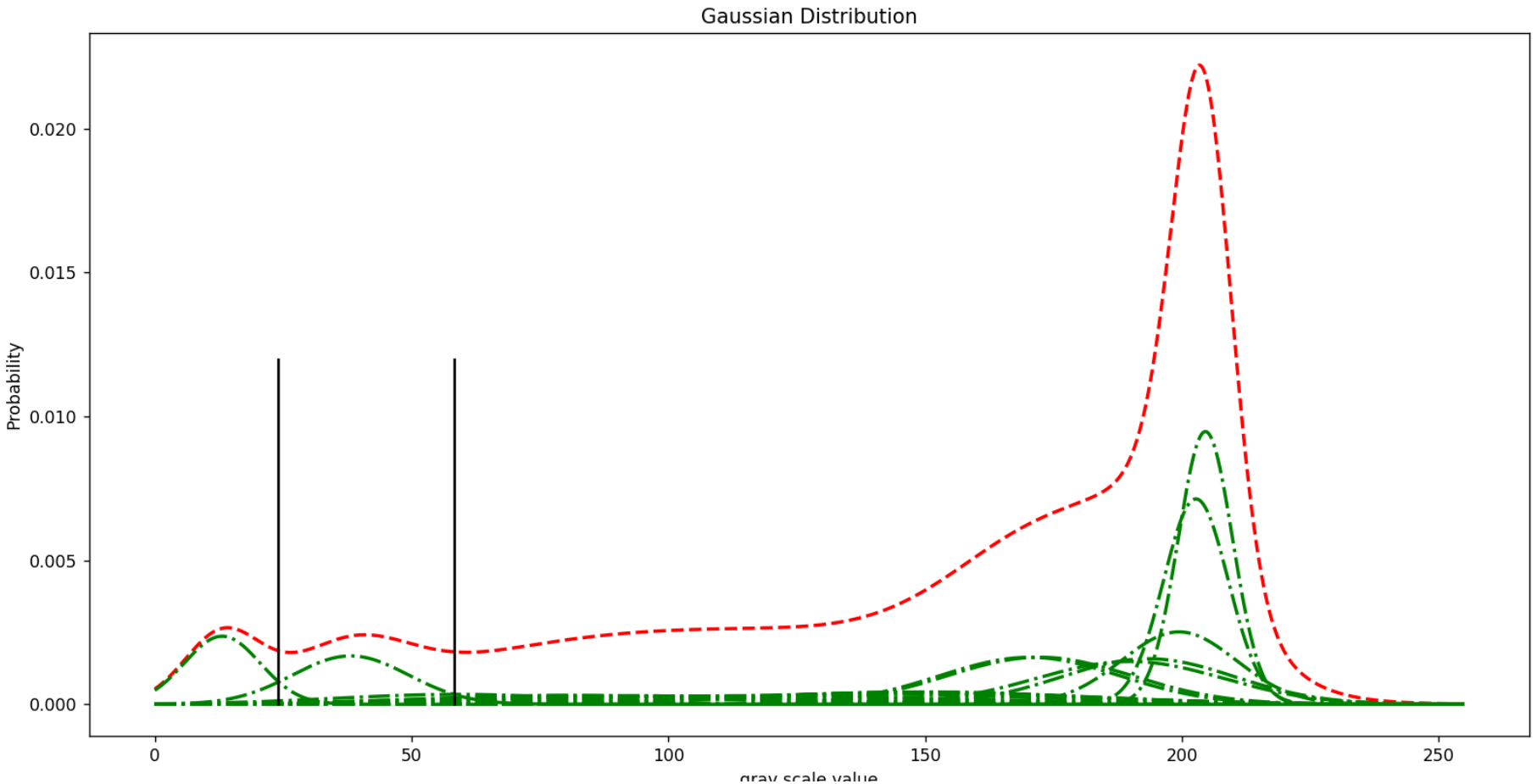
Gaussian Weight Distribution (c=15 ; Vmin=24 ; Vmax=58.22)



หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

Gaussian Mixture Model (c=15 ; Vmin=24 ; Vmax=58.22)



อภิปราย

การเลือกค่า Vmin และ Vmax สำหรับทำ Otsu's thresholding นั้น จะทำการพิจารณาจากจุดตัดของกราฟแต่ละเส้นที่ตัดกัน ของ Gaussian Mixture Model (GMM) แล้วทำการป้อนค่า Vmin-max ด้วยคำสั่ง plt.imshow(...) เมื่อมีจุดตัดหลากหลาย ให้ทำการเลือกค่าแล้วทำการ Simulation ให้ได้ค่าที่เหมาะสม และได้ภาพที่เหมาะสมสำหรับการแสดงผลต่อไป

การทดลองที่2 ทำการทดลองซ้ำ โดยเปลี่ยนไปใช้ไฟล์รูป ClassificationGS.jpg แล้วทำการคัดเลือกส่วนที่เป็น นก จากรูป ClassificationGS.jpg มานำเสนอ พร้อมทั้งมาร์คตำแหน่งดังกล่าวลงใน histogram ทั้งสองนำเสนอผลการทดลองพร้อมอภิปราย

คำสั่งที่ใช้งาน

```
from PIL import Image
import numpy as np
import cv2
img = Image.open('D:\Telecom_Lab\Lab5\Figure\ClassificationGS.jpg') # image extension
*.png,*.jpg
new_width  = 270
new_height = 480
img = img.resize((new_width, new_height))
img.save('D:\Telecom_Lab\Lab5\Figure\ClassificationGS_Resize.jpg') # format may what u
want ,*.png,*jpg,*.gif
from skimage.io import imread
from skimage.color import rgb2gray
mountain_r = cv2.imread('D:\Telecom_Lab\Lab5\Figure\ClassificationGS_Resize.jpg')
#Plot
import matplotlib.pyplot as plt
plt.figure(0)
plt.imshow(mountain_r,cmap="gray")
plt.show()
img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\ClassificationGS_Resize.jpg')
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
meandata = mean(data)
stddata = np.std(data)
x= meandata+(4*stddata)
```

```
c = np.arange(0, x, 1)

k = len(c)
i = len(data)
plt.figure(1)
hist,bin = np.histogram(data,c)
y = len(hist)
w = np.arange(0, x-1, 1)
r = hist/i
plt.bar(w,r)
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.show()

plt.figure(2)
plt.hist(img.ravel(),256,[0,256])
plt.ylabel('Number of pixel')
plt.xlabel('Intensity value')
plt.plot([15,15],[0,3000],'-k')
plt.plot([66.06,66.06],[0,3000],'-k')
plt.title('Histogram')
plt.show()

img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\ClassificationGS_Resize.jpg',0)
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]

m = len(data)
epsilon = 1.0e-3
difference = epsilon
counter = 0
```

```
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
c = 5

from numpy.random import seed
from numpy.random import rand

seed(1)

mu_est = 2*mean(data)*np.sort(rand(c,1))

sigma_est = np.ones(c)*np.std(data)

p_est = np.ones(c)/c

def gaussian_norm_density(x, mu, sig):
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))/(sig * np.sqrt(2 *
np.pi))
d = max(data)

xl = np.arange(0, d*3,0.1)


p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);


plt.figure(3)
plt.plot(xl,p1_est+p2_est+p3_est+p4_est+p5_est, 'r--',linewidth=2.0)
```

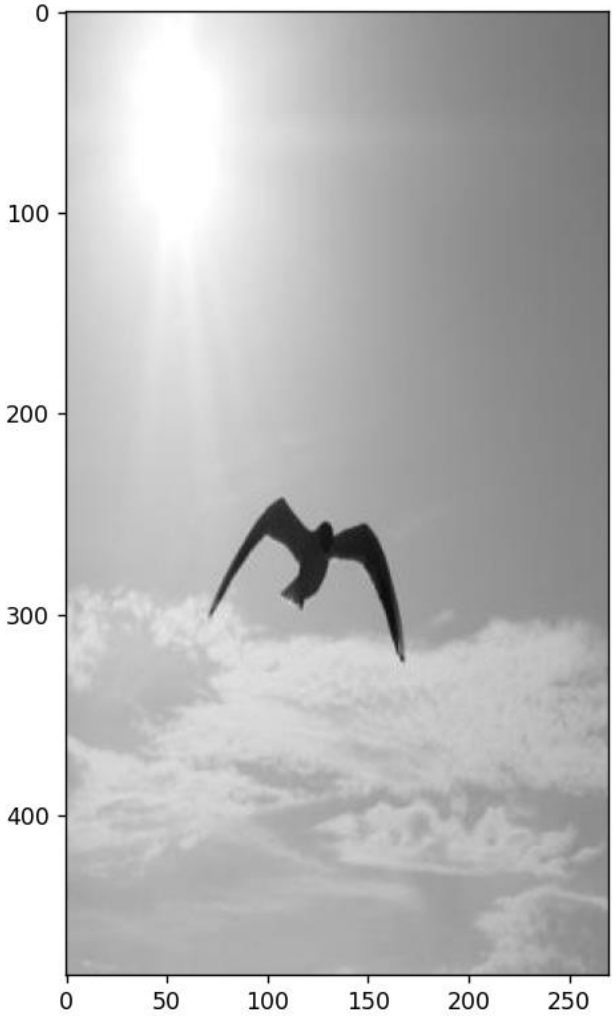
```
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);



plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Gaussian Weight Distribution')

plt.show()

clas = []
ok = []
while np.any(difference >= epsilon) and (counter < 25000):
    for j in range(0, c):

        clas.insert(j, p_est[j] * gaussian_norm_density(data, mu_est[j], sigma_est[j]))

    ok = clas[0] + clas[1] + clas[2] +clas[3]+clas[4]

    for j in range(0, c):
        clas[j] = clas[j] / ok


    mu_est_old = mu_est
    sigma_est_old = sigma_est
    p_est_old = p_est
    mu_est = []
    sigma_est = []
```

```
    p_est = []

    for j in range(0, c):
        mu_est.insert(j, sum((clas[j]) * data) / sum(clas[j]))
        sigma_est.insert(j, np.sqrt(sum((clas[j]) * np.power((data - mu_est[j]), 2)) /
sum(clas[j])))
        p_est.insert(j, mean(clas[j]))

    difference =
sum(abs(np.subtract(mu_est_old,mu_est)))+sum(abs(np.subtract(sigma_est_old,sigma_est)))\
                +sum(abs(np.subtract(p_est_old,p_est)))

    print(difference)

    counter = counter + 800
    print('counter =',counter)
xl = np.arange(0, d, 0.1)
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);
plt.figure(4)
plt.plot(xl, p1_est + p2_est + p3_est + p4_est + p5_est, 'r--', linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([15,15],[0,0.012],'-k')
```

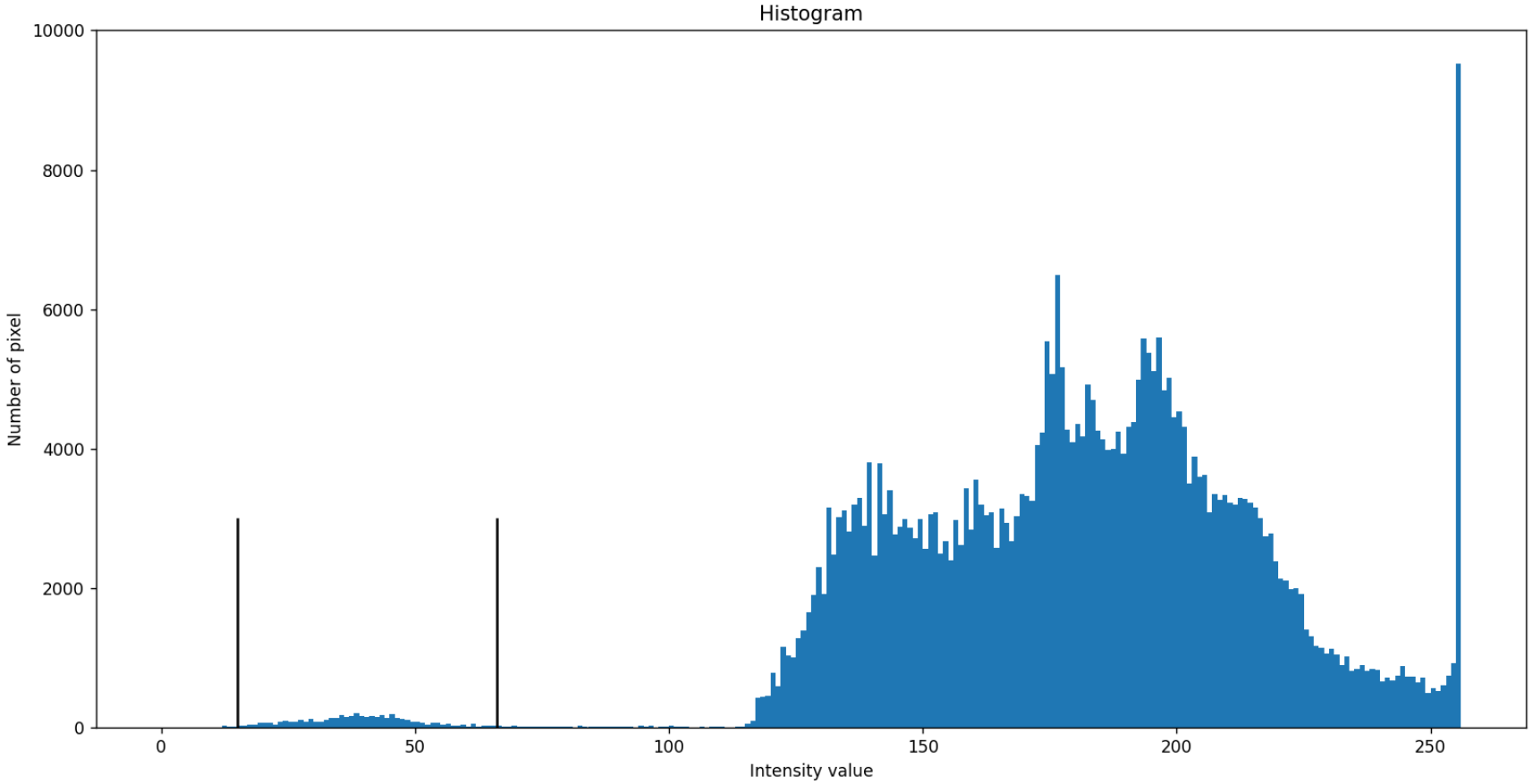
```
plt.plot([66.06,66.06],[0,0.012],'-k')

plt.show()

p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
p5_est = p_est[4] * gaussian_norm_density(xl, mu_est[4], sigma_est[4]);

#sum_est = p1_est + p2_est + p3_est + p4_est+p5_est

plt.figure(5)
plt.plot(xl, p1_est + p2_est + p3_est +p4_est + p5_est  , 'r--', linewidth=2.0)

plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.plot(xl, p5_est, 'g-.', linewidth=2.0);

plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([15,15],[0,0.012],'-k')
plt.plot([66.06,66.06],[0,0.012],'-k')

plt.title('Gaussian Distribution')
plt.show()
plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=15,vmax=66.06)
plt.title('Only Bird')
plt.show()
```

การแสดงผล **(c=5 ; Vmin=15 ; Vmax=66.06)**

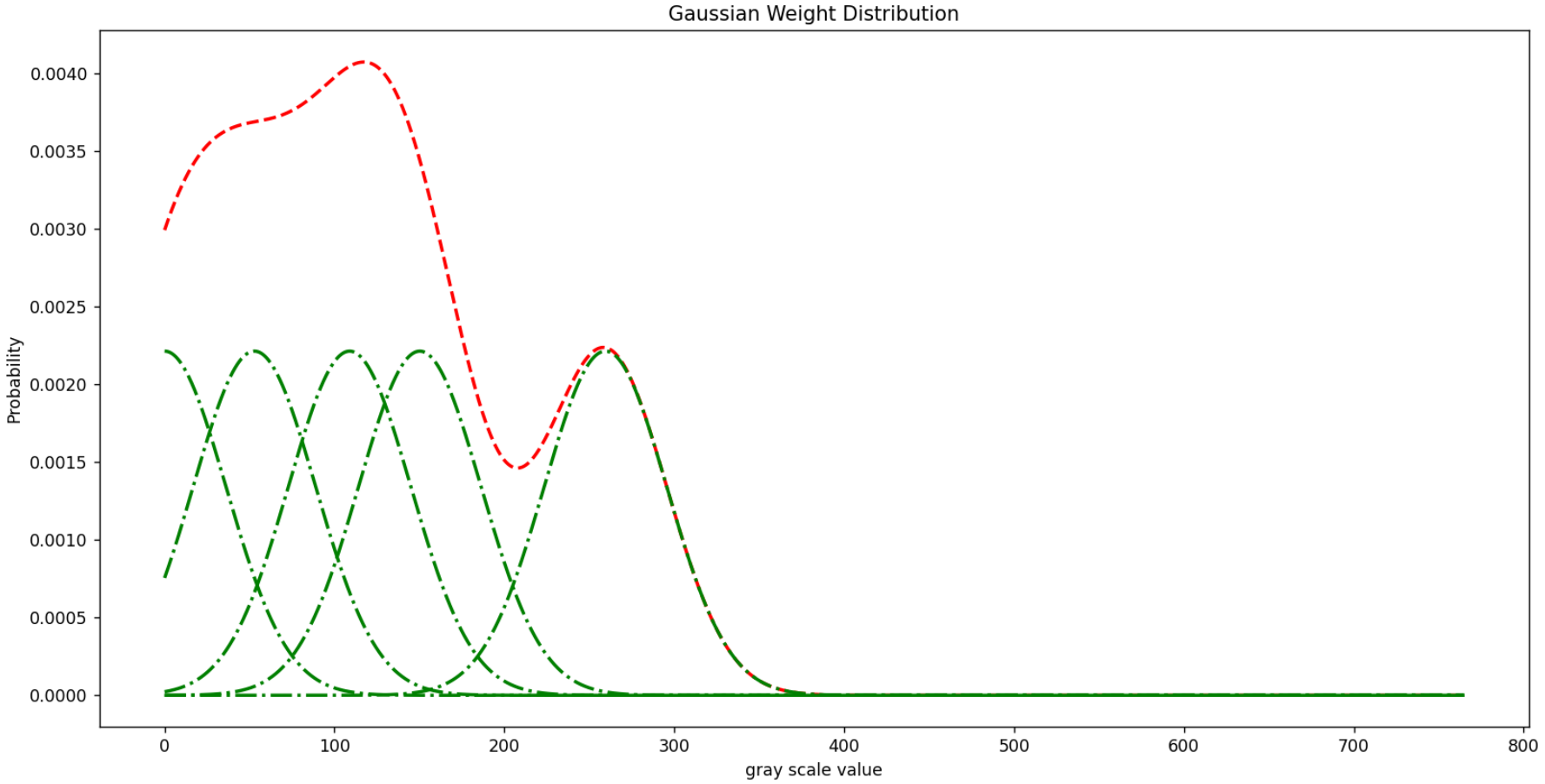
1. ClassificationGS_Resize (270x480)



2. Histogram (c=5 ; Vmin=15 ; Vmax=66.06)



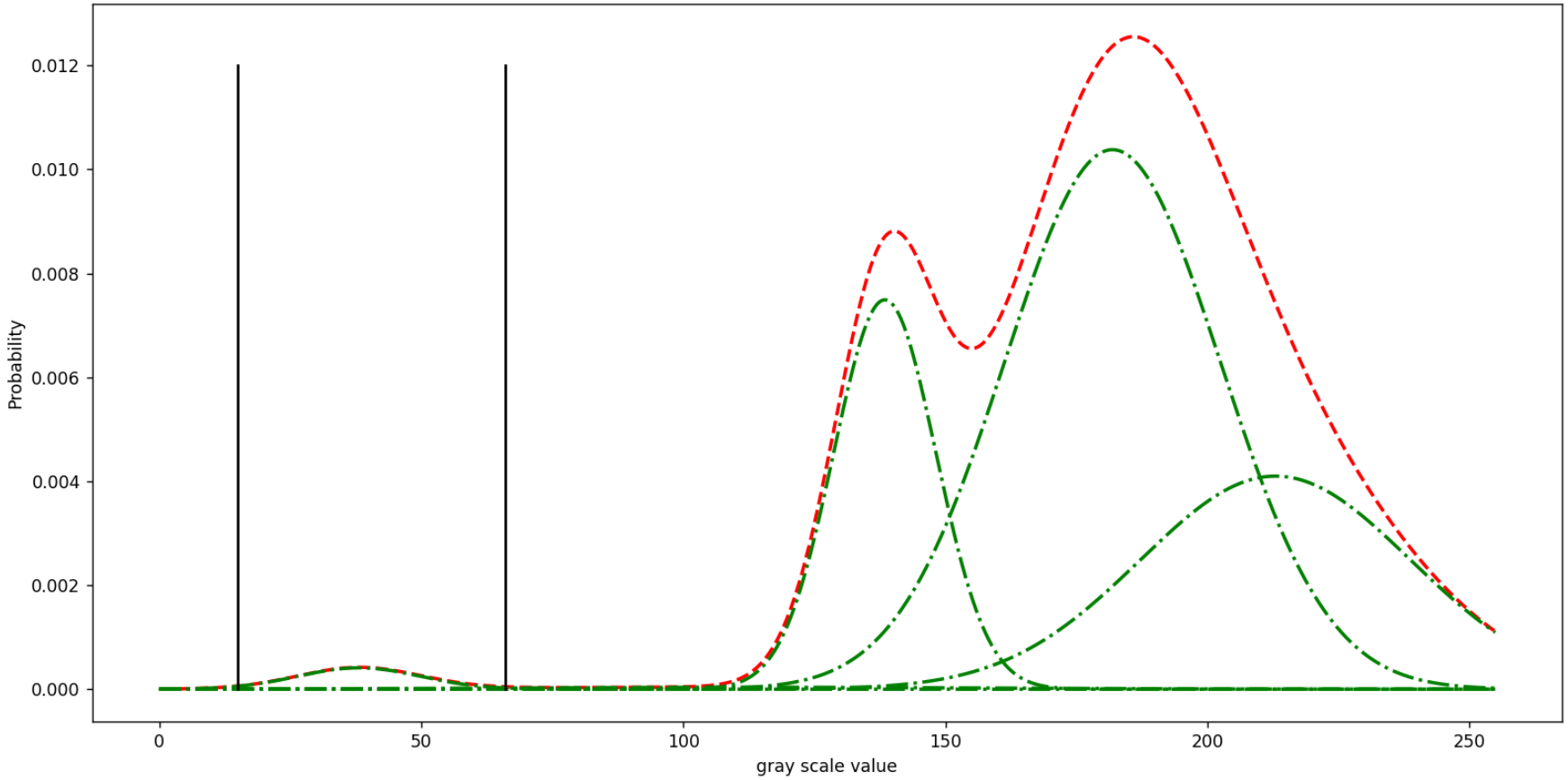
3.Gaussian Weight Distribution (c=5 ; Vmin=15 ; Vmax=66.06)



หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

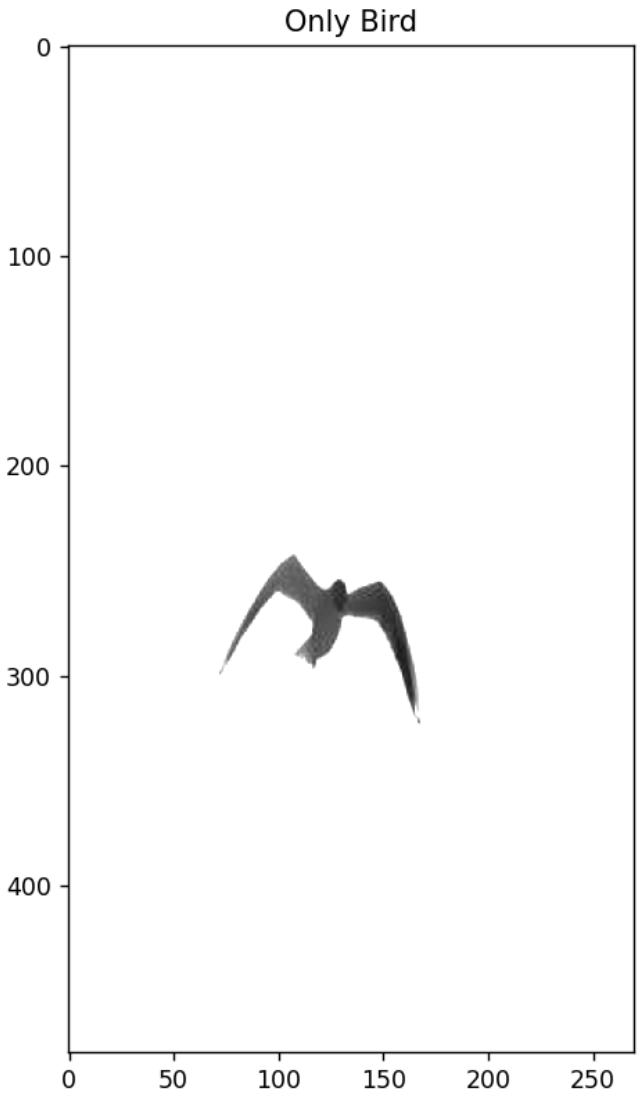
4.Gaussian Mixture Model (c=5 ; Vmin=15 ; Vmax=66.06)



หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

4. คัดเลือกส่วนเฉพาะที่เป็นนก



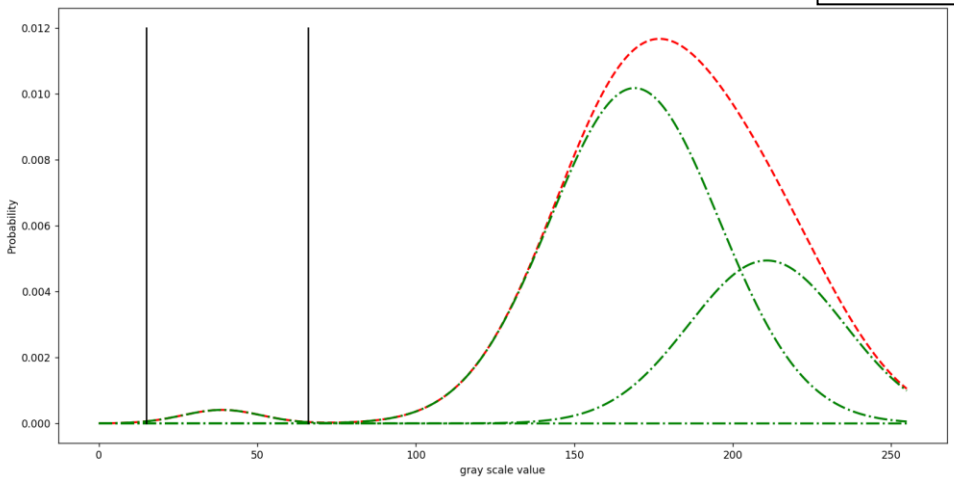
อภิปราย

จากการทดลองเราต้องทำการหาค่า Gaussian Mixture Model ของตัวนกก่อน เพื่อนทำการกำหนดค่า Vmin และ Vmax ของภาพ จากจุดตัดของกราฟ เนื่องจากเราต้องการเพียงแค่นกอย่างเดียว หลังจากที่กำหนดค่าVminและVmaxจากจุดตัดของกราฟเรียบร้อยแล้ว ก็ทำการกำหนดค่า Epsilon เช่นเดิม แต่เนื่องจากไม่ว่าค่า Epsilonจะมีเท่าใดก็ไม่ส่งผลต่อกราฟ จึงทำการเลือกมา1ค่า โดยค่า c ที่ทำการเลือกคือ 3,4 และ 5

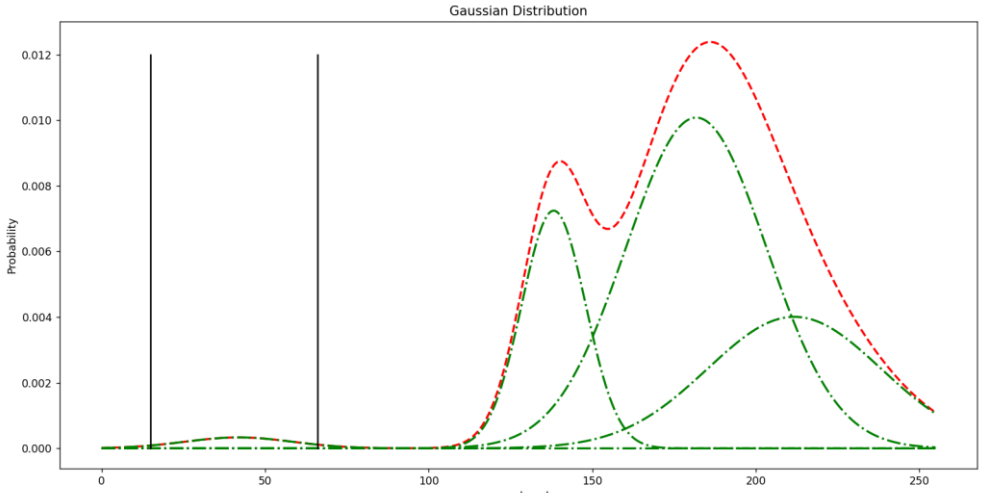
หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian

ค่าc=3



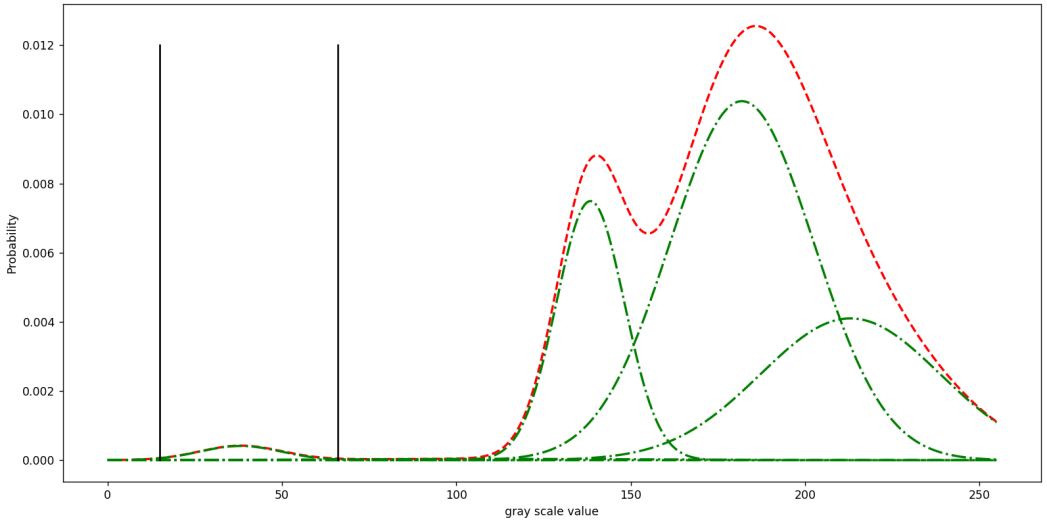
ค่าc=4



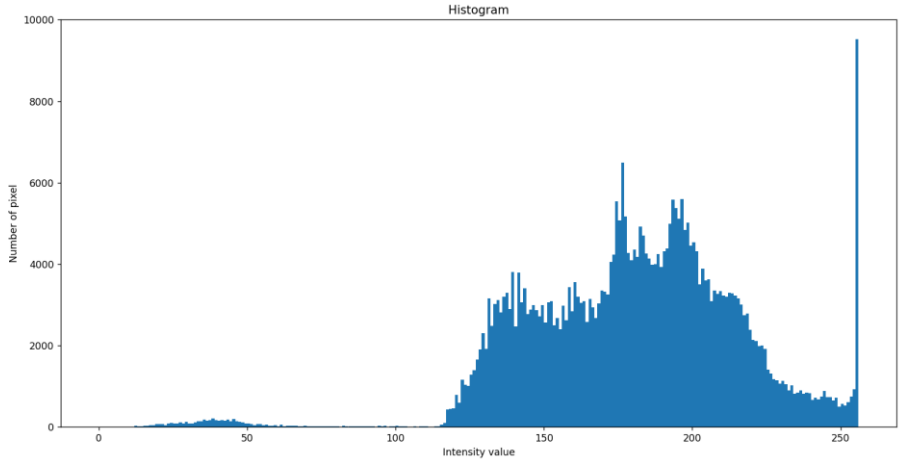
หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

ค่าc=5



จากภาพทั้ง3ที่แทนลักษณะของ Gaussian Mixture Model ของค่า c ทั้ง3ค่า ค่าcที่ให้ลักษณะคล้ายกับHistogramมากที่สุดคือ c=4 และ 5 นั่นเอง แต่ทำไมเราถึงเลือกc=5 นั่นเพราะ การหาค่า Vmin และ Vmax ทำได้ง่ายกว่า c=4 เมื่อทำการ Zoom กราฟเพื่อพิจารณาค่า V ที่ c=5 ทำได้ดีกว่า c=4 นั่นเอง สามารถดูภาพ Histogram ด้านล่างประกอบเพื่อเปรียบเทียบได้ และค่า Epsilon ไม่มีผลต่อลักษณะของรูปกราฟดังกล่าว



การทดลองที่3 ทำการทดลองซ้ำ โดยเปลี่ยนไปใช้ไฟล์รูป CTScan.jpg แล้วทำการคัดเลือกส่วนที่เป็น

a. ปอด

คำสั่งที่ใช้งาน

```
from PIL import Image
import numpy as np
import cv2
img = Image.open('D:\Telecom_Lab\Lab5\Figure\CTScan.jpg') # image extension *.png,*.jpg
new_width  = 270
new_height = 480
img = img.resize((new_width, new_height))
img.save('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg') # format may what u want
,*.png,*jpg,*.gif
from skimage.io import imread
from skimage.color import rgb2gray
mountain_r = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg')
#Plot
import matplotlib.pyplot as plt
plt.figure(0)
plt.imshow(mountain_r,cmap="gray")
plt.show()
img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg')
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
meandata = mean(data)
stddata = np.std(data)
x= meandata+(4*stddata)
```

```
c = np.arange(0, x, 1)

k = len(c)
i = len(data)
plt.figure(1)
hist,bin = np.histogram(data,c)
y = len(hist)
w = np.arange(0, x-1, 1)
r = hist/i
plt.bar(w,r)
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Probability density function')
plt.show()

plt.figure(2)
plt.hist(img.ravel(),256,[0,256])
plt.ylabel('Number of pixel')
plt.xlabel('Intensity value')
plt.plot([41.20,41.20],[0,12000],'-k')
plt.plot([83.118,83.118],[0,12000],'-k')
plt.title('Histogram')
plt.show()

img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg',0)
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]

m = len(data)
epsilon = 1.0e-3
difference = epsilon
counter = 0
```

```
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
c = 4

from numpy.random import seed
from numpy.random import rand

seed(1)

mu_est = 2*mean(data)*np.sort(rand(c,1))

sigma_est = np.ones(c)*np.std(data)

p_est = np.ones(c)/c

def gaussian_norm_density(x, mu, sig):
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))/(sig * np.sqrt(2 *
np.pi))
d = max(data)

xl = np.arange(0, d*3,0.1)


p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);


plt.figure(3)
plt.plot(xl,p1_est+p2_est+p3_est+p4_est, 'r--',linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
```

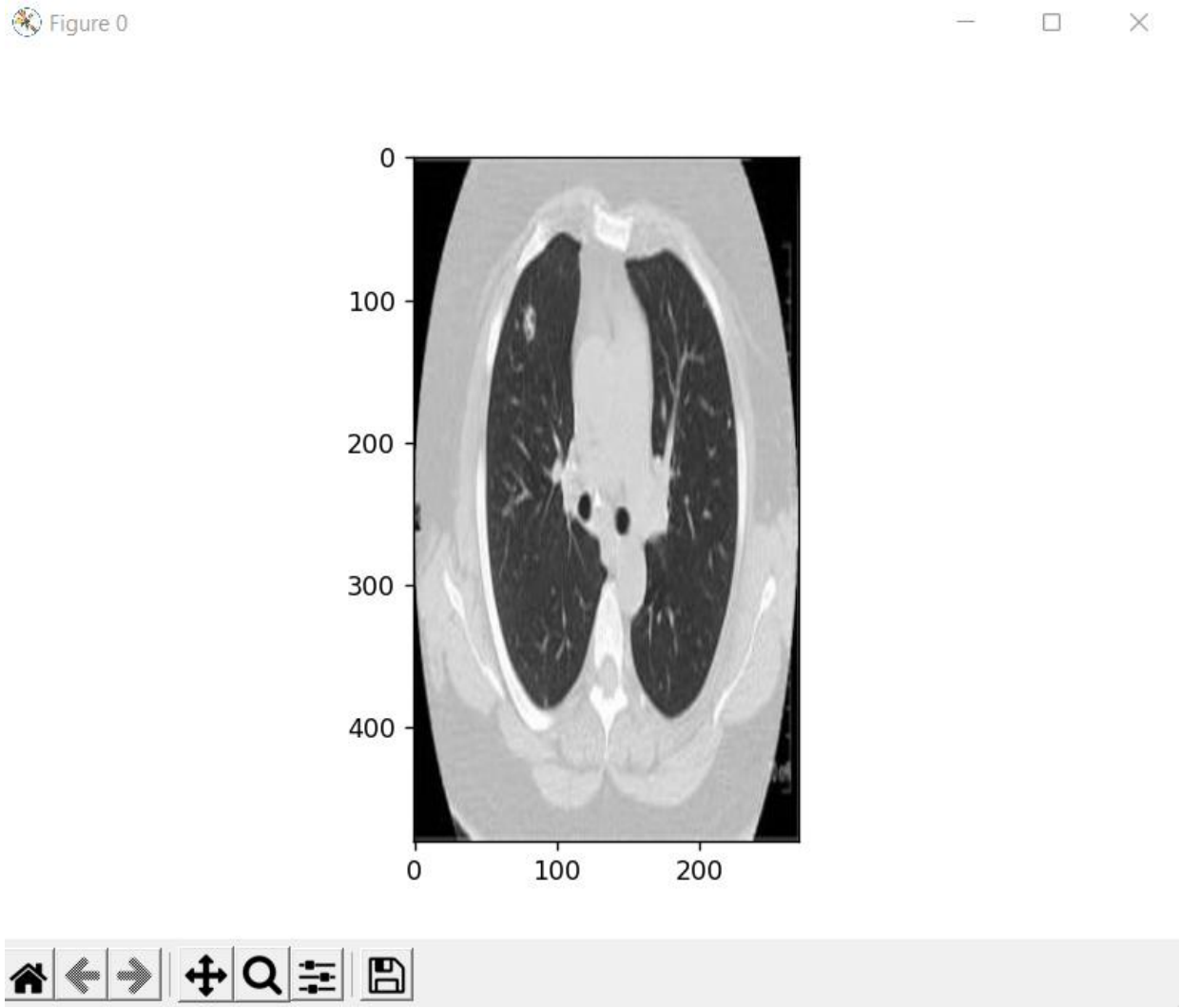
```
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);

plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Gaussian Weight Distribution')
plt.show()

clas = []
ok = []
while np.any(difference >= epsilon) and (counter < 25000):
    for j in range(0, c):

        clas.insert(j, p_est[j] * gaussian_norm_density(data, mu_est[j], sigma_est[j]))

    ok = clas[0] + clas[1] + clas[2] + clas[3]

    for j in range(0, c):
        clas[j] = clas[j] / ok


    mu_est_old = mu_est
    sigma_est_old = sigma_est
    p_est_old = p_est
    mu_est = []
    sigma_est = []
    p_est = []

    for j in range(0, c):
        mu_est.insert(j, sum((clas[j]) * data) / sum(clas[j]))
```

```
        sigma_est.insert(j, np.sqrt(sum((clas[j]) * np.power((data - mu_est[j]), 2)) /
sum(clas[j])))
        p_est.insert(j, mean(clas[j]))

    difference =
sum(abs(np.subtract(mu_est_old,mu_est)))+sum(abs(np.subtract(sigma_est_old,sigma_est)))\
                +sum(abs(np.subtract(p_est_old,p_est)))

    print(difference)

    counter = counter + 800
    print('counter =',counter)
xl = np.arange(0, d, 0.1)
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
plt.figure(4)
plt.plot(xl, p1_est + p2_est + p3_est+p4_est , 'r--', linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([41.20,41.20],[0,0.012],'-k')
plt.plot([83.118,83.118],[0,0.012],'-k')

plt.show()

p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
```

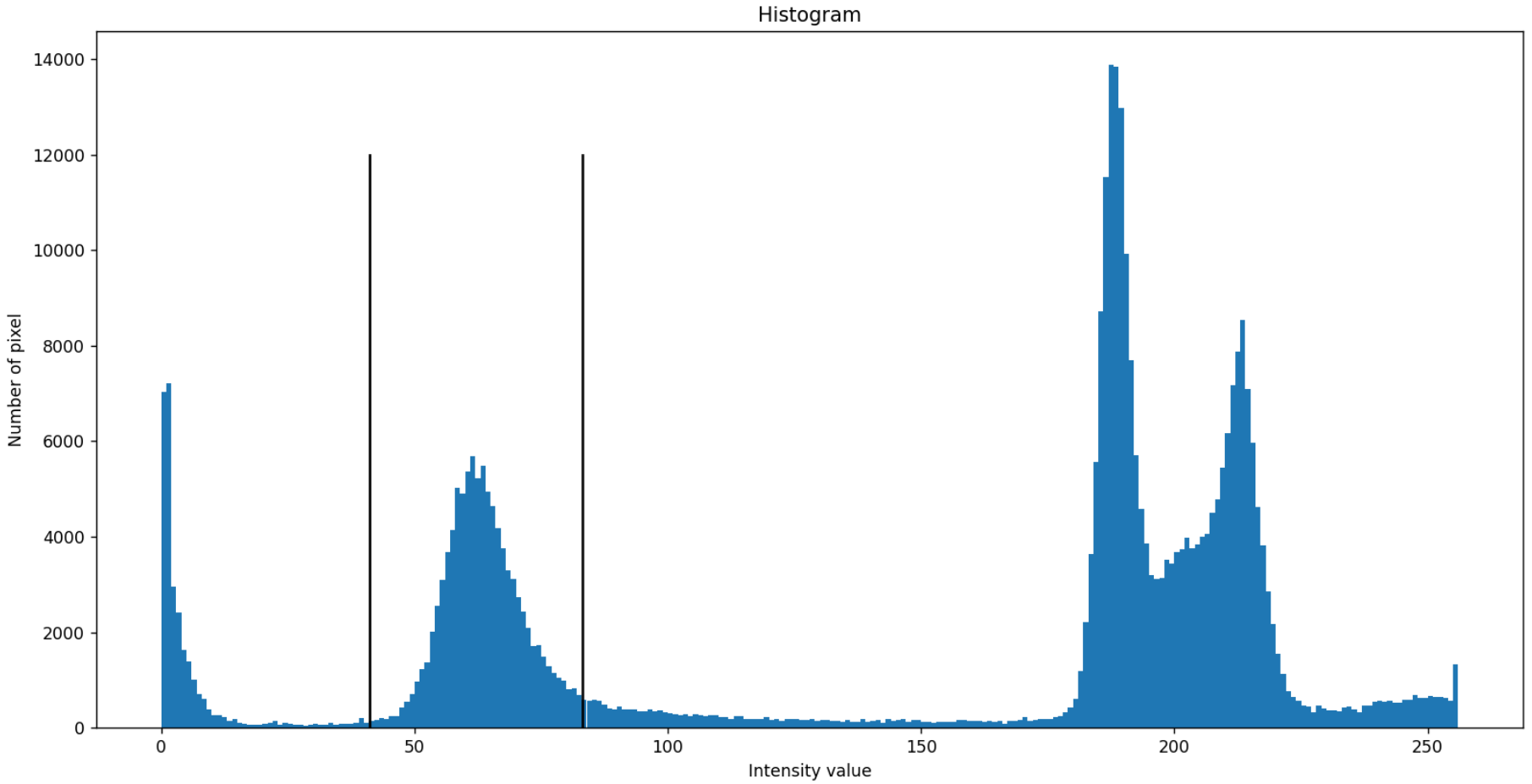
```
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
#sum_est = p1_est + p2_est + p3_est + p4_est+p5_est

plt.figure(5)
plt.plot(xl, p1_est + p2_est + p3_est+p4_est  , 'r--', linewidth=2.0)

plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([41.20,41.20],[0,0.012],'-k')
plt.plot([83.118,83.118],[0,0.012],'-k')

plt.title('Gaussian Distribution')
plt.show()

plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=41.20,vmax=83.118)
plt.title('Only Lung')

plt.show()
```

การแสดงผล

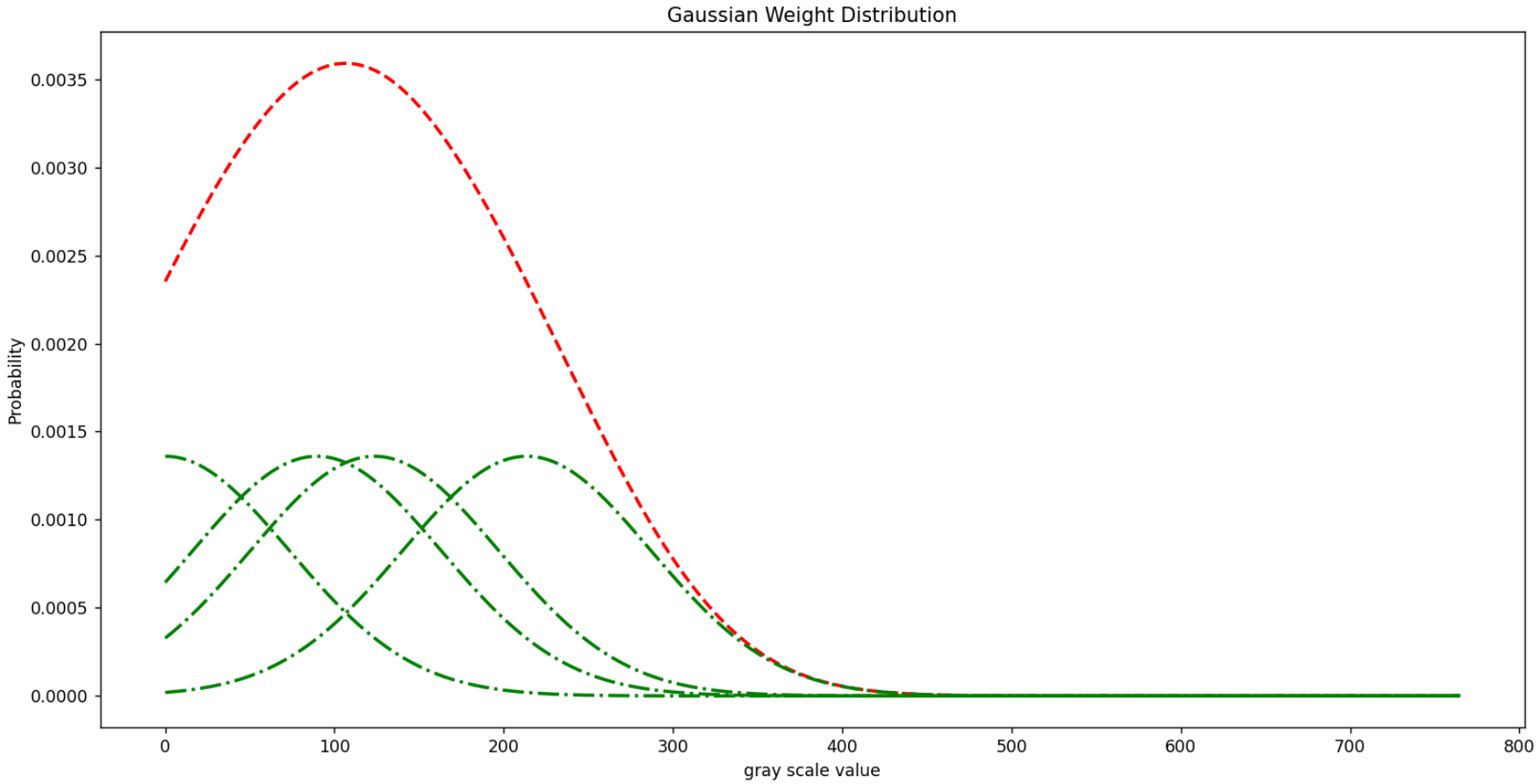
1.ภาพที่ Resize (370x480)



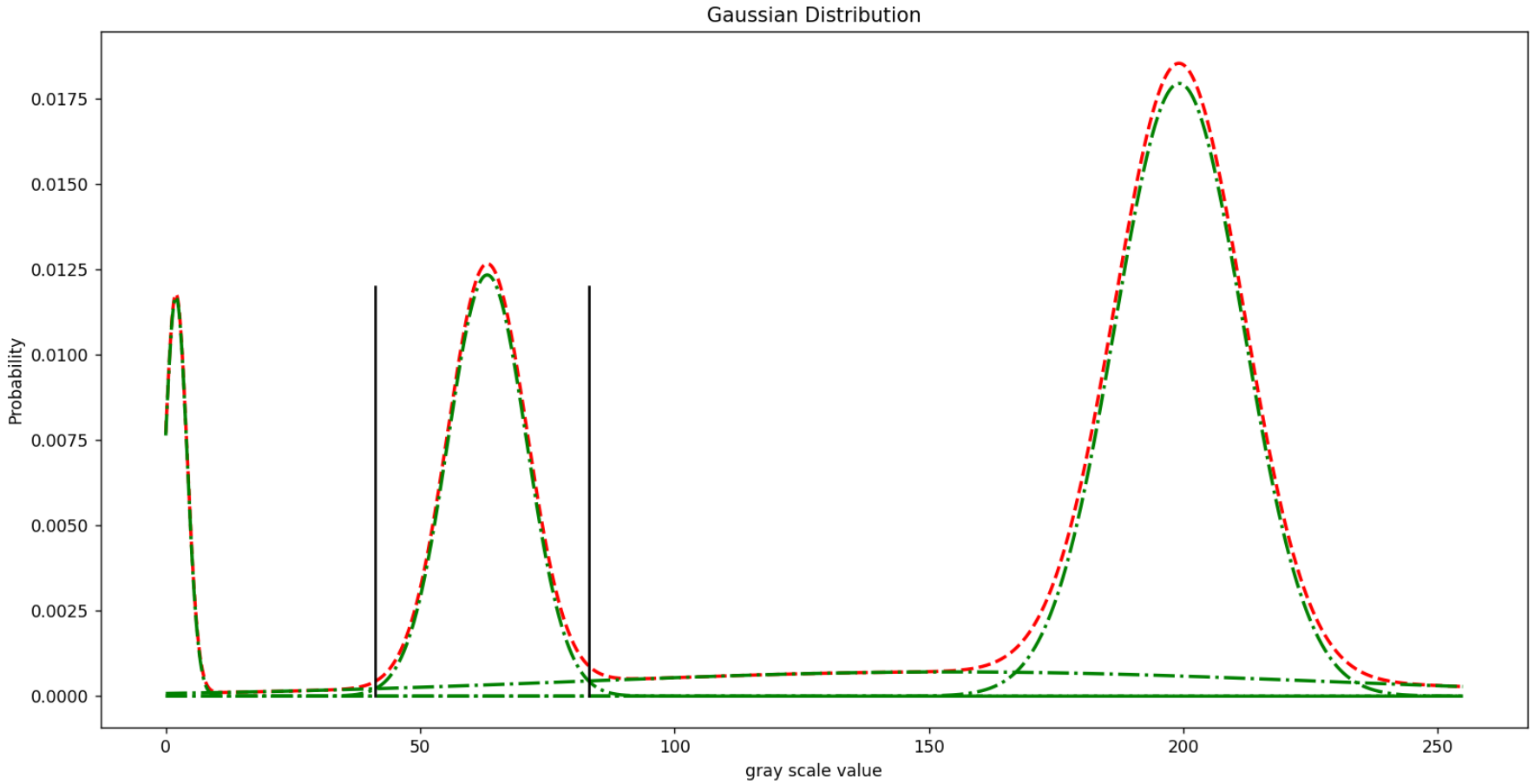
2.Histogram (c=4 ; Vmin=41.20 ; Vmax=83.118)



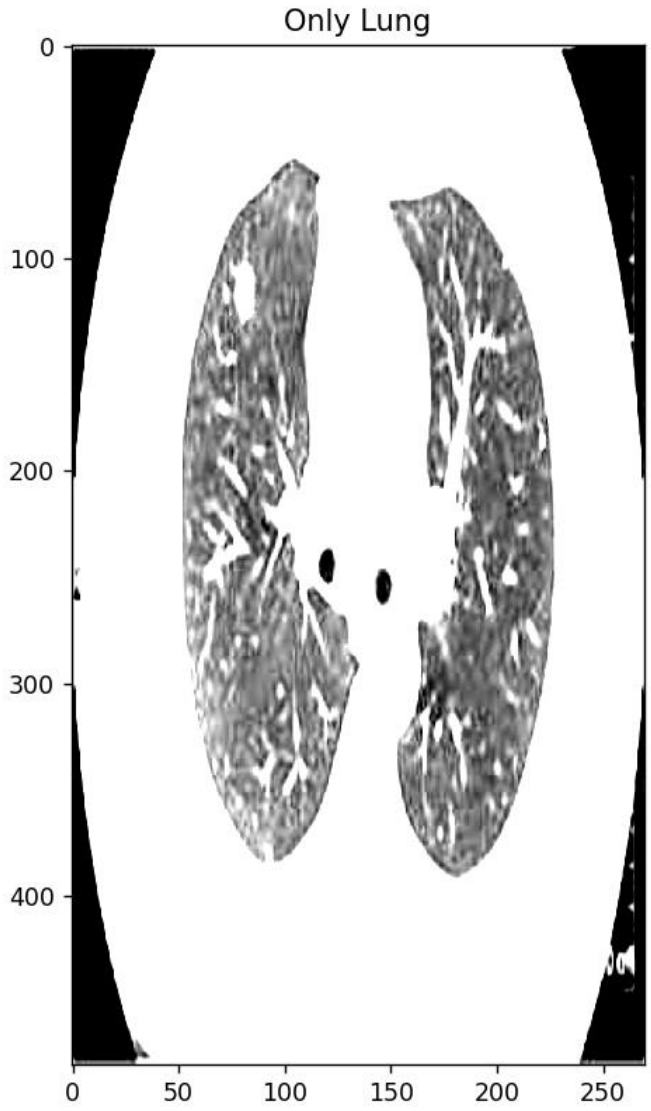
3.Gaussian Weight Distribution (c=4 ; Vmin=41.20 ; Vmax=83.118)



4.Gaussian Mixture Model (c=4 ; Vmin=41.20 ; Vmax=83.118)

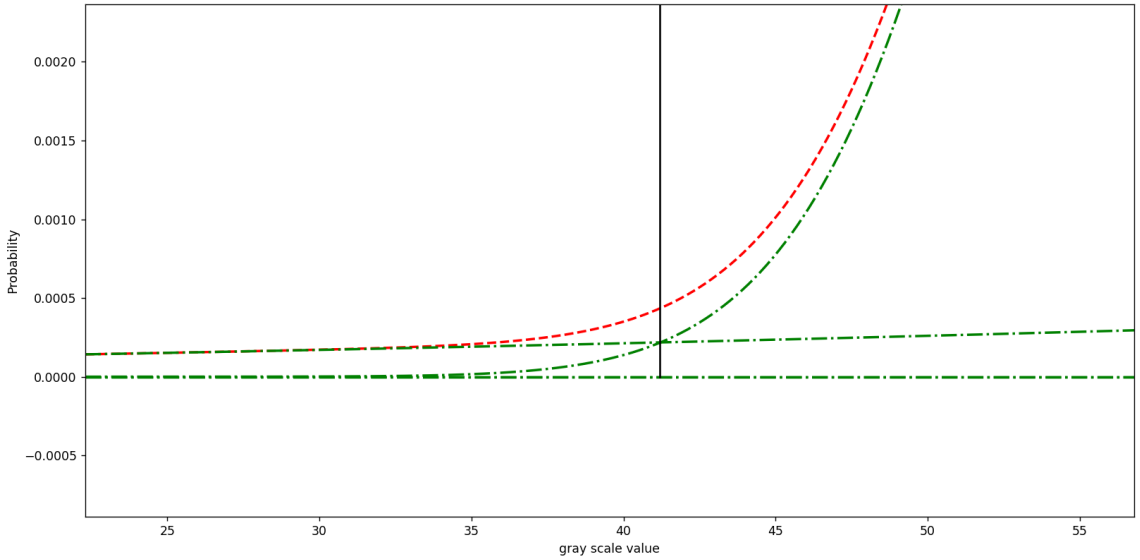


5.ภาพเฉพาะส่วนที่เป็นปอด

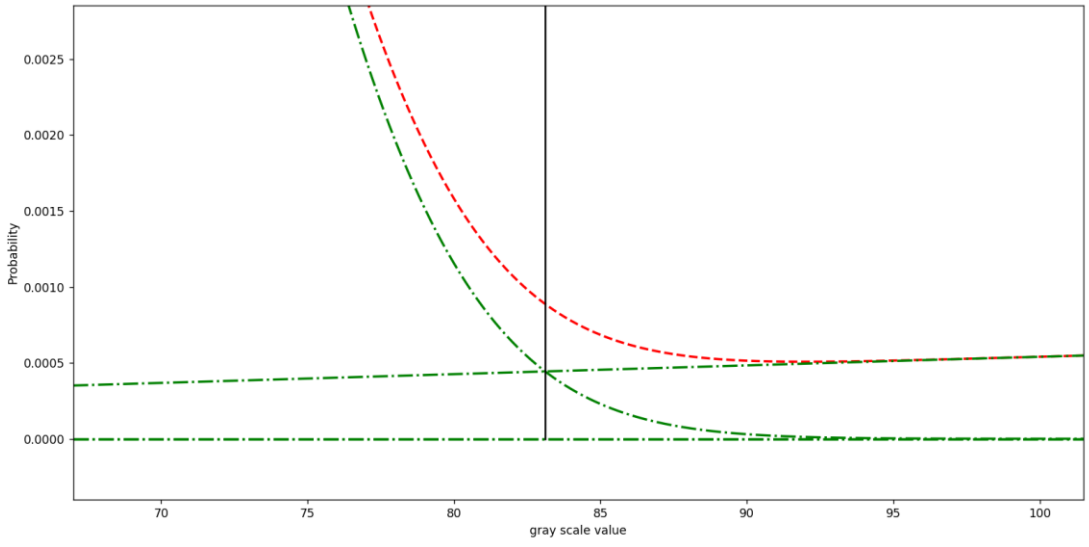


อภิปราย

-ขั้นตอนในการพิจารณาหาค่า V min และ V max ให้ทำการพิจารณาจากจุดตัดของกราฟ Gaussian Mixture Model ดังนี้



จากภาพจะสังเกตเห็นว่า จุดตัดของกราฟทั้ง2จะอยู่ที่ค่าประมาณ 41.20 ซึ่งเรากำหนดให้เป็น Vmin เนื่องจากเป็นจุดเดียวที่สามารถมองด้วยตาแล้วระบุค่าได้อย่างแม่นยำที่สุด ต่อมา



จากภาพ จุดตัดของกราฟทั้งสองจะอยู่ที่ประมาณ 83.118 เรากำหนดให้เป็น Vmax ด้วยเหตุผลเดียวกันกับ Vmin หลังจากที่เรากำหนดค่า Vmin และ Vmax เสร็จแล้ว นำค่าที่ได้ลงไปใส่ในคำสั่งต่อไปนี้

```
plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=41.20,vmax=83.118)
plt.title('Only Bones')
```

โดยภาพTCScan ปอดนั้นมีค่า Vmin และ Vmax เท่ากับ 41.20 และ 83.118 ซึ่งได้กำหนดในคำสั่งแล้ว

-ขั้นตอนต่อมา เมื่อเราได้ค่า Vmin และ Vmax แล้ว เราต้องไปกำหนดเส้นแบ่งใน Histogram และ Gaussian Mixture Model ด้วยคำสั่ง

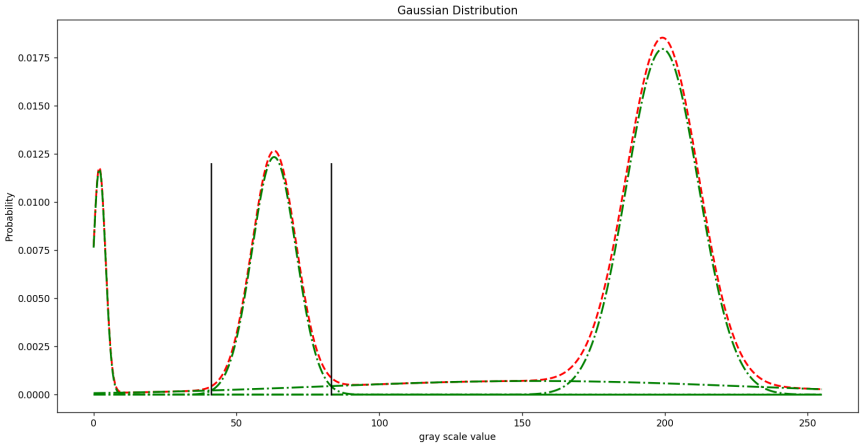
```
plt.plot([41.20,41.20],[0,12000],'-k')
plt.plot([83.118,83.118],[0,12000],'-k')
```

และ

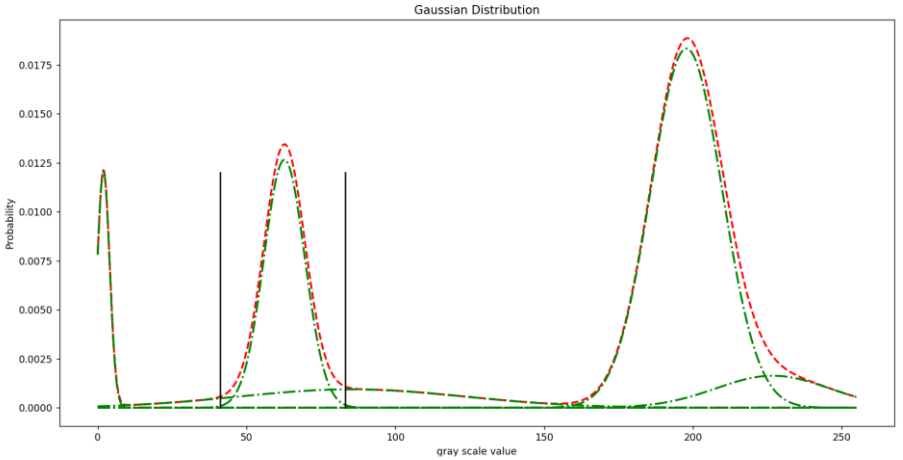
```
plt.plot([41.20,41.20],[0,0.012],'-k')
plt.plot([83.118,83.118],[0,0.012],'-k')
```

-การเลือกค่า c เราทำการเลือกค่า c=4และ5 เนื่องจากมีลักษณะคล้ายกับ Histogramมากที่สุด แต่เนื่องจากค่า c=5 จะเป็นการเสียเวลาในการประมวลผล ดังนั้นค่าที่เหมาะสมที่สุดคือ ค่า c=4 นั่นเอง สังเกตได้จากภาพด้านล่างนี้

ค่า c=4



ค่าc=5



จากภาพข้างต้น จะเห็นว่า กราฟ GMM มีลักษณะคล้ายกัน แทบไม่ต่างกัน ดังนั้นค่า c ที่เหมาะสมที่สุด คือ c=4

b. กระดูก

คำสั่งที่ใช้งาน

```
from PIL import Image
import numpy as np
import cv2
img = Image.open('D:\Telecom_Lab\Lab5\Figure\CTScan.jpg') # image extension *.png,*.jpg
new_width  = 270
new_height = 480
img = img.resize((new_width, new_height))
img.save('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg') # format may what u want
,*.png,*jpg,*.gif
from skimage.io import imread
from skimage.color import rgb2gray
mountain_r = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg')
#Plot
import matplotlib.pyplot as plt
plt.figure(0)
plt.imshow(mountain_r,cmap="gray")
plt.show()
img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg')
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
meandata = mean(data)
stddata = np.std(data)
x= meandata+(4*stddata)

c = np.arange(0, x, 1)
```

```
k = len(c)
i = len(data)
plt.figure(1)
hist,bin = np.histogram(data,c)
y = len(hist)
w = np.arange(0, x-1, 1)
r = hist/i
plt.bar(w,r)
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Probability density function')
plt.show()

plt.figure(2)
plt.hist(img.ravel(),256,[0,256])
plt.ylabel('Number of pixel')
plt.xlabel('Intensity value')
plt.plot([167.047,167.047],[0,12000],'-k')
plt.plot([233.83,233.83],[0,12000],'-k')
plt.title('Histogram')
plt.show()

img = cv2.imread('D:\Telecom_Lab\Lab5\Figure\CTScan_Resize.jpg',0)
arr = np.array(img)
data = np.reshape(arr, (1,np.product(arr.shape)))[0]

m = len(data)
epsilon = 1.0e-3
difference = epsilon
counter = 0
def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)
```

```
c = 4

from numpy.random import seed
from numpy.random import rand

seed(1)

mu_est = 2*mean(data)*np.sort(rand(c,1))

sigma_est = np.ones(c)*np.std(data)

p_est = np.ones(c)/c

def gaussian_norm_density(x, mu, sig):
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))/(sig * np.sqrt(2 *
np.pi))
d = max(data)

xl = np.arange(0, d*3,0.1)


p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);


plt.figure(3)
plt.plot(xl,p1_est+p2_est+p3_est+p4_est, 'r--',linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
```

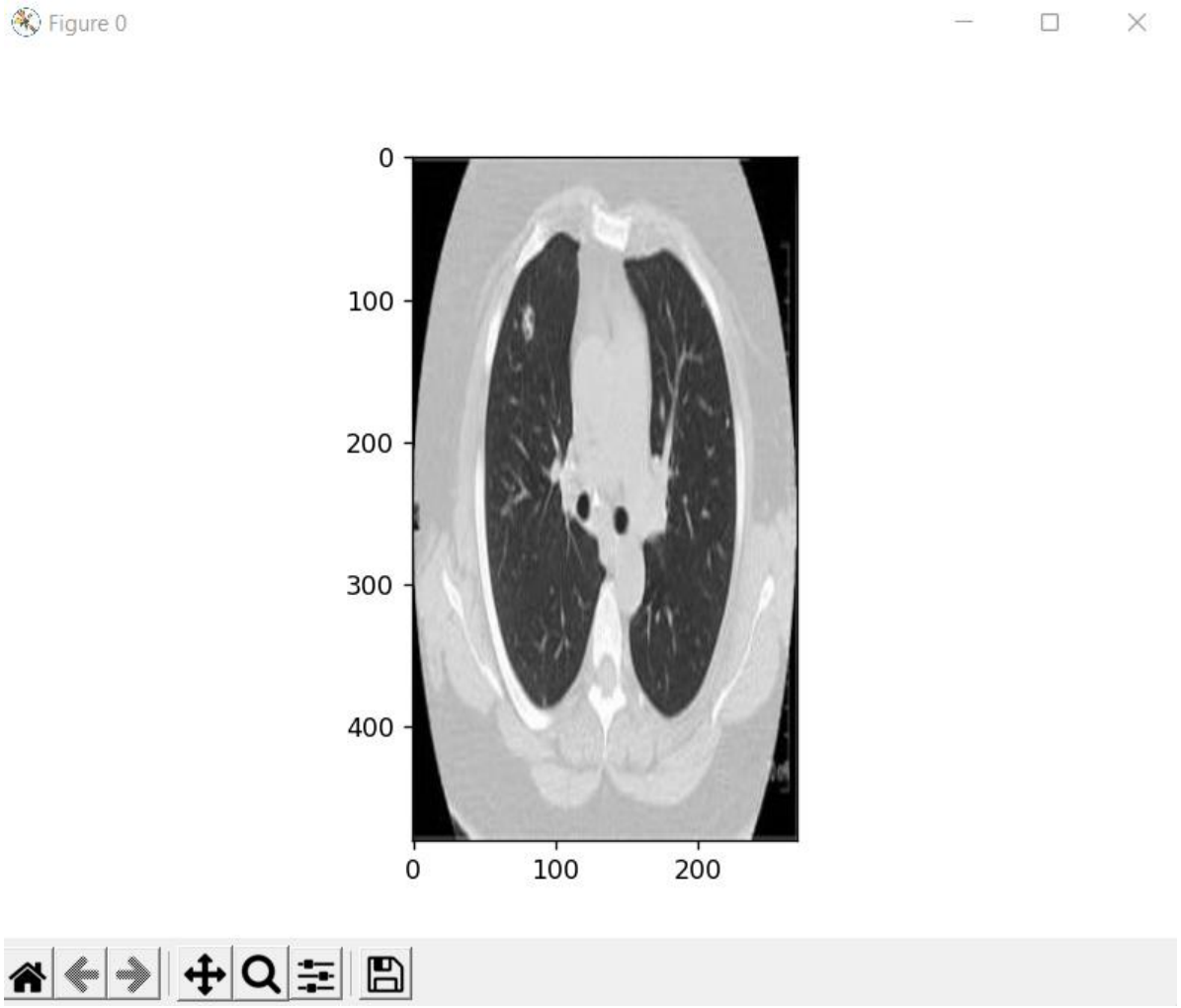
```
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);

plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.title('Gaussian Weight Distribution')
plt.show()

clas = []
ok = []
while np.any(difference >= epsilon) and (counter < 25000):
    for j in range(0, c):

        clas.insert(j, p_est[j] * gaussian_norm_density(data, mu_est[j], sigma_est[j]))

    ok = clas[0] + clas[1] + clas[2] + clas[3]

    for j in range(0, c):
        clas[j] = clas[j] / ok


    mu_est_old = mu_est
    sigma_est_old = sigma_est
    p_est_old = p_est
    mu_est = []
    sigma_est = []
    p_est = []

    for j in range(0, c):
        mu_est.insert(j, sum((clas[j]) * data) / sum(clas[j]))
        sigma_est.insert(j, np.sqrt(sum((clas[j]) * np.power((data - mu_est[j]), 2)) /
sum(clas[j])))
        p_est.insert(j, mean(clas[j]))
```

```
    difference =
sum(abs(np.subtract(mu_est_old,mu_est)))+sum(abs(np.subtract(sigma_est_old,sigma_est)))\
                +sum(abs(np.subtract(p_est_old,p_est)))

    print(difference)

    counter = counter + 800
    print('counter =',counter)
xl = np.arange(0, d, 0.1)
p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
plt.figure(4)
plt.plot(xl, p1_est + p2_est + p3_est+p4_est , 'r--', linewidth=2.0)
plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([167.047,167.047],[0,0.012],'-k')
plt.plot([233.83,233.83],[0,0.012],'-k')

plt.show()

p1_est = p_est[0] * gaussian_norm_density(xl, mu_est[0], sigma_est[0]);
p2_est = p_est[1] * gaussian_norm_density(xl, mu_est[1], sigma_est[1]);
p3_est = p_est[2] * gaussian_norm_density(xl, mu_est[2], sigma_est[2]);
p4_est = p_est[3] * gaussian_norm_density(xl, mu_est[3], sigma_est[3]);
#sum_est = p1_est + p2_est + p3_est + p4_est+p5_est
```

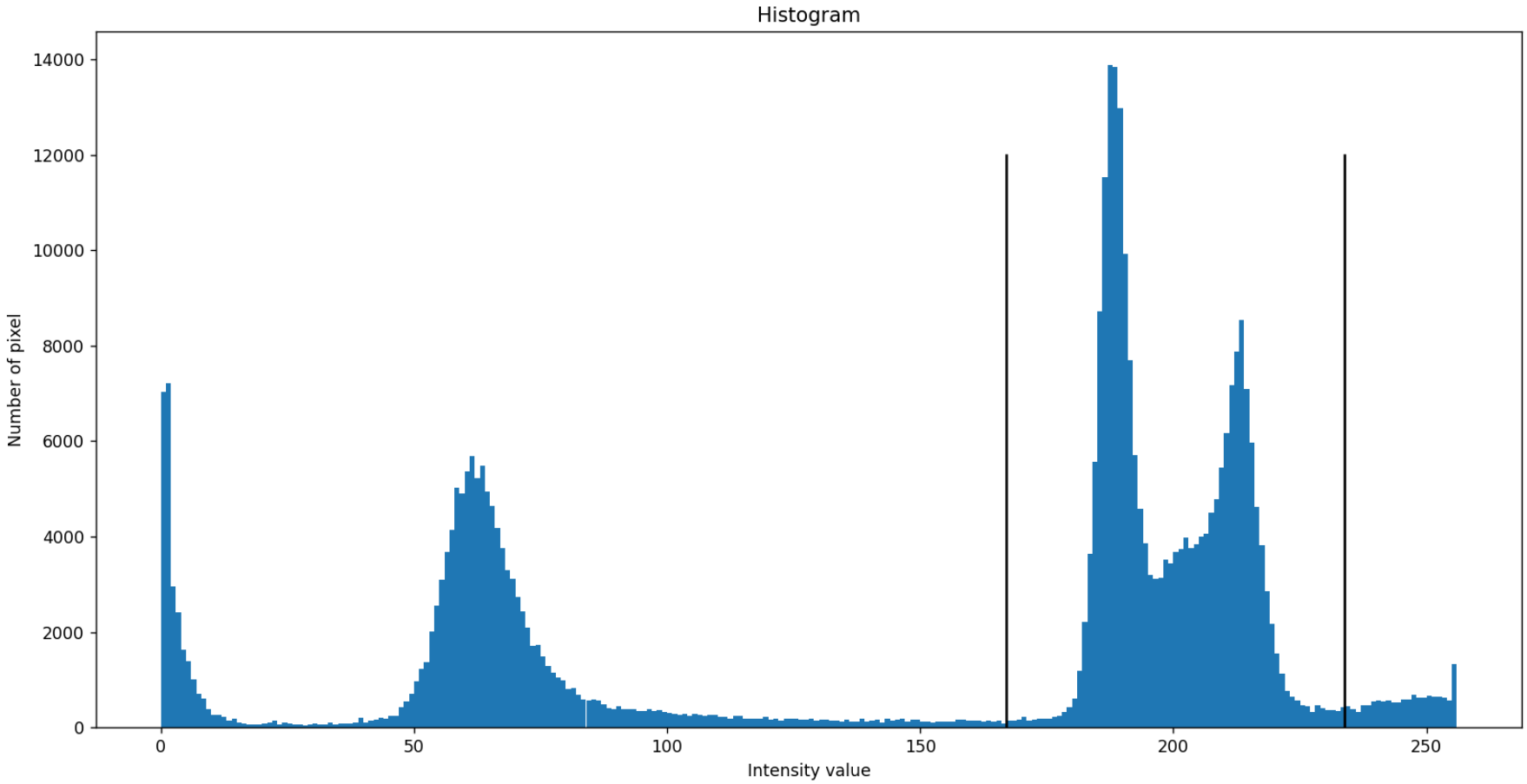
```
plt.figure(5)
plt.plot(xl, p1_est + p2_est + p3_est+p4_est  , 'r--', linewidth=2.0)

plt.plot(xl, p1_est, 'g-.', linewidth=2.0);
plt.plot(xl, p2_est, 'g-.', linewidth=2.0);
plt.plot(xl, p3_est, 'g-.', linewidth=2.0);
plt.plot(xl, p4_est, 'g-.', linewidth=2.0);
plt.xlabel('gray scale value')
plt.ylabel('Probability')
plt.plot([167.047,167.047],[0,0.012],'-k')
plt.plot([233.83,233.83],[0,0.012],'-k')

plt.title('Gaussian Distribution')
plt.show()

plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=167.047,vmax=233.83)
plt.title('Only Bones')

plt.show()
```

การแสดงผล

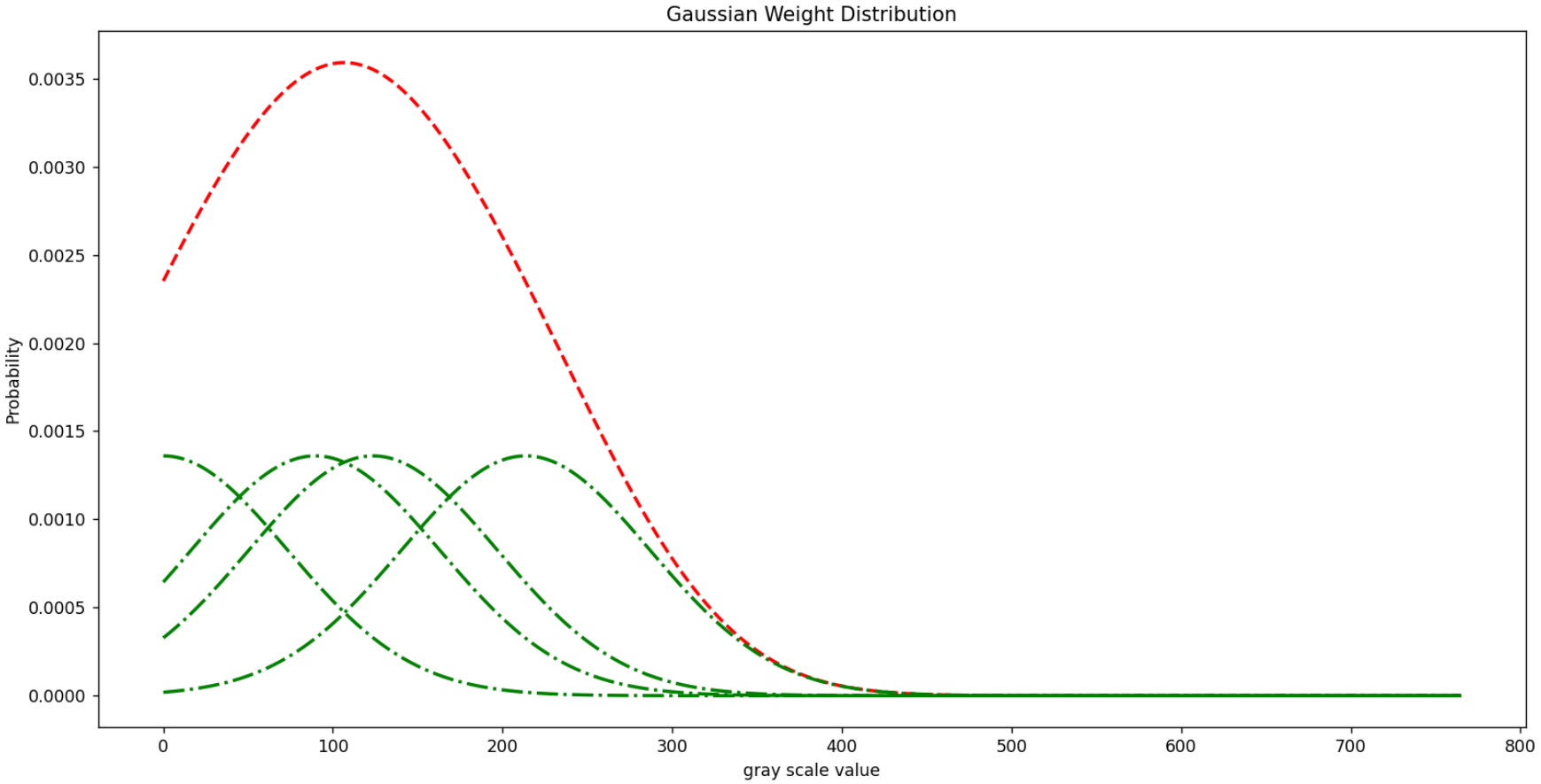
1.ภาพที่Resize (270x480)



2.Histogram (c=4 ; Vmin=167.047 ; Vmax=233.83)



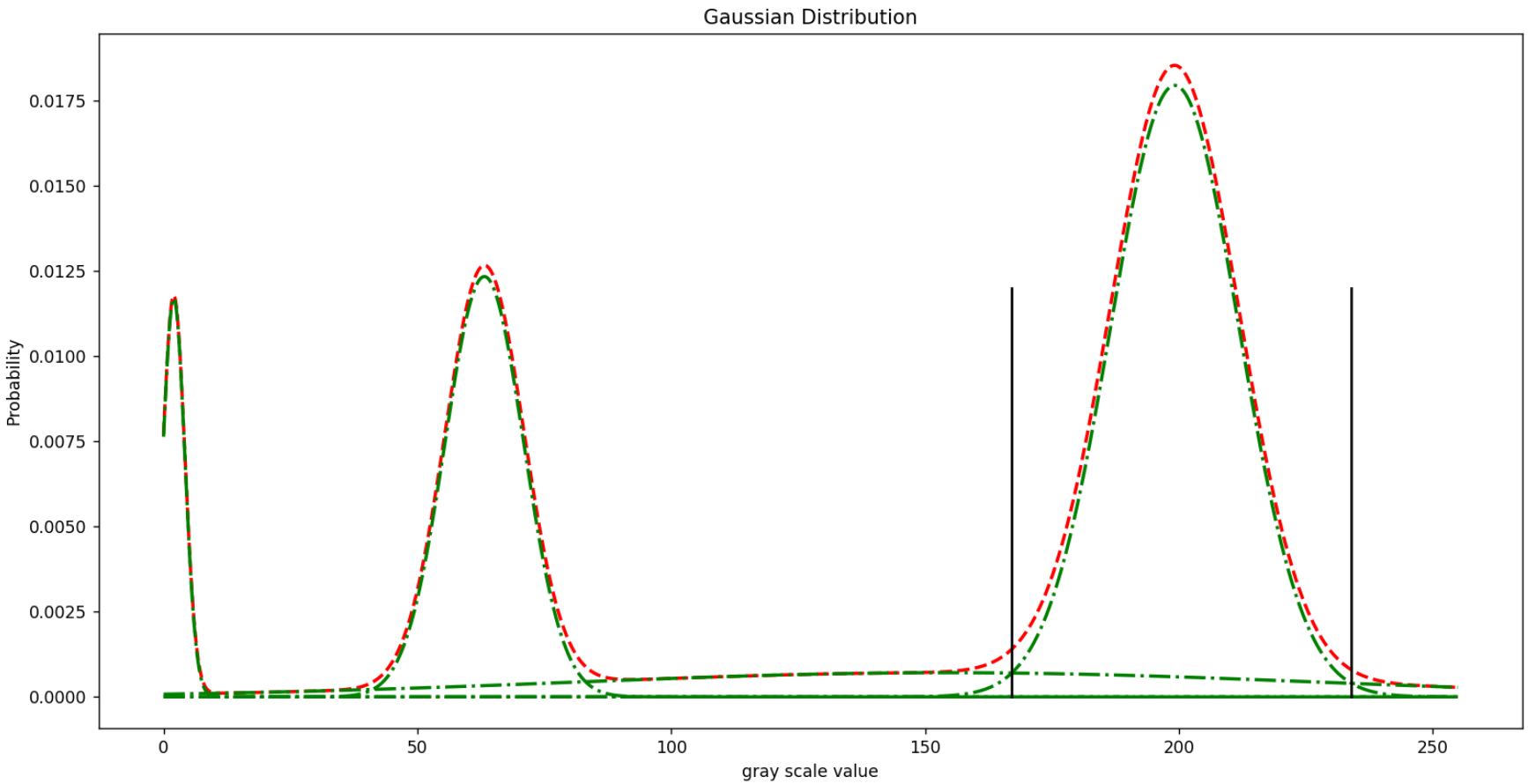
3.Gaussian Weight Distribution (c=4 ; Vmin=167.047 ; Vmax=233.83)



หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

4.Gaussian Mixture Model (c=4 ; Vmin=167.047 ; Vmax=233.83)



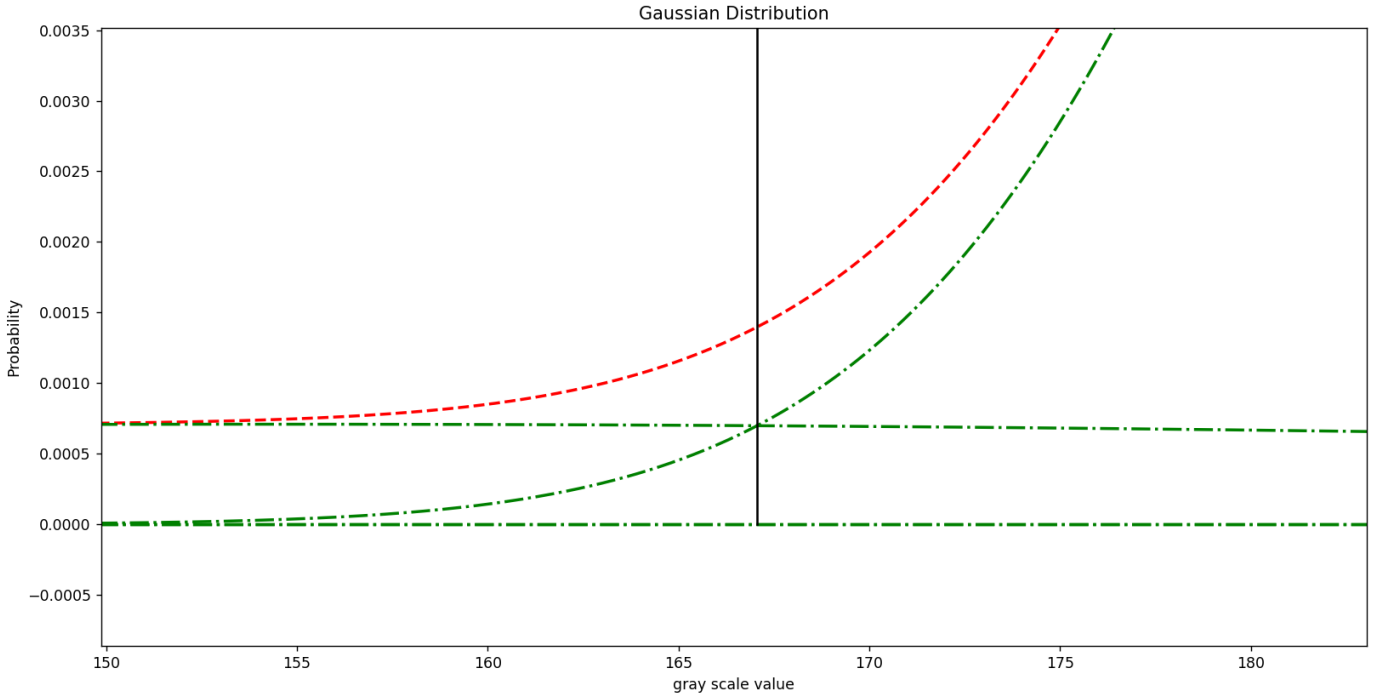
หมายเหตุ : กราฟเส้นปะสีแดง แทน Estimated Model

กราฟเส้นปะสีเขียว แทน Component ของ Gaussian Distribution

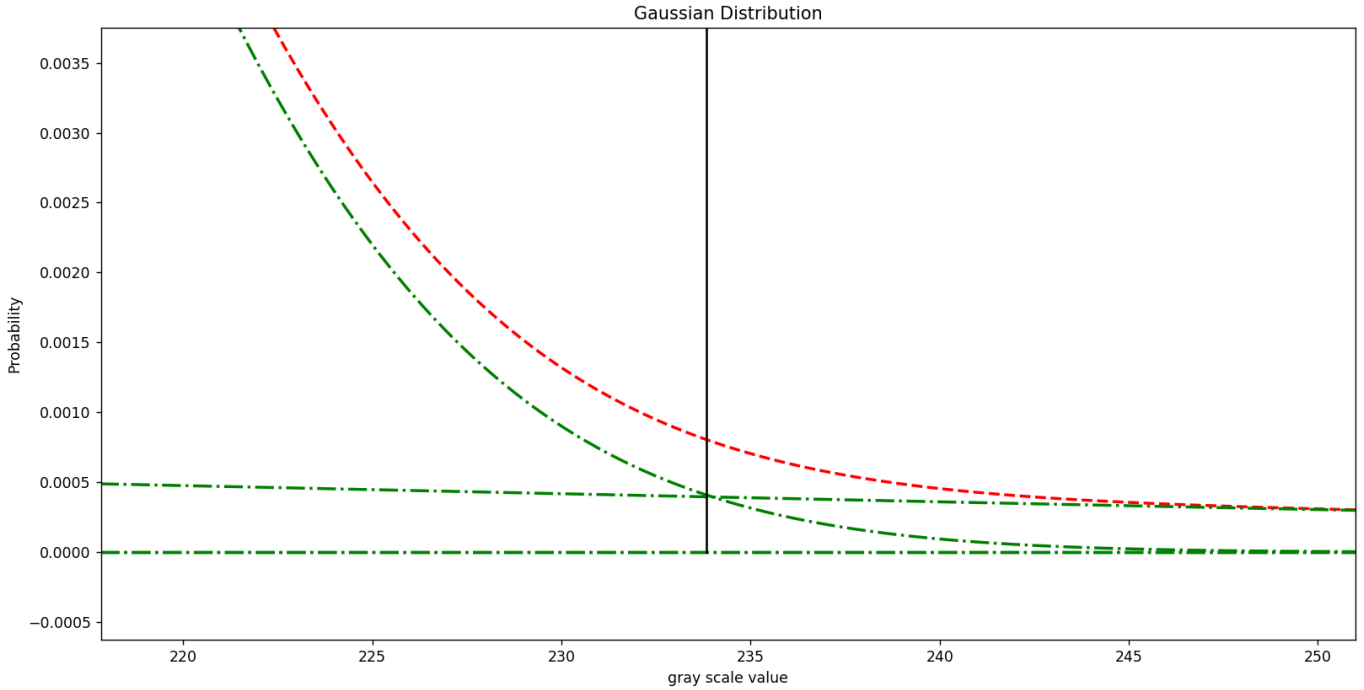
5.ภาพที่แสดงเฉพาะส่วนของกระดูก

อภิปราย

-จากการทดลอง เช่นเคย เราต้องทำการเลือกค่า Vmin และ Vmax ก่อน เนื่องจากเราต้องการให้แสดงผลเฉพาะส่วนของปอด วิธีการทำคือการไประบุจุดที่ตัดกันของกราฟ Gaussian Mixture Model ของภาพ TCScan ดังภาพต่อไปนี้



จากภาพข้างต้น แสดงจุดตัดของกราฟ ณ ตำแหน่ง 167.047 ซึ่งกำหนดให้เป็น Vmin ขั้นตอนต่อมา



จากภาพข้างต้น แสดงจุดตัดของกราฟ ณ ตำแหน่ง 233.83 ซึ่งกำหนดให้เป็น V Max ตามลำดับ

สำหรับขั้นตอนต่อมา ทำการป้อนคำสั่ง

```
plt.figure(6)
plt.imshow(arr,cmap='gray',vmin=167.047,vmax=233.83)
plt.title('Only Bones')
```

โดยภาพTCScan ปอดนั้นมีค่า Vmax และ Vmin คือ 167.047และ233.83 ตามลำดับ ซึ่งได้กำหนดในคำสั่งแล้ว

-ขั้นตอนต่อมา เมื่อเราได้ค่า Vmin และ Vmax แล้ว เราต้องไปกำหนดเส้นแบ่งใน Histogram และ Gaussian Mixture Model ด้วยคำสั่ง

```
plt.plot([167.047,167.047],[0,12000],'-k')
plt.plot([233.83,233.83],[0,12000],'-k')
```

และ

```
plt.plot([167.047,167.047],[0,0.012],'-k')
plt.plot([233.83,233.83],[0,0.012],'-k')
```

-ส่วนการเลือกค่า c จะเป็นเช่นเดียวกับกรณีที่ต้องการแค่ ปอด เนื่องจากใช้ภาพ TCScan ภาพเดียวกันนั่นเอง