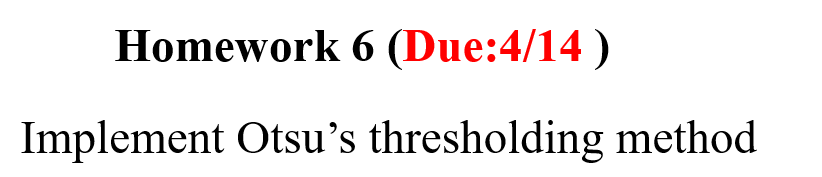
**40847041S朱自宇 hw3**

**Problem statement:**



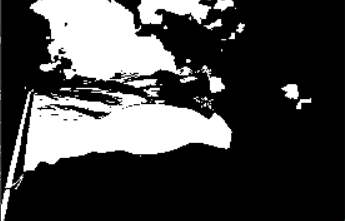
**Experimental results:**

* **Input/Output images:**

**Input grayscale image:**

****

**Output Otsu’s Thresholding Image:**



* **Source code:**

"""

110-2 NTNU CSIE

ImageProcessing

Assignment6

by 40847041S 朱自宇

"""

import numpy as np

import cv2

import matplotlib.pyplot as plt

# read the gray image

grayImg = cv2.imread("Ukraine.png", cv2.IMREAD\_GRAYSCALE)

# gray scale count of the image

grayHE = np.zeros(256,dtype=np.uint32)

for i in grayImg:

for j in i:

grayHE[j]+=1

# total size of image

ImgSize = grayImg.shape[0]\*grayImg.shape[1]

# record the Max t

OtsuMax = 0

t = 0

temp = 0

# calculate m

m = 0

for i in range(256):

m += i\*grayHE[i]

m = m / ImgSize

# find t

for i in range(256):

a\_t = 0

b\_t = 0

m\_a = 0

for j in range(i+1):

a\_t += grayHE[j]

m\_a += j\*grayHE[j]

b\_t = (ImgSize - a\_t)/ImgSize

a\_t = a\_t / ImgSize

m\_a = m\_a / ImgSize

if a\_t != 0 and b\_t != 0:

temp = ((m\_a - m\*a\_t)\*\*2)/(a\_t\*b\_t)

else: temp = 0

if OtsuMax < temp:

OtsuMax = temp

t = i

grayImg\_flat = grayImg.flatten()

output = np.zeros(grayImg\_flat.size,dtype=np.uint8)

for i in range(grayImg\_flat.size):

if grayImg\_flat[i] < t:

output[i] = 0

else: output[i] = 255

output = output.reshape( (grayImg.shape[0],grayImg.shape[1]) )

# show the original gray image

cv2.imshow("Input Gray image", grayImg)

# show the output gray image

cv2.imshow("Output Otsu threshold image", output)

# wait until the user press any button to close the img window

cv2.waitKey(0)

cv2.destroyAllWindows()

* **Comments:**

This is cool! Just separate the image into half but still looks nice.