**การกำหนดเกณฑ์ภาพ**

เป้าหมาย

* ในบทแนะนำนี้คุณจะได้เรียนรู้เกี่ยวกับเกณฑ์ง่ายๆ thresholding แบบปรับได้การกำหนดเกณฑ์ของ Otsu เป็นต้น
* คุณจะได้เรียนรู้ฟังก์ชันเหล่านี้: **cv2.threshold** , **cv2.adaptiveThreshold**เป็นต้น

เกณฑ์ง่าย

นี่เป็นเรื่องตรงไปข้างหน้า ถ้าค่าพิกเซลสูงกว่าค่า threshold จะกำหนดค่าหนึ่งค่า (อาจเป็นสีขาว) ค่าอื่นจะกำหนดค่าอื่น (อาจเป็นสีดำ) ฟังก์ชั่นที่ใช้เป็น cv2.threshold อาร์กิวเมนต์แรกเป็นภาพต้นฉบับซึ่งควรจะเป็น should be a grayscale image. ที่สองคือค่าเกณฑ์ที่ใช้ในการจำแนกค่าพิกเซล อาร์กิวเมนต์ที่สามคือ maxVal ซึ่งแสดงถึงค่าที่จะได้รับถ้าค่าพิกเซลมีค่า threshold มากกว่า (บางครั้งอาจน้อยกว่า) OpenCV มีลักษณะการกำหนดเกณฑ์ที่แตกต่างกันและมีการตัดสินใจด้วยพารามิเตอร์ที่สี่ของฟังก์ชัน ประเภทต่างๆ:

* cv2.THRESH\_BINARY
* cv2.THRESH\_BINARY\_INV
* cv2.THRESH\_TRUNC
* cv2.THRESH\_TOZERO
* cv2.THRESH\_TOZERO\_INV

เอกสารอธิบายอย่างชัดเจนว่าแต่ละประเภทมีความหมายอย่างไร โปรดตรวจสอบเอกสาร

เอาต์พุตสองชุด อันดับแรกคือ retval ซึ่งจะอธิบายในภายหลัง เอาท์พุทที่สองคือเราภาพ thresholded image.

Code :

**import** **cv2**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

img = cv2.imread('gradient.png',0)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY\_INV)

ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH\_TRUNC)

ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO)

ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO\_INV)

titles = ['Original Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']

images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

**for** i **in** xrange(6):

plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')

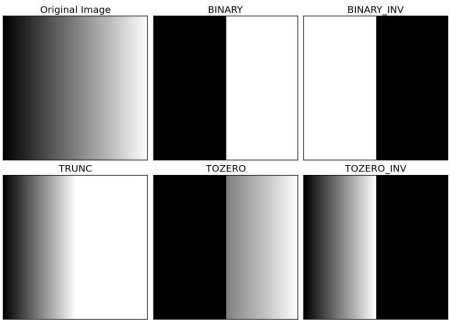
plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

**บันทึก :** ในการทำพล็อตหลายภาพเราได้ใช้ *plt.subplot ()* function กรุณาตรวจสอบเอกสาร Matplotlib สำหรับรายละเอียดเพิ่มเติม

ผลได้ดังนี้



การปรับเกณฑ์เหมาะสม

ในส่วนก่อนหน้าเราใช้ค่าทั่วโลกเป็นค่าเกณฑ์ แต่อาจไม่ดีในทุกสภาวะที่ภาพมีสภาพแสงต่างกันในแต่ละพื้นที่ ในกรณีนี้เราจะไปปรับเกณฑ์ ในขั้นตอนวิธีนี้จะคำนวณเกณฑ์สำหรับพื้นที่เล็ก ๆ ของภาพ ดังนั้นเราจึงได้รับเกณฑ์ที่แตกต่างกันสำหรับภูมิภาคต่างๆของภาพเดียวกันและทำให้เราได้ผลลัพธ์ที่ดีขึ้นสำหรับภาพที่มีการส่องสว่างที่แตกต่างกัน

มีพารามิเตอร์ params อินพุท 'params พิเศษ' และอาร์กิวเมนต์เอาต์พุตเพียงชุดเดียว

**Adaptive Method** - จะตัดสินใจว่าจะคำนวณค่า thresholding ได้อย่างไร

* cv2.ADAPTIVE\_THRESH\_MEAN\_C: ค่า threshold คือค่าเฉลี่ยของพื้นที่ใกล้เคียง
* cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C: ค่า threshold เป็นผลรวมถ่วงน้ำหนักของค่าที่อยู่ใกล้เคียงซึ่งน้ำหนักเป็นหน้าต่าง gaussian

**Block Size** - จะกำหนดขนาดของพื้นที่ใกล้เคียง

**C** - เป็นค่าคงที่ซึ่งถูกหักออกจากค่าเฉลี่ยหรือค่าถัวเฉลี่ยถ่วงน้ำหนัก

ด้านล่างของโค้ดเปรียบเทียบ thresholding ทั่วโลกและ thresholding ปรับสำหรับภาพที่มีการส่องสว่างที่แตกต่างกัน:

**import** **cv2**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

img = cv2.imread('dave.jpg',0)

img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\

cv2.THRESH\_BINARY,11,2)

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\

cv2.THRESH\_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',

'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

**for** i **in** xrange(4):

plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

Result :



Otsu ของ Binarization

ในส่วนแรกผมบอกคุณมีพารามิเตอร์ที่สอง RetValใช้มันมาเมื่อเราไปสำหรับ Otsu ของ Binarization ดังนั้นมันคืออะไร?

ในเกณฑ์ระดับโลกเราใช้ค่าตามอำเภอใจสำหรับค่าเกณฑ์ใช่มั้ย? ดังนั้นเราจะรู้ได้อย่างไรว่าค่าที่เราเลือกนั้นดีหรือไม่? คำตอบคือวิธีการทดลองและข้อผิดพลาด แต่พิจารณา**ภาพ bimodal image** ( *ในคำง่ายๆภาพสองส่วนคือภาพที่มีฮิสโทแกรมมีสองยอด* histogram ) สำหรับภาพนั้นเราสามารถประมาณค่าในช่วงกลางของยอดเหล่านี้เป็นค่า threshold ได้หรือไม่? นั่นคือสิ่งที่ Otsu binarization ไม่ ดังนั้นในคำง่ายๆจะคำนวณค่า threshold จาก histogram ภาพสำหรับรูปแบบ bimodal โดยอัตโนมัติ (สำหรับภาพที่ไม่ใช่แบบ bimodal การ binarization จะไม่ถูกต้อง)

สำหรับเรื่องนี้ cv2.threshold เรา () ฟังก์ชั่นคือการใช้ แต่ผ่านธงพิเศษcv2.THRESH\_OTSU For threshold value, simply pass zero.สำหรับค่าเกณฑ์เพียงผ่านศูนย์จากนั้นขั้นตอนวิธีพบว่าค่าเกณฑ์ที่เหมาะสมและผลตอบแทนที่คุณเป็นผู้ส่งออกที่สองRetVal หากไม่ได้ใช้เกณฑ์การ Otsu retVal จะเหมือนกับค่าเกณฑ์ที่คุณใช้

ลองดูตัวอย่างด้านล่าง ภาพอินพุตเป็นภาพที่มีเสียงดัง ในกรณีแรกฉันใช้เกณฑ์ระดับโลกสำหรับค่า 127 ในกรณีที่สองฉันใช้การกำหนดเกณฑ์ของ Otsu โดยตรง ในกรณีที่สามฉันกรองภาพด้วยเคอร์เนลแบบ Gaussian 5x5 เพื่อขจัดเสียงดังกล่าวและใช้เกณฑ์ Otsu ดูวิธีการกรองเสียงช่วยเพิ่มผลลัพธ์

**import** **cv2**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

img = cv2.imread('noisy2.png',0)

*# global thresholding*

ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

*# Otsu's thresholding*

ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

*# Otsu's thresholding after Gaussian filtering*

blur = cv2.GaussianBlur(img,(5,5),0)

ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

*# plot all the images and their histograms*

images = [img, 0, th1,

img, 0, th2,

blur, 0, th3]

titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',

'Original Noisy Image','Histogram',"Otsu's Thresholding",

'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

**for** i **in** xrange(3):

plt.subplot(3,3,i\*3+1),plt.imshow(images[i\*3],'gray')

plt.title(titles[i\*3]), plt.xticks([]), plt.yticks([])

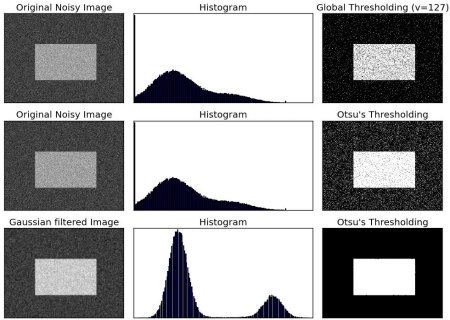
plt.subplot(3,3,i\*3+2),plt.hist(images[i\*3].ravel(),256)

plt.title(titles[i\*3+1]), plt.xticks([]), plt.yticks([])

plt.subplot(3,3,i\*3+3),plt.imshow(images[i\*3+2],'gray')

plt.title(titles[i\*3+2]), plt.xticks([]), plt.yticks([])

plt.show()



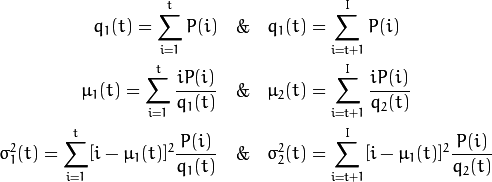
วิธี Binarization Otsu ทำงานอย่างไร

ส่วนนี้แสดงให้เห็นถึงการใช้งาน binary ของ Otsu ใน Python เพื่อแสดงวิธีการทำงานจริง ถ้าคุณไม่สนใจคุณสามารถข้ามสิ่งนี้ได้

เนื่องจากเรากำลังทำงานกับรูปแบบ bimodal อัลกอริธึมของ Otsu จึงพยายามหาค่า threshold (t) ซึ่งช่วยลด**ความแปรปรวนภายในชั้นที่ถ่วงน้ำหนัก**ได้โดยความสัมพันธ์:

\ sigma_w ^ 2 (t) = q_1 (t) \ sigma_1 ^ 2 (t) + q_2 (t) \ sigma_2 ^ 2 (t)

ที่ไหน



มันจริงพบว่าค่าของ t ซึ่งอยู่ในระหว่างสองยอดเช่นที่ variances ไปทั้งสองชั้นต่ำสุด สามารถใช้งานได้ง่ายใน Python ดังนี้

img = cv2.imread('noisy2.png',0)

blur = cv2.GaussianBlur(img,(5,5),0)

*# find normalized\_histogram, and its cumulative distribution function*

hist = cv2.calcHist([blur],[0],None,[256],[0,256])

hist\_norm = hist.ravel()/hist.max()

Q = hist\_norm.cumsum()

bins = np.arange(256)

fn\_min = np.inf

thresh = -1

**for** i **in** xrange(1,256):

p1,p2 = np.hsplit(hist\_norm,[i]) *# probabilities*

q1,q2 = Q[i],Q[255]-Q[i] *# cum sum of classes*

b1,b2 = np.hsplit(bins,[i]) *# weights*

*# finding means and variances*

m1,m2 = np.sum(p1\*b1)/q1, np.sum(p2\*b2)/q2

v1,v2 = np.sum(((b1-m1)\*\*2)\*p1)/q1,np.sum(((b2-m2)\*\*2)\*p2)/q2

*# calculates the minimization function*

fn = v1\*q1 + v2\*q2

**if** fn < fn\_min:

fn\_min = fn

thresh = i

*# find otsu's threshold value with OpenCV function*

ret, otsu = cv2.threshold(blur,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

**print** thresh,ret