**ฮิสโตแกรม - 2: Histogram Equalization**

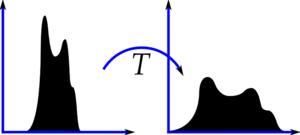
เป้าหมาย

ในส่วนนี้,

* เราจะเรียนรู้แนวคิดเรื่องการปรับสมดุลของฮิสโตแกรมและใช้เพื่อปรับปรุงความคมชัดของภาพของเรา

ทฤษฎี

พิจารณาภาพที่มีค่าพิกเซล จำกัด เฉพาะบางช่วงของค่าเท่านั้น เช่นภาพสว่างจะมีพิกเซลทั้งหมด จำกัด ด้วยค่าที่สูง แต่ภาพที่ดีจะมีพิกเซลจากทุกส่วนของภาพ ดังนั้นคุณจึงจำเป็นต้องยืดฮิสโตแกรมนี้ไปที่ปลายทั้งสองด้าน (ดังที่แสดงไว้ด้านล่างภาพจากวิกิพีเดีย) และนั่นคือสิ่งที่ Histogram Equalization ทำ (เป็นคำง่ายๆ) ซึ่งปกติจะเพิ่มความคมชัดของภาพ



ขอแนะนำให้คุณอ่านหน้าวิกิพีเดียเรื่อง[Histogram Equalization](http://en.wikipedia.org/wiki/Histogram_equalization)เพื่อดูรายละเอียดเพิ่มเติม มีคำอธิบายที่ดีด้วยตัวอย่างที่ดีเพื่อให้คุณเข้าใจเกือบทุกอย่างหลังจากอ่านแล้ว แต่ที่นี่เราจะเห็นการใช้ Numpy ของ หลังจากนั้นเราจะเห็นฟังก์ชัน OpenCV

**import** **cv2**

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

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

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

hist,bins = np.histogram(img.flatten(),256,[0,256])

cdf = hist.cumsum()

cdf\_normalized = cdf \* hist.max()/ cdf.max()

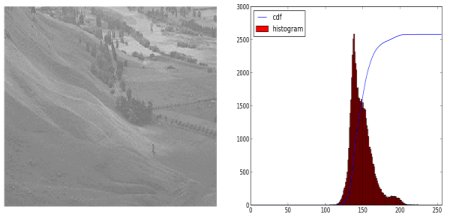
plt.plot(cdf\_normalized, color = 'b')

plt.hist(img.flatten(),256,[0,256], color = 'r')

plt.xlim([0,256])

plt.legend(('cdf','histogram'), loc = 'upper left')

plt.show()



คุณสามารถดูฮิสโทแกรมได้ในบริเวณที่สว่างขึ้น เราต้องการคลื่นเต็มรูปแบบ สำหรับที่เราต้องการฟังก์ชั่นการแปลงที่แมปพิกเซลเข้าในพื้นที่สว่างเป็นพิกเซลเอาต์พุตในพื้นที่เต็มรูปแบบ นั่นคือสิ่งที่ทำให้เท่าเทียมกัน histogram ไม่

ตอนนี้เราหาค่าฮิสโตแกรมต่ำสุด (ไม่รวม 0) และใช้สมการการปรับสมดุลของฮิสโตแกรมตามที่ระบุไว้ในหน้าวิกิพีเดีย แต่ฉันได้ใช้ที่นี่อาเรย์อาเรย์อาเรย์จาก Numpy สำหรับอาร์เรย์ที่สวมหน้ากากการดำเนินการทั้งหมดจะดำเนินการกับองค์ประกอบที่ไม่ได้สวมหน้ากาก คุณสามารถอ่านข้อมูลเพิ่มเติมได้จากเอกสาร Numpy บนอาร์เรย์ที่ปิดบัง

cdf\_m = np.ma.masked\_equal(cdf,0)

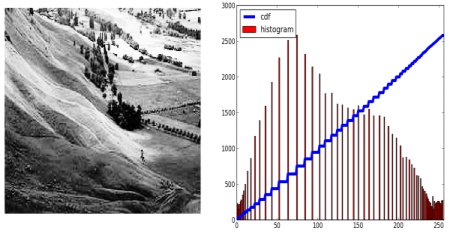
cdf\_m = (cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())

cdf = np.ma.filled(cdf\_m,0).astype('uint8')

ตอนนี้เรามีตารางการค้นหาที่ให้ข้อมูลเกี่ยวกับค่าพิกเซลเอาท์พุทสำหรับทุกค่าพิกเซลอินพุท ดังนั้นเราจึงใช้การแปลงนี้

img2 = cdf[img]

ตอนนี้เราคำนวณฮิสโตแกรมและ cdf ก่อนหน้า (คุณทำ) และผลลัพธ์ที่ได้จะเป็นดังนี้:



อีกประการหนึ่งคือแม้ว่าภาพจะเป็นภาพที่มืดกว่า (แทนที่จะเป็นภาพสว่างกว่าที่เราใช้อยู่) หลังจากการทำให้เท่าเทียมกันเราจะได้ภาพที่เหมือนกันเกือบเท่าที่เราได้รับ ด้วยเหตุนี้จึงใช้เป็นเครื่องมืออ้างอิงเพื่อสร้างภาพทั้งหมดที่มีสภาพแสงเหมือนกัน นี้เป็นประโยชน์ในหลาย ๆ กรณี ตัวอย่างเช่นในการจดจำใบหน้าก่อนที่จะทำการฝึกอบรมข้อมูลใบหน้ารูปภาพของใบหน้าจะมีการปรับ Histogram เพื่อให้ทุกฉากมีสภาพแสงเหมือนกัน

Histograms Equalization ใน OpenCV

OpenCV มีฟังก์ชั่นการทำเช่นนี้cv2.equalizeHist () การป้อนข้อมูลเป็นเพียงภาพสีเทาและภาพนิ่งเป็นภาพที่เท่าเทียมกันของ histogram

ด้านล่างนี้เป็นข้อมูลโค้ดง่ายๆที่แสดงถึงการใช้งานสำหรับภาพเดียวกันที่เราใช้:

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

equ = cv2.equalizeHist(img)

res = np.hstack((img,equ)) *#stacking images side-by-side*

cv2.imwrite('res.png',res)

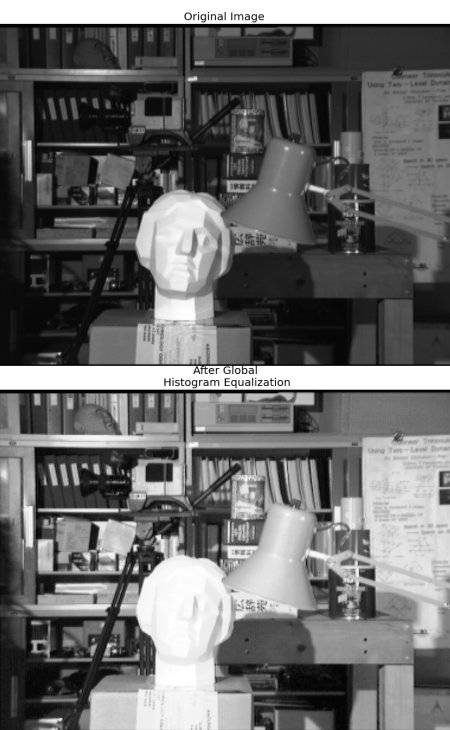


ดังนั้นตอนนี้คุณสามารถใช้ภาพที่แตกต่างกันกับสภาพแสงที่แตกต่างกันให้เท่ากันและตรวจสอบผล

การปรับ Histogram ให้ดีขึ้นเมื่อฮิสโตแกรมของภาพถูก จำกัด อยู่ในพื้นที่ใดภูมิภาคหนึ่ง จะไม่ทำงานได้ดีในสถานที่ที่มีการเปลี่ยนแปลงความเข้มสูงที่ histogram ครอบคลุมพื้นที่ขนาดใหญ่เช่นมีพิกเซลสว่างและมืดปรากฏอยู่ โปรดตรวจสอบลิงก์ SOF ในแหล่งข้อมูลเพิ่มเติม

CLAHE (ปรับความสมดุลของฮิสโตแกรม Contrast จำกัด )

การปรับค่าฮิสโตแกรมแรกที่เราเพิ่งเห็นถือว่าเป็นความคมชัดของภาพทั่วโลก ในหลาย ๆ กรณีไม่ใช่ความคิดที่ดี ตัวอย่างเช่นด้านล่างภาพจะแสดงภาพอินพุทและผลที่ได้หลังจากการปรับค่าฮิสโตแกรมทั่วโลก



ความแตกต่างของพื้นหลังดีขึ้นหลังจากการปรับสมดุลของฮิสโตแกรม แต่เปรียบเทียบรูปปั้นของรูปทั้งสองรูป เราสูญเสียมากที่สุดของข้อมูลที่มีเนื่องจากความสว่างเกิน เนื่องจากฮิสโตแกรมนั้นไม่ได้อยู่ในขอบเขตเฉพาะที่เราเห็นในกรณีที่ก่อนหน้านี้ (พยายามที่จะพล็อตฮิสโตแกรมของภาพอินพุทคุณจะได้สัญชาตญาณมากขึ้น)

ดังนั้นเพื่อแก้ปัญหานี้ทำให้เท่าเทียมกัน histogram ปรับตัวถูกนำมาใช้ ในภาพนี้ภาพจะถูกแบ่งออกเป็นบล็อคเล็ก ๆ ที่เรียกว่า "tiles" (tileSize เป็น 8x8 โดยค่าเริ่มต้นใน OpenCV) จากนั้นแต่ละบล็อคเหล่านี้จะมีการปรับ Histogram ตามปกติ ดังนั้นในพื้นที่เล็ก histogram จะ จำกัด ขอบเขตเล็ก ๆ (เว้นแต่มีเสียงรบกวน) หากเสียงดังอยู่ในนั้นจะขยายสัญญาณ เพื่อหลีกเลี่ยงปัญหานี้จะมีการ จำกัด ความคมชัด หากแท่งสตางค์ฮิสโตแกรมอยู่เหนือขีด จำกัด ความคมชัดที่ระบุ (โดยค่าเริ่มต้นที่ 40 ใน OpenCV) พิกเซลเหล่านี้จะถูกตัดและกระจายไปยังถังขยะอื่น ๆ อย่างสม่ำเสมอก่อนที่จะใช้การปรับค่า Histogram หลังจากการทำให้เท่าเทียมกันเพื่อลบสิ่งประดิษฐ์ในเส้นขอบของกระเบื้องจะมีการใช้การสอดแทรกแบบสองมิติ

ข้อมูลโค้ดด้านล่างแสดงวิธีใช้ CLAHE ใน OpenCV:

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

**import** **cv2**

img = cv2.imread('tsukuba\_l.png',0)

*# create a CLAHE object (Arguments are optional).*

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

cl1 = clahe.apply(img)

cv2.imwrite('clahe\_2.jpg',cl1)

ดูผลด้านล่างและเปรียบเทียบกับผลลัพธ์ข้างต้นโดยเฉพาะบริเวณรูปปั้น:

