Histograms - 1: Find, Plot, Analyze !!!

เป้าหมาย

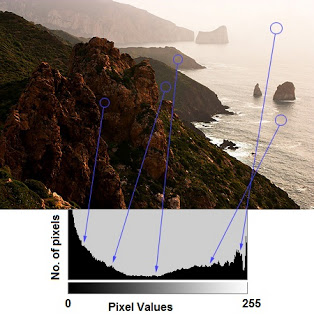
เรียนรู้

* ค้นหาฮิสโตแกรมโดยใช้ฟังก์ชัน OpenCV และ Numpy
* พล็อต histograms โดยใช้ฟังก์ชัน OpenCV และ Matplotlib
* คุณจะเห็นฟังก์ชันเหล่านี้: **cv2.calcHist ()** , **np.histogram ()**เป็นต้น

ทฤษฎี

ดังนั้น histogram คืออะไร? คุณสามารถพิจารณาฮิสโตแกรมเป็นกราฟหรือพล็อตซึ่งจะให้แนวคิดโดยรวมเกี่ยวกับการกระจายความเข้มของภาพ เป็นพล็อตที่มีค่าพิกเซล (ตั้งแต่ 0 ถึง 255 ไม่เสมอไป) ในแกน X และจำนวนพิกเซลที่เหมือนกันในภาพบนแกน Y

มันเป็นเพียงอีกวิธีหนึ่งในการทำความเข้าใจภาพ เมื่อมองที่ฮิสโตแกรมของภาพคุณจะได้สัญชาตญาณเกี่ยวกับความคมชัดความสว่างการกระจายความเข้มของภาพนั้น เกือบทุกเครื่องมือในการประมวลผลภาพในปัจจุบันมีคุณลักษณะเกี่ยวกับฮิสโตแกรม ด้านล่างเป็นภาพจาก[Cambridge ในเว็บไซต์ Color](http://www.cambridgeincolour.com/tutorials/histograms1.htm)และขอแนะนำให้คุณเยี่ยมชมเว็บไซต์เพื่อดูรายละเอียดเพิ่มเติม



คุณสามารถดูภาพและฮิสโตแกรมได้ (โปรดจำไว้ว่าฮิสโตแกรมนี้ถูกวาดขึ้นสำหรับภาพสีเทาไม่ใช่ภาพสี) บริเวณด้านซ้ายของฮิสโตแกรมแสดงจำนวนพิกเซลที่มืดลงในภาพและบริเวณด้านขวาจะแสดงจำนวนพิกเซลสว่างกว่า จากฮิสโตแกรมคุณสามารถมองเห็นพื้นที่มืดได้มากกว่าพื้นที่ที่สว่างกว่าและจำนวนแสงกึ่งกลาง (ค่าพิกเซลในช่วงกลางเดือนกล่าวประมาณ 127) มีน้อยมาก

ค้นหา Histogram

ตอนนี้เรามีแนวคิดเกี่ยวกับฮิสโตแกรมเราสามารถดูวิธีหาสิ่งนี้ได้อย่างไร ทั้ง OpenCV และ Numpy มาพร้อมกับฟังก์ชั่นที่สร้างขึ้นสำหรับสิ่งนี้ ก่อนที่จะใช้ฟังก์ชั่นเหล่านี้เราจำเป็นต้องเข้าใจคำศัพท์บางอย่างเกี่ยวกับฮิสโตแกรม

BINS : ฮิสโตแกรมข้างต้นแสดงจำนวนพิกเซลสำหรับทุกค่าพิกเซลคือ 0 ถึง 255 เช่นคุณต้องมีค่า 256 ค่าเพื่อแสดงฮิสโตแกรมดังกล่าว แต่พิจารณาสิ่งที่ถ้าคุณไม่จำเป็นต้องค้นหาจำนวนพิกเซลสำหรับค่าพิกเซลทั้งหมดแยกกัน แต่จำนวนพิกเซลในช่วงของค่าพิกเซล? พูดเช่นคุณต้องการหาจำนวนพิกเซลที่อยู่ระหว่าง 0 ถึง 15 แล้ว 16 ถึง 31, ... , 240 ถึง 255 คุณจะต้องมีเพียง 16 ค่าเพื่อแสดงฮิสโตแกรม และนั่นคือสิ่งที่จะปรากฏในตัวอย่างที่กำหนดใน[OpenCV สอนใน](http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html#histogram-calculation)histograms

ดังนั้นสิ่งที่คุณทำคือเพียงแค่แบ่งฮิสโทแกรมทั้งหมดเป็น 16 ส่วนย่อยและค่าของแต่ละส่วนย่อยคือผลรวมของจำนวนพิกเซลทั้งหมดในนั้น แต่ละส่วนย่อยนี้เรียกว่า "BIN" ในกรณีแรกจำนวนถังขยะที่ 256 (หนึ่งสำหรับแต่ละพิกเซล) ในขณะที่ในกรณีที่สองเป็นเพียง 16 BINS แสดงโดยhistSizeระยะใน OpenCV เอกสาร

DIMS : เป็นจำนวนพารามิเตอร์ที่เราเก็บรวบรวมข้อมูล ในกรณีนี้เราจะเก็บรวบรวมข้อมูลเกี่ยวกับค่าความเข้มเพียงอย่างเดียวเท่านั้น นี่คือ 1.

RANGE : เป็นช่วงของค่าความเข้มที่คุณต้องการวัด โดยปกติจะเป็น [0,256] เช่นค่าความเข้มทั้งหมด

1. การคำนวณฮิสโตแกรมใน OpenCV

ตอนนี้เราใช้cv2.calcHist ()เพื่อหาฮิสโตแกรม ลองทำความคุ้นเคยกับฟังก์ชันและพารามิเตอร์:

cv2.calcHist (ภาพช่องมาสก์ histSize ช่วง [, hist [, accumulate]])

1. ภาพ: เป็นภาพต้นฉบับของ uint8 หรือ float32 ควรให้เป็นรูปสี่เหลี่ยมจัตุรัสเช่น "[img]"
2. ช่องสัญญาณ: นอกจากนี้ยังมีให้ในวงเล็บเหลี่ยม เป็นดัชนีของช่องที่เราคำนวณฮิสโทแกรม ตัวอย่างเช่นถ้าข้อมูลเข้าเป็นภาพสีเทาค่าของมันคือ [0] สำหรับภาพสีคุณสามารถผ่าน [0], [1] หรือ [2] เพื่อคำนวณฮิสโตแกรมของช่องสีฟ้าสีเขียวหรือสีแดงตามลําดับ
3. หน้ากาก: รูปหน้ากาก หากต้องการค้นหาฮิสโตแกรมของภาพแบบเต็มรูปแบบจะแสดงเป็น "ไม่มี" แต่ถ้าคุณต้องการหาฮิสโตแกรมของพื้นที่ใดภาพหนึ่งคุณต้องสร้างภาพหน้ากากสำหรับสิ่งนั้นและให้เป็นหน้ากาก (ฉันจะแสดงตัวอย่างในภายหลัง)
4. histSize: นี่หมายถึงจำนวน BIN ของเรา จำเป็นต้องได้รับในวงเล็บเหลี่ยม สำหรับขนาดเต็มเราผ่าน [256]
5. ช่วง: นี่คือ RANGE ของเรา โดยปกติจะเป็น [0,256]

ลองเริ่มต้นด้วยภาพตัวอย่าง เพียงแค่โหลดภาพในโหมดสีเทาและค้นหาฮิสโตแกรมเต็มรูปแบบ

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

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

hist เป็นอาร์เรย์ 256x1 แต่ละค่าจะตรงกับจำนวนพิกเซลในภาพที่มีค่าพิกเซลที่ตรงกัน

2. การคำนวณฮิสโตแกรมใน Numpy

Numpy ยังให้คุณฟังก์ชั่น, np.histogram () ดังนั้นแทน calcHist () ฟังก์ชันคุณสามารถลองด้านล่างบรรทัด:

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

hist เป็นเช่นเดียวกับที่เราคำนวณมาก่อน แต่ถังขยะจะมี 257 องค์ประกอบเพราะ Numpy คำนวณถังเป็น 0-0.99, 1-1.99, 2-2.99 เป็นต้นดังนั้นช่วงสุดท้ายจะเป็น 255-255.99 เพื่อแสดงว่าพวกเขายังเพิ่ม 256 ที่ท้ายถัง แต่เราไม่จำเป็นต้อง 256 ที่ไม่เกิน 255 ก็เพียงพอแล้ว

ดูสิ่งนี้ด้วย : Numpy มีฟังก์ชันอื่นnp.bincount ()ซึ่งเร็วกว่า (ประมาณ 10X) np.histogram () ดังนั้นสำหรับฮิสโตแกรมแบบหนึ่งมิติคุณสามารถลองทำแบบนั้นได้ดียิ่งขึ้น อย่าลืมตั้งminlength = 256ใน np.bincount ตัวอย่างเช่นhist = np.bincount (img.ravel (), minlength = 256)

บันทึก : ฟังก์ชัน OpenCV จะเร็วกว่า (รอบ 40X) กว่า np.histogram () ดังนั้นติดกับ OpenCV function

ตอนนี้เราควรวางแผน histograms แต่อย่างไร?

Plotting Histograms

มีสองวิธีสำหรับเรื่องนี้

1. Short Way: ใช้ฟังก์ชั่นการวางแผน Matplotlib
2. Long Way: ใช้ฟังก์ชันการวาดภาพ OpenCV

1. ใช้ Matplotlib

Matplotlib มาพร้อมกับ histogram plotting function: matplotlib.pyplot.hist ()

พบฮิสโทแกรมโดยตรงและวางแผน คุณไม่จำเป็นต้องใช้ฟังก์ชัน calcHist () หรือ np.histogram () เพื่อค้นหาฮิสโตแกรม ดูรหัสด้านล่าง:

**import** **cv2**

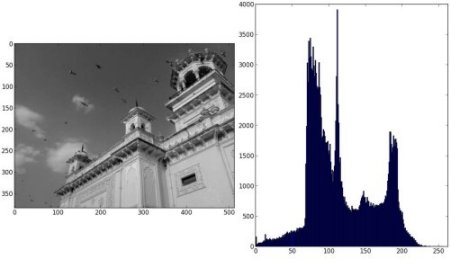
**import** **numpy** **as** **np**

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

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

plt.hist(img.ravel(),256,[0,256]); plt.show()

คุณจะได้รับพล็อตด้านล่าง:



หรือคุณสามารถใช้พล็อตปกติของ matplotlib ซึ่งจะเป็นประโยชน์สำหรับพล็อต BGR สำหรับข้อมูลนี้คุณต้องหาข้อมูล histogram ก่อน ลองโค้ดด้านล่าง:

**import** **cv2**

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

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

img = cv2.imread('home.jpg')

color = ('b','g','r')

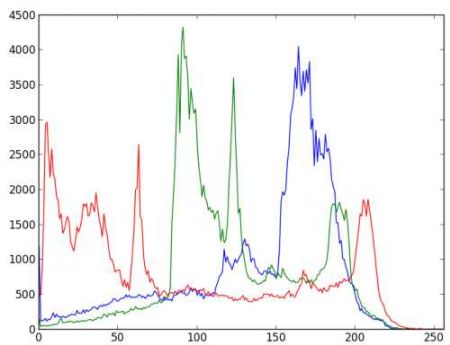
**for** i,col **in** enumerate(color):

histr = cv2.calcHist([img],[i],None,[256],[0,256])

plt.plot(histr,color = col)

plt.xlim([0,256])

plt.show()



คุณสามารถหักจากกราฟข้างต้นได้ว่าสีน้ำเงินมีพื้นที่ที่มีมูลค่าสูงในภาพ (ชัดว่าควรเป็นเพราะท้องฟ้า)

2. การใช้ OpenCV

ดีที่นี่คุณปรับค่า histograms พร้อมกับค่า bin ให้มีลักษณะเหมือน x, y พิกัดเพื่อให้คุณสามารถวาดโดยใช้ cv2.line () หรือ cv2.polyline () เพื่อสร้างภาพเดียวกันข้างต้น นี้มีอยู่แล้วกับ OpenCV ตัวอย่าง Python2 อย่างเป็นทางการ [ตรวจสอบรหัส](https://github.com/Itseez/opencv/raw/master/samples/python2/hist.py)

การใช้ Mask

เราใช้ cv2.calcHist () เพื่อหาฮิสโตแกรมของภาพแบบเต็ม จะทำอย่างไรถ้าคุณต้องการค้นหาฮิสโตแกรมของบางภูมิภาคของรูปภาพ เพียงแค่สร้างภาพหน้ากากด้วยสีขาวในพื้นที่ที่คุณต้องการหาฮิสโทแกรมและสีดำเป็นอย่างอื่น จากนั้นจึงส่งผ่านมาเป็นหน้ากาก

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

*# create a mask*

mask = np.zeros(img.shape[:2], np.uint8)

mask[100:300, 100:400] = 255

masked\_img = cv2.bitwise\_and(img,img,mask = mask)

*# Calculate histogram with mask and without mask*

*# Check third argument for mask*

hist\_full = cv2.calcHist([img],[0],None,[256],[0,256])

hist\_mask = cv2.calcHist([img],[0],mask,[256],[0,256])

plt.subplot(221), plt.imshow(img, 'gray')

plt.subplot(222), plt.imshow(mask,'gray')

plt.subplot(223), plt.imshow(masked\_img, 'gray')

plt.subplot(224), plt.plot(hist\_full), plt.plot(hist\_mask)

plt.xlim([0,256])

plt.show()

ดูผลลัพธ์ ในพล็อตฮิสโตแกรมเส้นสีน้ำเงินจะแสดงฮิสโตแกรมของภาพทั้งหมดในขณะที่เส้นสีเขียวแสดงฮิสโตแกรมของพื้นที่ที่สวมหน้ากาก

