**Hough Line Transform**

เป้าหมาย

ในบทนี้,

* เราจะเข้าใจแนวคิดของ Hough Tranform
* เราจะดูวิธีใช้สายตรวจจับเส้นในภาพ
* เราจะเห็นฟังก์ชันต่อไปนี้: **cv2.HoughLines ()** , **cv2.HoughLinesP ()**

ทฤษฎี

Hough Transform เป็นเทคนิคยอดนิยมในการตรวจจับรูปร่างใด ๆ หากคุณสามารถแสดงรูปร่างนั้นในรูปแบบทางคณิตศาสตร์ สามารถตรวจจับรูปร่างได้แม้จะหักหรือบิดเบี้ยวเล็กน้อยก็ตาม เราจะเห็นวิธีการทำงานสำหรับบรรทัด

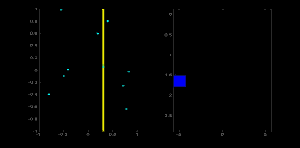
สายสามารถแสดงเป็นy = mx + cหรือในรูปแบบตัวแปรเป็น\ rho = x \ cos \ theta + y \ sin \ thetaที่\ โรเป็นระยะตั้งฉากจากต้นทางไปยังเส้นและ\ thetaเป็นมุมที่เกิดขึ้นจากเส้นตั้งฉากนี้และแกนนอนในวัดทวนเข็มนาฬิกา (ทิศทางที่แตกต่างกันเกี่ยวกับวิธีการที่คุณเป็นตัวแทนประสานงาน ระบบการแทนนี้ใช้ใน OpenCV) ตรวจสอบด้านล่างภาพ:

ดังนั้นถ้าเส้นผ่านด้านล่างต้นกำเนิดจะมีมุม rho บวกและมุมน้อยกว่า 180 ถ้าเป็นไปเหนือต้นกำเนิดแทนการใช้มุมมากกว่า 180 มุมจะถูกถ่ายน้อยกว่า 180 และ rho จะถูกลบ เส้นแนวตั้งจะมี 0 องศาและเส้นแนวนอนจะมี 90 องศา

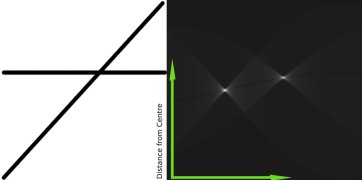
ตอนนี้ขอดูว่า Hough Transform ใช้งานได้ดีเพียงใด บรรทัดใด ๆ สามารถแสดงในสองคำนี้, (\ rho, \ theta). อันดับแรกจะสร้างอาร์เรย์ 2D หรือตัวสะสม (ถือค่าสองพารามิเตอร์) และจะตั้งค่าเป็น 0 ในตอนแรก Let แถวแสดงถึงและคอลัมน์ที่แสดงว่า\ โร \ thetaขนาดของอาร์เรย์ขึ้นอยู่กับความแม่นยำที่คุณต้องการ สมมติว่าคุณต้องการความถูกต้องของมุมเป็น 1 องศาคุณต้องมี 180 คอลัมน์ สำหรับ\ โรระยะทางสูงสุดที่เป็นไปได้คือความยาวเส้นทแยงมุมของภาพ ดังนั้นการคำนวณความถูกต้องของพิกเซลหนึ่งจำนวนแถวจึงเป็นความยาวเส้นทแยงมุมของภาพ

พิจารณาภาพขนาด 100x100 ที่มีเส้นแนวนอนตรงกลาง ใช้จุดแรกของเส้น คุณรู้จักค่าของ (x, y) แล้ว ตอนนี้ในสมการของบรรทัดใส่ค่า\ theta = 0,1,2, .... , 180และตรวจสอบที่\ โรคุณได้รับ สำหรับ(\ rho, \ theta)คู่ทุกครั้งคุณจะเพิ่มมูลค่าให้กับหนึ่งใน accumulator ของเราใน(\ rho, \ theta)เซลล์ที่สัมพันธ์กัน ดังนั้นตอนนี้ใน accumulator เซลล์ (50,90) = 1 พร้อมกับเซลล์อื่น ๆ

ตอนนี้ใช้จุดที่สองในบรรทัด ทำเช่นเดียวกับข้างต้น เพิ่มค่าในเซลล์ที่ตรงกับที่(\ rho, \ theta)คุณได้รับ เวลานี้เซลล์ (50,90) = 2 สิ่งที่คุณทำคือการออกเสียงลงคะแนน(\ rho, \ theta)ค่า คุณดำเนินการขั้นตอนนี้ต่อไปทุกจุดบนเส้น ในแต่ละจุดเซลล์ (50,90) จะเพิ่มขึ้นหรือโหวตขึ้นขณะที่เซลล์อื่น ๆ อาจได้รับการโหวตหรือไม่ก็ได้ ด้วยวิธีนี้ตอนท้ายเซลล์ (50,90) จะมีคะแนนสูงสุด ดังนั้นถ้าคุณค้นหา accumulator สำหรับคะแนนสูงสุดคุณจะได้รับค่า (50,90) ซึ่งระบุว่ามีเส้นในภาพนี้ที่ระยะ 50 จากต้นกำเนิดและที่มุม 90 องศา มันแสดงให้เห็นได้ดีในภาพเคลื่อนไหวด้านล่าง (ภาพมารยาท: [Amos Storkey](http://homepages.inf.ed.ac.uk/amos/hough.html) )



นี่เป็นวิธีที่การแปลง hough สำหรับการทำงานของสายงาน เป็นเรื่องง่ายและอาจใช้งานได้โดยใช้ Numpy ด้วยตัวคุณเอง ด้านล่างเป็นภาพที่แสดง accumulator จุดสว่างในบางตำแหน่งบ่งบอกว่าเป็นพารามิเตอร์ของเส้นที่เป็นไปได้ในภาพ (ภาพมารยาท: [Wikipedia](http://en.wikipedia.org/wiki/Hough_transform) )



Hough Tranform ใน OpenCV

ทุกอย่างที่อธิบายข้างต้นจะห่อหุ้มในการทำงาน OpenCV ที่cv2.HoughLines () เพียงส่งกลับอาร์เรย์ของ(\ rho, \ theta)ค่า \ โรวัดเป็นพิกเซลและ\ thetaวัดเป็นเรเดียน พารามิเตอร์แรกรูปภาพอินพุตควรเป็นรูปแบบไบนารีดังนั้นให้ใช้เกณฑ์หรือใช้การตรวจหาขอบแบบไม่ชำนาญก่อนที่จะหาการใช้การแปลง Hough พารามิเตอร์ที่สองและสามเป็น\ โรและ\ thetaความแม่นยำตามลำดับ อาร์กิวเมนต์ที่สี่เป็นเกณฑ์ซึ่งหมายถึงคะแนนขั้นต่ำที่ควรจะได้รับเพื่อให้ได้รับการพิจารณาเป็นบรรทัด โปรดจำไว้ว่าจำนวนคะแนนเสียงขึ้นอยู่กับจำนวนคะแนนในบรรทัด ดังนั้นจึงแสดงถึงความยาวขั้นต่ำของเส้นที่ควรตรวจพบ

**import** **cv2**

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

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

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

edges = cv2.Canny(gray,50,150,apertureSize = 3)

lines = cv2.HoughLines(edges,1,np.pi/180,200)

**for** rho,theta **in** lines[0]:

a = np.cos(theta)

b = np.sin(theta)

x0 = a\*rho

y0 = b\*rho

x1 = int(x0 + 1000\*(-b))

y1 = int(y0 + 1000\*(a))

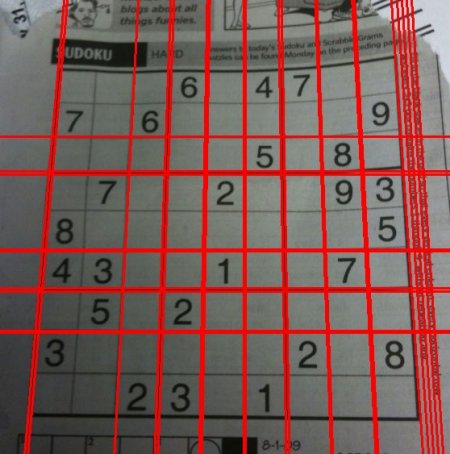
x2 = int(x0 - 1000\*(-b))

y2 = int(y0 - 1000\*(a))

cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

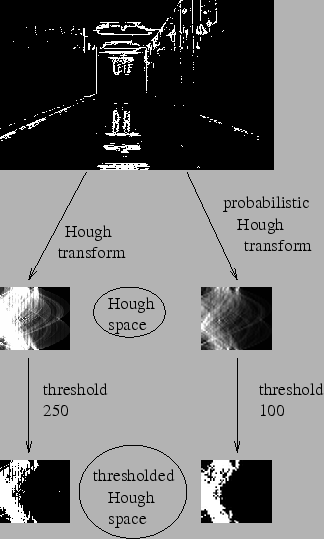
cv2.imwrite('houghlines3.jpg',img)

ตรวจสอบผลลัพธ์ด้านล่าง:



Probabilistic Hough Transform

ในการแปลงค่า hough คุณจะเห็นได้ว่าแม้แต่บรรทัดเดียวกับอาร์กิวเมนต์สองตัวก็ต้องใช้การคำนวณเป็นจำนวนมาก Proofabilistic Hough Transform คือการเพิ่มประสิทธิภาพของ Hough Transform ที่เราเห็น ไม่คำนึงถึงประเด็นทั้งหมดแทนที่จะใช้เฉพาะเซตย่อยแบบสุ่มของจุดและนั่นก็เพียงพอสำหรับการตรวจจับเส้น เพียงแค่เราต้องลดเกณฑ์ ดูภาพด้านล่างซึ่งเปรียบเทียบ Hough Transform และ Probabilistic Hough Transform ในพื้นที่ที่มีอยู่ (ภาพมารยาท: [หน้าแรกของ Franck Bettinger](http://phdfb1.free.fr/robot/mscthesis/node14.html)



การใช้ OpenCV จะขึ้นอยู่กับการตรวจจับที่ชัดเจนของเส้นโดยใช้การแปลง Hough Probabilistic Hough โดย Matas, J. และ Galambos, C. และ Kittler, JV ฟังก์ชั่นที่ใช้เป็น**cv2.HoughLinesP**() มีข้อโต้แย้งใหม่สองข้อ

* **minLineLength** - ความยาวขั้นต่ำของบรรทัด กลุ่มสายที่สั้นกว่านี้ถูกปฏิเสธ
* **maxLineGap** - ช่องว่างสูงสุดที่อนุญาตระหว่างกลุ่มเส้นเพื่อให้เป็นบรรทัดเดียว

สิ่งที่ดีที่สุดคือการส่งกลับค่าปลายทางสองบรรทัดโดยตรง ในกรณีก่อนหน้านี้คุณมีเฉพาะพารามิเตอร์ของเส้นและคุณต้องค้นหาทุกจุด ที่นี่ทุกอย่างเป็นเรื่องง่ายและง่าย

**import** **cv2**

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

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

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

edges = cv2.Canny(gray,50,150,apertureSize = 3)

minLineLength = 100

maxLineGap = 10

lines = cv2.HoughLinesP(edges,1,np.pi/180,100,minLineLength,maxLineGap)

**for** x1,y1,x2,y2 **in** lines[0]:

cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)

cv2.imwrite('houghlines5.jpg',img)

