**Image Gradients**

เป้าหมาย

ในบทนี้เราจะเรียนรู้เพื่อ:

* ค้นหาการไล่ระดับสีของภาพขอบ ฯลฯ
* เราจะเห็นฟังก์ชันต่อไปนี้: **cv2.Sobel ()** , **cv2.Scharr ()** , **cv2.Laplacian ()**ฯลฯ

ทฤษฎี

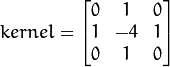
OpenCV มีตัวกรองการไล่ระดับสีสามแบบหรือตัวกรอง High-pass, Sobel, Scharr และ Laplacian เราจะเห็นแต่ละคน

1. Sobel และ Scharr Derivatives

โอเปอเรเตอร์โชเบลคือการปรับให้เรียบแบบ Gausssian และการแยกแยะความแตกต่างดังนั้นจึงมีความทนทานต่อเสียงรบกวนมากขึ้น คุณสามารถกำหนดทิศทางของอนุพันธ์ที่จะต้องดำเนินการแนวตั้งหรือแนวนอน (โดยอาร์กิวเมนต์ yorder และ xorder ตามลำดับ) นอกจากนี้คุณยังสามารถระบุขนาดของเมล็ดโดยอาร์กิวเมนต์ ksize หาก ksize = -1 จะใช้ตัวกรอง Scharr 3x3 ซึ่งให้ผลลัพธ์ที่ดีกว่าตัวกรอง Sobel 3x3 โปรดดูเอกสารสำหรับเมล็ดที่ใช้

2. อนุพันธ์ Laplacian

จะคำนวณ Laplacian ของภาพที่กำหนดโดยความสัมพันธ์\ Delta src = \ frac {\ partial ^ 2 {src}} {\ partial x ^ 2} + \ frac {\ partial ^ 2 {src}} {\ partial y ^ 2}ซึ่งแต่ละ derivative จะพบโดยใช้อนุพันธ์ของ Sobel ถ้า ksize = 1แล้วเคอร์เนลต่อไปนี้ใช้สำหรับการกรอง:



CODE

ด้านล่างโค้ดแสดงตัวดำเนินการทั้งหมดในแผนภาพเดียว เมล็ดทั้งหมดมีขนาด 5x5 ความลึกของภาพที่ส่งออกจะถูกส่งผ่าน -1 เพื่อให้ได้ผลลัพธ์ในรูปแบบ np.uint8

**import** **cv2**

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

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

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

laplacian = cv2.Laplacian(img,cv2.CV\_64F)

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

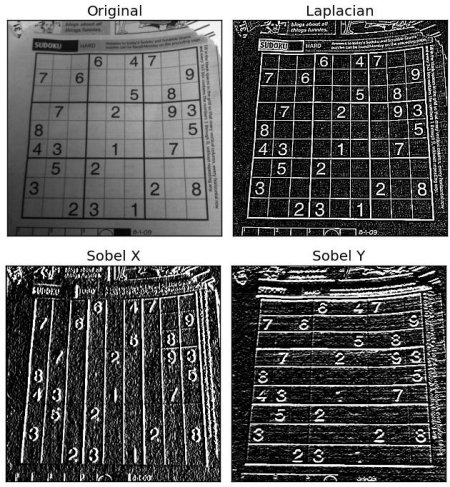
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()



หนึ่งเรื่องสำคัญ!

ในตัวอย่างสุดท้ายประเภทข้อมูลขาออกคือ cv2.CV\_8U หรือ np.uint8 แต่มีปัญหาเล็กน้อยคือ (มีค่าเป็นบวก) ในขณะที่การเปลี่ยนสีขาว - ดำเป็นความลาดชันเชิงลบ (มีค่าเป็นลบ) ดังนั้นเมื่อคุณแปลงข้อมูลเป็น np.uint8 ความชันเชิงลบทั้งหมดจะเป็นศูนย์ ในคำง่ายๆคุณพลาดขอบที่

ถ้าคุณต้องการตรวจหาทั้งสองขอบตัวเลือกที่ดีกว่าคือเก็บประเภทข้อมูลขาออกให้อยู่ในรูปแบบที่สูงกว่าเช่น cv2.CV\_16S, cv2.CV\_64F ฯลฯ ให้ใช้ค่าสัมบูรณ์แล้วแปลงกลับเป็น cv2.CV\_8U โค้ดด้านล่างแสดงขั้นตอนนี้สำหรับตัวกรอง Sobel แนวนอนและผลต่างกัน

**import** **cv2**

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

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

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

*# Output dtype = cv2.CV\_8U*

sobelx8u = cv2.Sobel(img,cv2.CV\_8U,1,0,ksize=5)

*# Output dtype = cv2.CV\_64F. Then take its absolute and convert to cv2.CV\_8U*

sobelx64f = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

abs\_sobel64f = np.absolute(sobelx64f)

sobel\_8u = np.uint8(abs\_sobel64f)

plt.subplot(1,3,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(1,3,2),plt.imshow(sobelx8u,cmap = 'gray')

plt.title('Sobel CV\_8U'), plt.xticks([]), plt.yticks([])

plt.subplot(1,3,3),plt.imshow(sobel\_8u,cmap = 'gray')

plt.title('Sobel abs(CV\_64F)'), plt.xticks([]), plt.yticks([])

plt.show()

