**การแปลงทางสัณฐานวิทยา**

เป้าหมาย

ในบทนี้,

* เราจะเรียนรู้การดำเนินการทางสัณฐานวิทยาที่แตกต่างกันเช่นการกร่อน, การขยาย, การเปิด, การปิด ฯลฯ
* เราจะเห็นฟังก์ชันต่างๆเช่น**cv2.erode ()** , **cv2.dilate ()** , **cv2.morphologyEx ()**เป็นต้น

ทฤษฎี

การแปลงทางสัณฐานวิทยาเป็นการดำเนินการง่ายๆตามรูปร่าง โดยปกติจะใช้กับภาพไบนารี จำเป็นต้องใช้สองปัจจัยหนึ่งคือภาพต้นฉบับของเราส่วนที่สองเรียกว่าstructuring elementหรือkernelซึ่งกำหนดลักษณะการทำงาน ตัวดำเนินการโครงสร้างพื้นฐานสองตัวคือการกร่อนและการกรีด จากนั้นรูปแบบที่หลากหลายเช่น Opening, Closing, Gradient ฯลฯ ก็เข้ามาเล่น เราจะเห็นพวกเขาทีละภาพด้วยความช่วยเหลือของภาพต่อไปนี้:



1. Erosion

ความคิดพื้นฐานของการพังทลายเป็นเพียงการพังทลายของดินเท่านั้นทำให้พื้นผิวของวัตถุเบื้องหน้าพังทลายลง ดังนั้นสิ่งที่ไม่? เคอร์เนลสไลด์ผ่านภาพ (เช่นเดียวกับ 2D convolution) พิกเซลในภาพต้นฉบับ (1 หรือ 0) จะถือเป็น 1 หากว่าพิกเซลทั้งหมดภายใต้เคอร์เนลอยู่ที่ 1 มิฉะนั้นจะถูกกัดเซาะ (ทำเป็นศูนย์)

ดังนั้นสิ่งที่เกิดขึ้นคือพิกเซลทั้งหมดที่อยู่ใกล้กับขอบเขตจะถูกละทิ้งขึ้นอยู่กับขนาดของเคอร์เนล ดังนั้นความหนาหรือขนาดของวัตถุเบื้องหน้าจะลดลงหรือสีขาวจะลดลงในภาพ เป็นประโยชน์สำหรับการลบเสียงสีขาวขนาดเล็ก (ตามที่เราได้เห็นในบท colorspace) ถอดชิ้นส่วนที่เชื่อมต่อสองชิ้นออกเป็นต้น

นี่เป็นตัวอย่างฉันจะใช้เมล็ด 5x5 กับเต็ม ลองดูวิธีการทำงาน:

**import** **cv2**

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

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

kernel = np.ones((5,5),np.uint8)

erosion = cv2.erode(img,kernel,iterations = 1)



2. Dilation

มันเป็นเพียงตรงข้ามของการกัดเซาะ ที่นี่องค์ประกอบพิกเซลคือ '1' ถ้าอย่างน้อยหนึ่งพิกเซลภายใต้เคอร์เนลคือ '1' ดังนั้นจึงเพิ่มสีขาวในภาพหรือขนาดของวัตถุที่มีการเพิ่มขึ้น โดยปกติในกรณีเช่นการกำจัดเสียงรบกวนการพังทลายตามมาด้วยการขยายตัว เนื่องจากการกัดเซาะจะช่วยขจัดเสียงสีขาว แต่ยังทำให้วัตถุของเราหดตัว ดังนั้นเราจึงขยายมัน เนื่องจากเสียงดังหายไปพวกเขาจะไม่กลับมา แต่พื้นที่ของเราจะเพิ่มขึ้น นอกจากนี้ยังมีประโยชน์ในการเชื่อมต่อชิ้นส่วนที่หักของวัตถุ

dilation = cv2.dilate(img,kernel,iterations = 1)



3. Opening

เปิดเป็นเพียงชื่อของผู้อื่นการกัดเซาะตามด้วยการขยายจะเป็นประโยชน์ในการขจัดเสียงดังที่เราได้อธิบายไว้ข้างต้น ที่นี่เราใช้ฟังก์ชัน**cv2.morphologyEx ()**

opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)



4. Closing

ปิดเป็นสิ่งที่ตรงกันข้ามของการเปิดขยายตามด้วยการกัดเซาะมีประโยชน์ในการปิดรูเล็ก ๆ ภายในวัตถุเบื้องหน้าหรือจุดสีดำเล็ก ๆ บนวัตถุ

closing = cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)



5. Morphological Gradient

มันเป็นความแตกต่างระหว่างการขยายตัวและการกัดกร่อนของภาพผลลัพธ์จะมีลักษณะเหมือนเค้าร่างของวัตถุ

gradient = cv2.morphologyEx(img, cv2.MORPH\_GRADIENT, kernel)



6. Top Hat

มันเป็นความแตกต่างระหว่างภาพเข้าและการเปิดภาพ ตัวอย่างด้านล่างนี้ใช้สำหรับเคอร์เนล 9x9

tophat = cv2.morphologyEx(img, cv2.MORPH\_TOPHAT, kernel)



Black Hat

มันเป็นความแตกต่างระหว่างการปิดของภาพเข้าและภาพเข้า

blackhat = cv2.morphologyEx(img, cv2.MORPH\_BLACKHAT, kernel)



Structuring Element

เราได้สร้างองค์ประกอบโครงสร้างไว้ในตัวอย่างก่อนหน้าด้วยความช่วยเหลือของ Numpy เป็นรูปสี่เหลี่ยมผืนผ้า แต่ในบางกรณีคุณอาจต้องใช้รูปไข่ / รูปวงกลม ดังนั้นเพื่อวัตถุประสงค์นี้ OpenCV มีฟังก์ชั่นcv2.getStructuringElement () คุณเพียงแค่ผ่านรูปร่างและขนาดของเคอร์เนลคุณจะได้เคอร์เนลที่ต้องการ

*# Rectangular Kernel*

>>> cv2.getStructuringElement(cv2.MORPH\_RECT,(5,5))

array([[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1]], dtype=uint8)

*# Elliptical Kernel*

>>> cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

array([[0, 0, 1, 0, 0],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[0, 0, 1, 0, 0]], dtype=uint8)

*# Cross-shaped Kernel*

>>> cv2.getStructuringElement(cv2.MORPH\_CROSS,(5,5))

array([[0, 0, 1, 0, 0],

[0, 0, 1, 0, 0],

[1, 1, 1, 1, 1],

[0, 0, 1, 0, 0],

[0, 0, 1, 0, 0]], dtype=uint8)