**การแปลงรูปทรงเรขาคณิตของรูปภาพ**

เป้าหมาย

* เรียนรู้การเปลี่ยนแปลงทางเรขาคณิตที่แตกต่างกันไปกับภาพเช่นการแปลหมุนเวียนการเปลี่ยนแปลงเลียนแบบเป็นต้น
* คุณจะเห็นฟังก์ชันเหล่านี้: **cv2.getPerspectiveTransform**

Transformations

OpenCV มีฟังก์ชันการแปลงสองอย่างคือ cv2.warpAffine และ cv2.warpPerspective ซึ่งคุณสามารถแปลงได้ทุกรูปแบบ cv2.warpAffineใช้เมตริกซ์การแปลง 2x3 ในขณะที่ cv2.warpPerspective ใช้เมทริกซ์การเปลี่ยนแปลง 3x3 เป็นอินพุต

Scaling

การปรับขนาดเป็นเพียงการปรับขนาดของรูปภาพ OpenCV มาพร้อมกับฟังก์ชัน cv2.resize () เพื่อการนี้ คุณสามารถระบุขนาดของภาพได้ด้วยตัวเองหรือคุณสามารถระบุค่าปรับได้ ใช้วิธีการแก้ไขที่แตกต่างกัน วิธีการแก้ไขที่แนะนำคือ cv2.INTER\_AREA สำหรับการหดและ cv2.INTER\_CUBIC (ช้า) และcv2.INTER\_LINEARสำหรับการซูม โดยค่าเริ่มต้นการแก้ไขจะใช้เป็น cv2.INTER\_LINEAR สำหรับการปรับขนาดทั้งหมด คุณสามารถปรับขนาดภาพนำเข้าหนึ่งในสองวิธีดังต่อไปนี้:

**import** **cv2**

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

img = cv2.imread('messi5.jpg')

res = cv2.resize(img,None,fx=2, fy=2, interpolation = cv2.INTER\_CUBIC)

*#OR*

height, width = img.shape[:2]

res = cv2.resize(img,(2\*width, 2\*height), interpolation = cv2.INTER\_CUBIC)

Translation

การแปลคือการขยับตำแหน่งของวัตถุ ถ้าคุณทราบทิศทาง shift (x, y) ให้เป็นดังนี้(t_x, t_y)คุณสามารถสร้างเมตริกซ์การแปลงได้\ textbf {M}ดังนี้:

M = \ begin {bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \ end {bmatrix}

คุณสามารถใช้ทำให้มันกลายเป็นอาร์เรย์ Numpy ประเภทnp.float32และผ่านมันเข้าไปใน**cv2.warpAffine ()**ฟังก์ชั่น ดูตัวอย่างด้านล่างสำหรับการเปลี่ยน (100,50):

**import** **cv2**

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

img = cv2.imread('messi5.jpg',0)

rows,cols = img.shape

M = np.float32([[1,0,100],[0,1,50]])

dst = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow('img',dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

**การเตือน :** อาร์กิวเมนต์ที่สามของ **cv2.warpAffine ()** ฟังก์ชั่นคือขนาดของภาพออกซึ่งควรจะอยู่ในรูปแบบของ(ความกว้างความสูง)จำความกว้าง = จำนวนคอลัมน์และความสูง = จำนวนแถว

ดูผลลัพธ์ด้านล่าง:



หมุน

การหมุนภาพของมุมหนึ่ง\ thetaทำได้โดยเมทริกซ์การเปลี่ยนแปลงของแบบฟอร์ม

M = \ begin {bmatrix} cos \ theta & -sin \ theta \\ sin \ theta & cos \ theta \ end {bmatrix}

แต่ OpenCV ให้การหมุนที่ปรับขนาดได้โดยปรับศูนย์กลางของการหมุนเพื่อให้คุณสามารถหมุนได้ทุกที่ที่คุณต้องการ เมทริกซ์การเปลี่ยนแปลงที่เปลี่ยนแปลงได้มาจาก

\ begin {bmatrix} \ alpha & \ beta & (1- \ alpha) \ cdot center.x - \ beta \ cdot center.y \\ - \ beta & \ alpha & \ beta \ cdot center.x + (1- \ alpha) \ cdot center.y \ end {bmatrix}

ที่อยู่:

\ begin {array} {l} \ alpha = scale \ cdot \ cos \ theta, \\ \ beta = scale \ cdot \ sin \ theta \ end {array}

เพื่อหาเมทริกซ์การเปลี่ยนแปลงนี้ OpenCV ให้ฟังก์ชั่นcv2.getRotationMatrix2Dตรวจสอบตัวอย่างด้านล่างที่หมุนภาพโดย 90 องศาโดยคำนึงถึงจุดศูนย์กลางโดยไม่มีการปรับขนาดใด ๆ

img = cv2.imread('messi5.jpg',0)

rows,cols = img.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)

dst = cv2.warpAffine(img,M,(cols,rows))

See the result:



Affine Transformation

ในการแปลง affine เส้นคู่ขนานทั้งหมดในภาพต้นฉบับจะยังคงขนานในภาพเอาต์พุต ในการหาเมทริกซ์การแปลงเราต้องการคะแนนจากภาพอินพุทและตำแหน่งที่สัมพันธ์กันในภาพเอาต์พุต 3 จุด จากนั้น**cv2.getAffineTransform** จะสร้างเมทริกซ์ 2x3 ซึ่งจะถูกส่งผ่านไป cv2.warpAffine

ตรวจสอบตัวอย่างด้านล่างและดูที่จุดที่ฉันเลือก (ซึ่งมีสีเขียว):

img = cv2.imread('drawing.png')

rows,cols,ch = img.shape

pts1 = np.float32([[50,50],[200,50],[50,200]])

pts2 = np.float32([[10,100],[200,50],[100,250]])

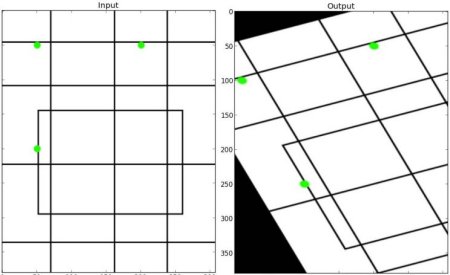
M = cv2.getAffineTransform(pts1,pts2)

dst = cv2.warpAffine(img,M,(cols,rows))

plt.subplot(121),plt.imshow(img),plt.title('Input')

plt.subplot(122),plt.imshow(dst),plt.title('Output')

plt.show()



การเปลี่ยนมุมมอง

สำหรับการแปลงมุมมองคุณต้องมีเมตริกซ์การแปลง 3x3 เส้นตรงจะยังคงตรงแม้หลังจากการเปลี่ยนแปลง เพื่อหาเมตริกซ์การแปลงนี้คุณต้องมี 4 จุดในภาพอินพุทและจุดที่ตรงกันบนภาพเอาต์พุต ใน 4 ประเด็นนี้ 3 ข้อควรไม่เป็นเส้นตรง จากนั้นการเปลี่ยนแปลงเมทริกซ์สามารถพบได้โดยฟังก์ชั่นcv2.getPerspectiveTransformจากนั้นใช้**cv2.warpPerspective**กับเมทริกซ์การเปลี่ยนแปลง 3x3

ดูรหัสด้านล่าง:

img = cv2.imread('sudokusmall.png')

rows,cols,ch = img.shape

pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])

pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])

M = cv2.getPerspectiveTransform(pts1,pts2)

dst = cv2.warpPerspective(img,M,(300,300))

plt.subplot(121),plt.imshow(img),plt.title('Input')

plt.subplot(122),plt.imshow(dst),plt.title('Output')

plt.show()

