**การจับคู่เทมเพลต**

เป้าหมาย

ในบทนี้คุณจะได้เรียนรู้

* เมื่อต้องการค้นหาวัตถุในภาพโดยใช้การจับคู่เทมเพลต
* คุณจะเห็นฟังก์ชันเหล่านี้: **cv2.matchTemplate ()** , **cv2.minMaxLoc ()**

ทฤษฎี

การจับคู่เทมเพลตเป็นวิธีการค้นหาและค้นหาตำแหน่งของภาพเทมเพลตในภาพขนาดใหญ่ OpenCV มาพร้อมกับฟังก์ชัน**cv2.matchTemplate ()**เพื่อการนี้ เพียงเลื่อนภาพเทมเพลตไปยังภาพที่ป้อน (เช่นเดียวกับ 2D convolution) และเปรียบเทียบเทมเพลตและแพทช์ของภาพอินพุทภายใต้ภาพเทมเพลต มีการใช้วิธีการเปรียบเทียบหลายวิธีใน OpenCV (คุณสามารถตรวจสอบเอกสารสำหรับรายละเอียดเพิ่มเติมได้) จะส่งกลับภาพระดับสีเทาโดยที่แต่ละพิกเซลระบุว่าพื้นที่ใกล้เคียงของพิกเซลนั้นตรงกับเทมเพลตเท่าใด

หากภาพที่นำเข้ามีขนาด*(WxH)*และแม่แบบภาพมีขนาด*(WxH)* , ภาพออกจะมีขนาดของ*(W-W + 1, H-H +*1) เมื่อได้ผลลัพธ์แล้วคุณสามารถใช้**ฟังก์ชัน cv2.minMaxLoc ()**เพื่อหาค่าสูงสุด / ต่ำสุดได้ที่ใด ใช้มุมนี้เป็นมุมซ้ายบนของสี่เหลี่ยมผืนผ้าและใช้ความกว้างและความสูงของรูปสี่เหลี่ยมผืนผ้า*(w, h)*สี่เหลี่ยมผืนผ้าเป็นพื้นที่ของเทมเพลต

**บันทึก :** ถ้าคุณใช้cv2.TM\_SQDIFFเป็นวิธีการเปรียบเทียบค่าต่ำสุดให้การจับคู่ที่ดีที่สุด

การจับคู่เทมเพลตใน OpenCV

นี่เป็นตัวอย่างเราจะค้นหาใบหน้า Messi ในรูปภาพของเขา ดังนั้นฉันจึงสร้างแม่แบบดังนี้:

ภาพเทมเพลต

เราจะลองใช้วิธีการเปรียบเทียบทั้งหมดเพื่อให้เราสามารถดูผลการค้นหาของพวกเขาได้อย่างไร:

**import** **cv2**

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

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

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

img2 = img.copy()

template = cv2.imread('template.jpg',0)

w, h = template.shape[::-1]

*# All the 6 methods for comparison in a list*

methods = ['cv2.TM\_CCOEFF', 'cv2.TM\_CCOEFF\_NORMED', 'cv2.TM\_CCORR',

'cv2.TM\_CCORR\_NORMED', 'cv2.TM\_SQDIFF', 'cv2.TM\_SQDIFF\_NORMED']

**for** meth **in** methods:

img = img2.copy()

method = eval(meth)

*# Apply template Matching*

res = cv2.matchTemplate(img,template,method)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

*# If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum*

**if** method **in** [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

top\_left = min\_loc

**else**:

top\_left = max\_loc

bottom\_right = (top\_left[0] + w, top\_left[1] + h)

cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

plt.subplot(121),plt.imshow(res,cmap = 'gray')

plt.title('Matching Result'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(img,cmap = 'gray')

plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

plt.suptitle(meth)

plt.show()

ดูผลลัพธ์ด้านล่าง:

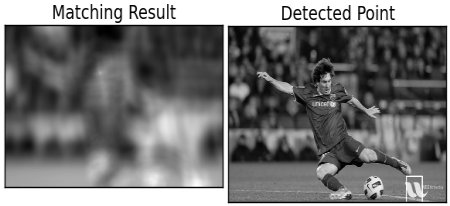
* cv2.TM\_CCOEFF



* cv2.TM\_CCOEFF\_NORMED



* cv2.TM\_CCORR



* cv2.TM\_CCORR\_NORMED



* cv2.TM\_SQDIFF



* cv2.TM\_SQDIFF\_NORMED



คุณสามารถเห็นได้ว่าผลการค้นหาโดยใช้cv2.TM\_CCORRไม่ดีเท่าที่เราคาดไว้

การจับคู่แม่แบบกับวัตถุหลายชิ้น

ในส่วนก่อนหน้านี้เราค้นหาภาพใบหน้า Messi ซึ่งเกิดขึ้นเพียงครั้งเดียวในภาพเท่านั้น สมมติว่าคุณกำลังค้นหาวัตถุที่มีการเกิดขึ้นหลายครั้งcv2.minMaxLoc ()จะไม่ให้ตำแหน่งทั้งหมด ในกรณีนี้เราจะใช้ thresholding ดังนั้นในตัวอย่างนี้เราจะใช้ภาพหน้าจอของเกมMario ที่มีชื่อเสียงและเราจะหาเหรียญในนั้น

**import** **cv2**

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

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

img\_rgb = cv2.imread('mario.png')

img\_gray = cv2.cvtColor(img\_rgb, cv2.COLOR\_BGR2GRAY)

template = cv2.imread('mario\_coin.png',0)

w, h = template.shape[::-1]

res = cv2.matchTemplate(img\_gray,template,cv2.TM\_CCOEFF\_NORMED)

threshold = 0.8

loc = np.where( res >= threshold)

**for** pt **in** zip(\*loc[::-1]):

cv2.rectangle(img\_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)

cv2.imwrite('res.png',img\_rgb)

ผล:

