## **การใช้งานพื้นฐานเกี่ยวกับรูปภาพ**

เป้าหมาย

## เรียนรู้เพื่อ:

* เข้าถึงค่าพิกเซลและแก้ไขค่าเหล่านี้
* เข้าถึงคุณสมบัติภาพ
* การตั้งค่าภูมิภาคของภาพ (ROI)
* การแยกและรวมภาพ

การดำเนินการเกือบทั้งหมดในส่วนนี้เกี่ยวข้องกับ Numpy มากกว่า OpenCV ความรู้ที่ดีเกี่ยวกับ Numpy จำเป็นต้องเขียน CODE ที่ดีขึ้นด้วย OpenCV

*(ตัวอย่างจะปรากฏใน terminal Python เนื่องจากส่วนใหญ่เป็นเพียงรหัสบรรทัดเดียว)*

การเข้าถึงและแก้ไขค่าพิกเซล

## ลองโหลดภาพสีก่อน:

**import** **cv2**

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

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

คุณสามารถเข้าถึงค่าพิกเซลตามแถวและพิกัดของคอลัมน์ สำหรับภาพ BGR จะแสดงอาร์เรย์ของค่า Blue, Green, Red สำหรับภาพสีเทาจะส่งกลับค่าความเข้มที่ตรงกัน

px = img[100,100]

**print** px

[157 166 200]

# accessing only blue pixel

blue = img[100,100,0]

**print** blue

157

คุณสามารถปรับเปลี่ยนค่าพิกเซลได้เช่นเดียวกัน

**>>>** img[100,100] = [255,255,255]

**>>> print** img[100,100]

[255 255 255]

การเตือน : Numpy เป็นไลบรารีที่เหมาะสำหรับการคำนวณอาร์เรย์ที่รวดเร็ว ดังนั้นการเข้าถึงค่าพิกเซลแต่ละพิกเซลและการปรับเปลี่ยนจะช้ามากและไม่เป็นที่พอใจ

บันทึก : วิธีการดังกล่าวข้างต้นถูกใช้โดยทั่วไปสำหรับการเลือกพื้นที่ของอาร์เรย์ 5 แถวแรกและ 3 คอลัมน์สุดท้ายเช่นนี้ สำหรับการเข้าถึงพิกเซลแต่ละวิธีอาร์เรย์ Numpy, array.item ()และarray.itemset ()จะถือว่าดีขึ้น แต่มักจะส่งกลับค่า scalar ดังนั้นถ้าคุณต้องการเข้าถึงค่า B, G, R ทั้งหมดคุณจำเป็นต้องเรียก array.item ()แยกกันสำหรับทั้งหมด

วิธีการเข้าถึงและแก้ไขพิกเซลที่ดีขึ้น:

การเข้าถึงคุณสมบัติภาพ

## คุณสมบัติรูปภาพประกอบด้วยจำนวนแถวคอลัมน์และช่องประเภทข้อมูลรูปภาพจำนวนพิกเซล ฯลฯ

## รูปร่างของภาพที่มีการเข้าถึงโดย img.shape จะส่งกลับจำนวนทวีคูณของแถว,คอลัมน์และช่อง (ถ้ารูปภาพเป็นสี):

**>>> print** img.shape

(342, 548, 3)

จำนวนพิกเซลทั้งหมดที่เข้าถึงได้โดย img.size :

**>>> print** img.size

562248

ประเภทข้อมูลรูปภาพได้จากimg.dtype

รูปภาพ ROI

บางครั้งคุณจะต้องเล่นกับบางพื้นที่ของภาพ สำหรับการตรวจจับตาในภาพการตรวจจับใบหน้าครั้งแรกจะกระทำทั่วทั้งภาพและเมื่อได้รับใบหน้าแล้วเราจะเลือกพื้นที่ใบหน้าเพียงอย่างเดียวและค้นหาดวงตาภายในแทนการค้นหาทั้งภาพ ปรับปรุงความถูกต้อง (เพราะดวงตาอยู่บนใบหน้าเสมอ) และประสิทธิภาพ (เนื่องจากเราค้นหาพื้นที่ขนาดเล็ก)

ROI ได้รับการจัดทำดัชนี Numpy อีกครั้ง ที่นี่ฉันเลือกลูกบอลและคัดลอกไปยังพื้นที่อื่นในภาพ:

ตรวจสอบผลลัพธ์ด้านล่าง:



แยกและรวมช่องภาพ

บางครั้งคุณจำเป็นต้องทำงานแยกต่างหากจากช่อง B, G, R จากนั้นคุณจะต้องแบ่งภาพ BGR เป็นระนาบเดี่ยว หรืออีกครั้งคุณอาจต้องเข้าร่วมช่องเหล่านี้กับภาพ BGR คุณสามารถทำได้ง่ายๆโดย:

**>>>** b,g,r = cv2.split(img)

**>>>** img = cv2.merge((b,g,r))

หรือ

**>>>** b = img[:,:,0]

สมมติว่าคุณต้องการให้พิกเซลสีแดงทั้งหมดเป็นศูนย์คุณไม่จำเป็นต้องแบ่งแบบนี้และให้เท่ากับศูนย์ คุณสามารถใช้การจัดทำดัชนีของ Numpy ได้อย่างรวดเร็วและเร็วขึ้น

**>>>** img[:,:,2] = 0

สร้างเส้นขอบสำหรับภาพ (Padding)

หากคุณต้องการที่จะสร้างเส้นขอบรอบ ๆ ภาพบางอย่างเช่นกรอบรูป, คุณสามารถใช้ cv2.copyMakeBorder ()ฟังก์ชั่น แต่ก็มีการใช้งานมากขึ้นสำหรับการดำเนินการ convolution, padding ศูนย์ ฯลฯ ฟังก์ชันนี้ใช้อาร์กิวเมนต์ต่อไปนี้:

* src - ภาพนำเข้า
* ด้านบน , ล่าง , ซ้าย , ขวา - ความกว้างของเส้นขอบพิกเซลจำนวนพิกเซลในทิศทางที่สอดคล้องกัน
* borderType - Flag กำหนดชนิดของเส้นขอบที่จะเพิ่ม ประเภทดังต่อไปนี้:
  + cv2.BORDER\_CONSTANT - เพิ่มเส้นขอบสีคงที่ ค่าควรได้รับเป็นอาร์กิวเมนต์ถัดไป
  + cv2.BORDER\_REFLECT - เส้นขอบจะสะท้อนเงาขององค์ประกอบเส้นขอบเช่น: fedcba | abcdefgh | hgfedcb
  + cv2.BORDER\_REFLECT\_101หรือcv2.BORDER\_DEFAULT - เหมือนกับข้างต้น แต่มีการเปลี่ยนแปลงเล็กน้อยเช่นนี้gfedcb | abcdefgh | gfedcba
  + cv2.BORDER\_REPLICATE - องค์ประกอบสุดท้ายถูกจำลองแบบตลอดเช่นนี้aaaaaa | abcdefgh | hhhhhhhh
  + cv2.BORDER\_WRAP - ไม่สามารถอธิบายได้ว่าจะมีลักษณะดังนี้: cdefgh | abcdefgh | abcdefg
* ค่า - สีของเส้นขอบถ้าชนิดเส้นขอบคือcv2.BORDER\_CONSTANT

ด้านล่างนี้เป็นตัวอย่างรหัสที่แสดงถึงเส้นขอบทั้งหมดนี้เพื่อความเข้าใจที่ดีขึ้น:

**import** **cv2**

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

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

BLUE = [255,0,0]

img1 = cv2.imread('opencv\_logo.png')

replicate = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REPLICATE)

reflect = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REFLECT)

reflect101 = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REFLECT\_101)

wrap = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_WRAP)

constant= cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_CONSTANT,value=BLUE)

plt.subplot(231),plt.imshow(img1,'gray'),plt.title('ORIGINAL')

plt.subplot(232),plt.imshow(replicate,'gray'),plt.title('REPLICATE')

plt.subplot(233),plt.imshow(reflect,'gray'),plt.title('REFLECT')

plt.subplot(234),plt.imshow(reflect101,'gray'),plt.title('REFLECT\_101')

plt.subplot(235),plt.imshow(wrap,'gray'),plt.title('WRAP')

plt.subplot(236),plt.imshow(constant,'gray'),plt.title('CONSTANT')

plt.show()