**การแนะนำ SURF (คุณสมบัติที่แข็งแรงขึ้น)**

เป้าหมาย

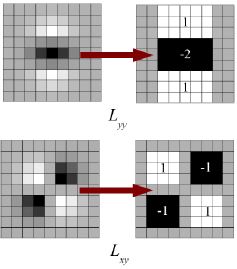
ในบทนี้,

* เราจะดูข้อมูลพื้นฐานของ SURF
* เราจะเห็นฟังก์ชันการทำงาน SURF ใน OpenCV

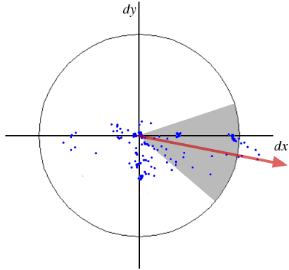
ทฤษฎี

ในบทสุดท้ายเราเห็น SIFT สำหรับการตรวจหาคีย์และคำอธิบาย แต่มันก็ค่อนข้างช้าและคนต้องการรุ่น speeded ขึ้นมากขึ้น ในปี 2549 มีคนสามคนคือ Bay H. H. Tuytelaars T. และ Van Gool L ได้ตีพิมพ์บทความเรื่อง "SURF: Speeded Up Features ที่แข็งแกร่ง" ซึ่งนำเสนออัลกอริทึมใหม่ที่เรียกว่า SURF เป็นชื่อแนะนำมันเป็นรุ่นที่รวดเร็วขึ้นของ SIFT

ใน SIFT, Lowe ประมาณ Laplacian ของ Gaussian กับความแตกต่างของ Gaussian เพื่อหาพื้นที่ขนาด SURF ไปอีกเล็กน้อยและประมาณกับ LoG กับ Filter Box ด้านล่างภาพแสดงการสาธิตของการประมาณดังกล่าว ข้อได้เปรียบอย่างหนึ่งของการประมาณนี้ก็คือการรวมตัวกับกล่องกรองสามารถคำนวณได้ง่ายด้วยความช่วยเหลือของภาพรวม และสามารถทำได้แบบขนานสำหรับเครื่องชั่งที่แตกต่างกัน นอกจากนี้พึ่งพาปัจจัยของเฮสเซอรี่เมทริกซ์สำหรับทั้งระดับและสถานที่



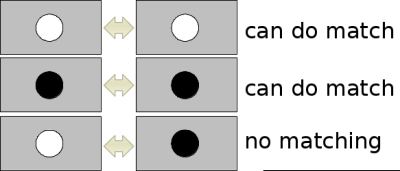
สำหรับการกำหนดทิศทาง SURF ใช้การตอบสนองเวฟเล็ตในทิศทางแนวนอนและแนวตั้งสำหรับพื้นที่ใกล้เคียงขนาด 6 วินาที นอกจากนี้ยังมีการใช้น้ำหนักเครื่องชั่งน้ำหนักที่เพียงพอ จากนั้นจะมีการวางแผนในช่องว่างตามภาพด้านล่าง การปฐมนิเทศที่สำคัญจะคำนวณโดยการคำนวณผลรวมของการตอบสนองทั้งหมดภายในหน้าต่างเลื่อนมุม 60 องศา สิ่งที่น่าสนใจคือการตอบสนองเวฟเล็ตสามารถพบได้โดยใช้ภาพรวมอย่างง่ายดายที่ระดับใด ๆ สำหรับการใช้งานหลายประเภทความไม่แปรเปลี่ยนของการหมุนไม่จำเป็นต้องใช้ดังนั้นจึงไม่จำเป็นต้องหาทิศทางนี้ซึ่งจะช่วยเพิ่มความเร็วในกระบวนการ SURF มีฟังก์ชันที่เรียกว่า Upright-SURF หรือ U-SURF จะช่วยเพิ่มความเร็วและมีเสถียรภาพมาก\ n 15 ^ {\ circ}เกินไป OpenCV รองรับทั้งสองแบบขึ้นอยู่กับธงตรง. ถ้าเป็น 0 จะคำนวณการวางแนว ถ้าเป็น 1 การวางแนวจะไม่คำนวณและเร็วขึ้น



สำหรับคำอธิบายคุณลักษณะ SURF จะใช้การตอบสนองเวฟเล็ตในทิศทางแนวนอนและแนวตั้ง (อีกครั้งการใช้ภาพรวมทำให้สิ่งต่างๆง่ายขึ้น) พื้นที่ใกล้เคียงของขนาด 20sX20s ถูกยึดรอบจุดสำคัญที่ s มีขนาด แบ่งออกเป็น 4 กลุ่มย่อย สำหรับแต่ละอนุภูมิภาค, v = (\ sum {d_x}, \ sum {d_y}, \ sum {| d_x |}, \ sum {| d_y |}แนวนอนและแนวการตอบสนองเวฟจะถูกนำและเวกเตอร์จะเกิดขึ้นเช่นนี้ นี้เมื่อแสดงเป็นเวกเตอร์ให้ตัวบอกลักษณะ SURF มีทั้งหมด 64 มิติ ลดมิติข้อมูลเพิ่มความเร็วในการคำนวณและการจับคู่ แต่ให้ความโดดเด่นที่เหนือกว่าของคุณลักษณะ

สำหรับความแตกต่างมากขึ้น descriptor SURF คุณลักษณะมีรุ่น 128 มิติที่ขยาย ผลรวมของd_xและ| d_x |คำนวณแยกต่างหากสำหรับd_y <0และd_y \ geq 0. ในทำนองเดียวกันผลรวมของd_yและ| d_y |มีการแบ่งแยกตามเครื่องหมายของd_xดังนั้นจึงเป็นสองเท่าของจำนวนคุณลักษณะ ไม่ซับซ้อนมากขึ้น OpenCV สนับสนุนทั้งโดยตั้งค่าธงที่ขยายด้วย 0 และ 1 สำหรับ 64-dim และ 128-dim ตามลำดับ (ค่าเริ่มต้นคือ 128-dim)

การปรับปรุงที่สำคัญอื่น ๆ ก็คือการใช้เครื่องหมาย Laplacian (ร่องรอยของ Hessian Matrix) สำหรับจุดสนใจที่อ้างอิง ไม่มีค่าใช้จ่ายในการคำนวณเนื่องจากมีการคำนวณระหว่างการตรวจสอบแล้ว เครื่องหมายของ Laplacian แยกความสว่างบนพื้นหลังที่มืดออกจากสถานการณ์ย้อนกลับ ในขั้นตอนการจับคู่เราจะเปรียบเทียบคุณลักษณะเฉพาะหากมีความคมชัดเหมือนกัน (ดังแสดงในรูปด้านล่าง) ข้อมูลที่น้อยที่สุดนี้ช่วยให้สามารถจับคู่ได้เร็วขึ้นโดยไม่ต้องลดประสิทธิภาพการทำงานของ descriptor



ในระยะสั้น SURF เพิ่มคุณสมบัติมากมายเพื่อปรับปรุงความเร็วในทุกขั้นตอน การวิเคราะห์แสดงว่าเร็วกว่า SIFT ถึง 3 เท่าในขณะที่ประสิทธิภาพเทียบเท่ากับ SIFT SURF สามารถจัดการภาพที่เบลอและหมุนได้ แต่ไม่ดีในการจัดการกับการเปลี่ยนแปลงมุมมองและการเปลี่ยนแปลงการส่องสว่าง

SURF ใน OpenCV

OpenCV ให้ฟังก์ชัน SURF เช่นเดียวกับ SIFT คุณเริ่มต้นวัตถุ SURF กับเงื่อนไขบางอย่างเช่น 64/128-dim descriptors, Upright / Normal SURF ฯลฯ รายละเอียดทั้งหมดจะอธิบายได้ดีในเอกสาร จากนั้นในขณะที่เราทำใน SIFT เราสามารถใช้ SURF.detect (), SURF.compute () ฯลฯ ในการค้นหา keypoints และ descriptor

ขั้นแรกเราจะดูการสาธิตง่ายๆเกี่ยวกับวิธีการหาคีย์พอยน์เซอร์และ descriptors SURF และวาดมัน ตัวอย่างทั้งหมดจะแสดงในเทอร์มินัล Python เนื่องจากมีลักษณะเหมือนกับ SIFT เท่านั้น

**>>>** img = cv2.imread('fly.png',0)

# Create SURF object. You can specify params here or later.

# Here I set Hessian Threshold to 400

**>>>** surf = cv2.SURF(400)

# Find keypoints and descriptors directly

**>>>** kp, des = surf.detectAndCompute(img,None)

**>>>** len(kp)

699

จุดสำคัญ 1199 มีมากเกินไปที่จะแสดงในรูปภาพ เราลดจำนวนภาพลงเหลือประมาณ 50 ภาพ ในขณะที่จับคู่เราอาจต้องการคุณลักษณะทั้งหมดเหล่านี้ แต่ตอนนี้ไม่ได้ ดังนั้นเราจึงเพิ่มเกณฑ์ Hessian

# Check present Hessian threshold

>>> print surf.hessianThreshold

400.0

# We set it to some 50000. Remember, it is just for representing in picture.

# In actual cases, it is better to have a value 300-500

>>> surf.hessianThreshold = 50000

# Again compute keypoints and check its number.

>>> kp, des = surf.detectAndCompute(img,None)

>>> print len(kp)

47

มันน้อยกว่า 50 ลองมาวาดบนภาพ

**>>>** img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)

**>>>** plt.imshow(img2),plt.show()

ดูผลลัพธ์ด้านล่าง คุณสามารถเห็นได้ว่า SURF เป็นเหมือนเครื่องตรวจจับหยด ตรวจพบรอยบึกสีขาวบนปีกของผีเสื้อ คุณสามารถทดสอบได้ด้วยภาพอื่น ๆ



ตอนนี้ฉันต้องการใช้ U-SURF เพื่อที่จะไม่พบการวางแนว

# Check upright flag, if it False, set it to True

>>> print surf.upright

False

>>> surf.upright = True

# Recompute the feature points and draw it

>>> kp = surf.detect(img,None)

>>> img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)

>>> plt.imshow(img2),plt.show()

ดูผลลัพธ์ด้านล่าง ทิศทางทั้งหมดจะแสดงในทิศทางเดียวกัน มันเร็วกว่าก่อนหน้านี้ หากคุณกำลังทำงานในกรณีที่การปฐมนิเทศไม่มีปัญหา (เช่นการเย็บตะเข็บพาโนรามา) ฯลฯ นี้จะดีกว่า



สุดท้ายเราตรวจสอบขนาด descriptor และเปลี่ยนเป็น 128 ถ้าเป็นเพียง 64-dim

# Find size of descriptor

>>> print surf.descriptorSize()

64

# That means flag, "extended" is False.

>>> surf.extended

False

# So we make it to True to get 128-dim descriptors.

>>> surf.extended = True

>>> kp, des = surf.detectAndCompute(img,None)

>>> print surf.descriptorSize()

128

>>> print des.shape

(47, 128)

ส่วนที่เหลืออยู่ตรงกับที่เราจะทำในบทอื่น