**ความรู้เบื้องต้นเกี่ยวกับ SIFT (การแปลงองค์ประกอบแบบไม่แปรเปลี่ยน)**

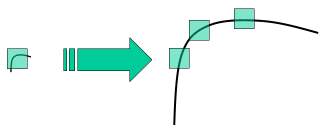
เป้าหมาย

ในบทนี้,

* เราจะเรียนรู้เกี่ยวกับแนวคิดของอัลกอริทึม SIFT
* เราจะเรียนรู้ที่จะหา SIFT Keypoints และ Descriptors

ทฤษฎี

ในคู่สุดท้ายของบทที่เราเห็นเครื่องตรวจจับมุมบางอย่างเช่นแฮร์ริส ฯลฯ พวกเขาหมุนคงที่ซึ่งหมายความว่าแม้ว่าภาพจะหมุนเราสามารถหามุมเดียวกัน เป็นที่ชัดเจนเพราะมุมยังคงมุมในภาพที่หมุนด้วย แต่สิ่งที่เกี่ยวกับการปรับ? มุมอาจไม่ใช่มุมหากปรับสัดส่วนภาพ ตัวอย่างเช่นตรวจสอบภาพง่ายๆด้านล่าง มุมในภาพขนาดเล็กภายในหน้าต่างเล็ก ๆ จะแบนเมื่อซูมเข้าในหน้าต่างเดียวกัน มุมแฮร์ริสจึงไม่แปรผันตามขนาด



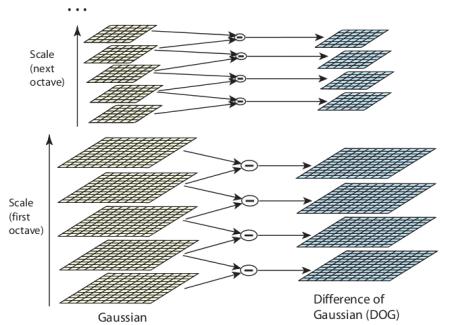
ดังนั้นในปี 2004 D.Loweมหาวิทยาลัยบริติชโคลัมเบียจึงได้ใช้อัลกอริธึมใหม่การปรับเปลี่ยนคุณสมบัติที่ไม่แปรเปลี่ยน (SLEAD) ในเอกสารของเขาคุณลักษณะภาพที่โดดเด่นจาก Keypoints ที่ไม่ขึ้นกับขนาดซึ่งจะดึงคีย์พอยท์และคำนวณ descriptors ของมัน (บทความนี้เป็นเรื่องง่ายที่จะเข้าใจและถือว่าเป็นวัสดุที่ดีที่สุดที่มีอยู่บนร่อน. ดังนั้นคำอธิบายนี้เป็นเพียงการสรุปสั้น ๆ ของบทความนี้)

ส่วนใหญ่มีขั้นตอนสี่ขั้นตอนที่เกี่ยวข้องกับอัลกอริทึม SIFT เราจะเห็นพวกเขาทีละคน

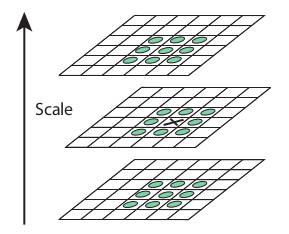
1. Scale-space Extrema Detection

จากภาพด้านบนเราเห็นได้ชัดว่าเราไม่สามารถใช้หน้าต่างเดียวกันในการตรวจหาคีย์พอยน์ที่มีขนาดต่างกัน เป็นมุมที่มีมุมเล็ก ๆ แต่ในการตรวจจับมุมขนาดใหญ่เราจำเป็นต้องมีหน้าต่างขนาดใหญ่ สำหรับกรณีนี้ระบบจะใช้การกรองแบบช่องว่างขนาดใหญ่ ในนั้นพบ Laplacian of Gaussian สำหรับภาพที่มี\ ซิกค่าต่างๆ เข้าสู่ระบบทำหน้าที่เป็นเครื่องตรวจจับหยดซึ่งตรวจจับ blobs \ ซิกในขนาดต่างๆเนื่องจากการเปลี่ยนแปลง ในระยะสั้น\ ซิกทำหน้าที่เป็นพารามิเตอร์การปรับขนาด ตัวอย่างเช่นในภาพด้านบน kernel แบบเกาส์กับ low \ ซิกจะให้ค่ามุมที่เล็กที่สุดในขณะที่เคอร์เนล guassian สูง\ ซิกเหมาะกับมุมที่ใหญ่ขึ้น ดังนั้นเราจึงสามารถหา maxima ในพื้นที่ทั่วทั้งขนาดและพื้นที่ซึ่งจะแสดงรายการ(x, y, \ ซิก)ค่าซึ่งหมายความว่ามี keypoint ที่อาจเป็น (x, y) ใน\ ซิกระดับ

แต่นี้ LoG เป็นค่าใช้จ่ายเพียงเล็กน้อยดังนั้นอัลกอริทึม SIFT ใช้ความแตกต่างของ Gaussians ซึ่งเป็นค่าประมาณของ LoG ความแตกต่างของเกาส์จะได้รับเป็นความแตกต่างของการเบลอของภาพเสียนของภาพที่มีสองที่แตกต่างกัน\ ซิกให้มันเป็นและ\ ซิก k \ ซิกขั้นตอนนี้ทำขึ้นสำหรับ octaves ที่ต่างกันของภาพใน Gaussian Pyramid มันแสดงอยู่ในภาพด้านล่าง:



เมื่อค้นพบเอกสารนี้จะมีการค้นหาภาพ extrema ในพื้นที่มากกว่าขนาดและพื้นที่ ตัวอย่างเช่นหนึ่งพิกเซลในภาพจะถูกเปรียบเทียบกับ 8 เพื่อนบ้านและ 9 พิกเซลในระดับต่อไปและ 9 พิกเซลในเครื่องชั่งที่ผ่านมา ถ้าเป็น extrema ในท้องถิ่นก็จะเป็น keypoint ที่มีศักยภาพ โดยทั่วไปหมายถึงคีย์พ้อยท์ที่ดีที่สุดในระดับดังกล่าว มันแสดงให้เห็นในภาพด้านล่าง:



เกี่ยวกับพารามิเตอร์ที่แตกต่างกัน, กระดาษให้ข้อมูลบางอย่างเกี่ยวกับการทดลองซึ่งสามารถสรุปได้จำนวนเลอะเลือน = 4, จำนวนของระดับขนาด = 5, เริ่มต้น\ ซิก = 1.6, K = \ sqrt {2}ฯลฯ เป็นค่าที่ดีที่สุด

2. Keypoint Localization

เมื่อพบสถานที่สำคัญที่สำคัญแล้วพวกเขาจะต้องได้รับการปรับแต่งเพื่อให้ได้ผลลัพธ์ที่ถูกต้องมากขึ้น พวกเขาใช้เทย์เลอร์ขยายพื้นที่ของสเกลเพื่อให้ได้ตำแหน่งที่ถูกต้องมากยิ่งขึ้นและหากความเข้มของ extrema น้อยกว่าค่าที่กำหนด (0.03 ตามกระดาษ) ก็จะถูกปฏิเสธ เกณฑ์นี้เรียกว่าcontrastThresholdใน OpenCV

DoG มีการตอบสนองที่สูงขึ้นสำหรับขอบดังนั้นขอบต้องถอดออกด้วย สำหรับแนวคิดนี้จะใช้แนวคิดที่คล้ายกับเครื่องตรวจจับมุมแฮร์ริส พวกเขาใช้เมทริกซ์ Hessian 2x2 (H) เพื่อคำนวณความโค้งส่วนโค้ง เราทราบจากเครื่องตรวจจับมุมแฮร์ริสว่าสำหรับขอบหนึ่งค่า eigen หนึ่งมีค่ามากกว่าค่าอื่น ดังนั้นที่นี่พวกเขาใช้ฟังก์ชันง่ายๆ,

ถ้าอัตราส่วนนี้มากกว่า threshold ที่เรียกว่าedgeThresholdใน OpenCV คีย์พอยต์ดังกล่าวจะถูกละทิ้ง มันถูกกำหนดเป็น 10 ในกระดาษ

ดังนั้นจึงช่วยลดจุดสำคัญและคีย์แพดคีย์ที่มีความคมชัดต่ำและสิ่งที่ยังคงเป็นจุดสนใจที่แข็งแกร่ง

3. การกำหนดปฐมนิเทศ

ขณะนี้การวางแนวจะถูกกำหนดให้กับคีย์พอยต์แต่ละจุดเพื่อให้เกิดการหมุนภาพได้อย่างไม่เปลี่ยนแปลง ขอบเขตของพื้นที่รอบ ๆ จะขึ้นอยู่กับขนาดและมีการคำนวณขนาดและทิศทางการไล่ระดับสีในบริเวณนั้น มีการสร้างฮิสโทแกรมการจัดแนวด้วย 36 ถังซึ่งครอบคลุม 360 องศา (มีน้ำหนักโดยการไล่ระดับสีและหน้าต่างแบบวงกลมที่มีน้ำหนักประมาณ gaussian \ ซิกเท่ากับ 1.5 เท่าของคีย์พอยต์ยอดสูงสุดในฮิสโตแกรมจะถูกนำมาใช้และยอดที่สูงกว่า 80% ของมันจะถูกพิจารณาในการคำนวณการวางแนวด้วยเช่นกันสร้างคีย์พอยน์ มีตำแหน่งและระดับเดียวกัน แต่มีทิศทางที่แตกต่างกันซึ่งจะมีส่วนช่วยในการจับคู่

4. Descriptor พวงกุญแจ

ตอนนี้สร้างคีย์อธิบายพวงกุญแจแล้ว ถ่ายทำบริเวณรอบ ๆ 16x16 รอบจุดสำคัญ แบ่งออกเป็น 16 บล็อกย่อยขนาด 4x4 สำหรับแต่ละบล็อกย่อยจะมีการสร้างฮิสโตแกรมการจัดวาง 8 แชนแนล ดังนั้นมีค่า 128 ถังอยู่ จะแสดงเป็นเวกเตอร์เพื่อสร้างตัวบ่งชี้จุดสำคัญ (keypoint descriptor) นอกจากนี้ยังมีมาตรการหลายอย่างเพื่อให้เกิดความทนทานต่อการเปลี่ยนแปลงการส่องสว่างการหมุน ฯลฯ

5. การจับคู่

Keypoint ระหว่างภาพสองภาพจะจับคู่โดยการระบุเพื่อนบ้านที่อยู่ใกล้ที่สุด แต่ในบางกรณีการจับคู่ที่ใกล้เคียงที่สุดครั้งที่สองอาจอยู่ใกล้กับอันดับแรก อาจเกิดขึ้นเนื่องจากเสียงรบกวนหรือเหตุผลอื่น ๆ ในกรณีนี้จะใช้อัตราส่วนระยะใกล้เคียงกับระยะทางที่ใกล้ที่สุดเป็นอันดับสอง หากมีค่ามากกว่า 0.8 จะถูกปฏิเสธ มัน eliminaters ประมาณ 90% ของการแข่งขันที่ผิดพลาดในขณะที่ทิ้งเพียง 5% ตรงกับที่ถูกต้องตามที่กระดาษ

ดังนั้นนี่คือบทสรุปของอัลกอริทึม SIFT สำหรับข้อมูลเพิ่มเติมและทำความเข้าใจขอแนะนำให้อ่านเอกสารต้นฉบับ จำสิ่งหนึ่งที่อัลกอริทึมนี้ได้รับการจดสิทธิบัตรแล้ว ดังนั้นอัลกอริทึมนี้รวมอยู่ในโมดูล Non-free ใน OpenCV

SIFT ใน OpenCV

ตอนนี้ขอดูฟังก์ชัน SIFT ที่มีอยู่ใน OpenCV เริ่มต้นด้วยการตรวจหา keypoint และวาดมัน ก่อนอื่นเราต้องสร้างวัตถุ SIFT เราสามารถส่งพารามิเตอร์ที่แตกต่างกันไปซึ่งเป็นตัวเลือกและอธิบายได้ดีในเอกสาร

**import** **cv2**

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

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

gray= cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

sift = cv2.SIFT()

kp = sift.detect(gray,None)

img=cv2.drawKeypoints(gray,kp)

cv2.imwrite('sift\_keypoints.jpg',img)

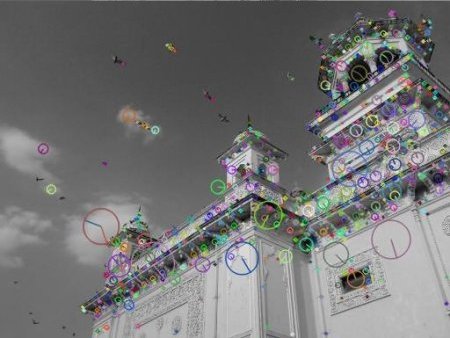
ฟังก์ชัน sift.detect ()พบคีย์พอยน์ในภาพ คุณสามารถผ่านหน้ากากได้หากต้องการค้นหาเฉพาะส่วนของภาพ จุดสำคัญแต่ละตัวเป็นโครงสร้างพิเศษซึ่งมีคุณลักษณะหลายอย่างเช่นพิกัด (x, y) ขนาดของละแวกใกล้เคียงที่มีความหมายมุมซึ่งระบุการวางแนวการตอบสนองที่ระบุความแข็งแรงของจุดสำคัญ ฯลฯ

OpenCV ยังมีฟังก์ชัน cv2.drawKeyPoints ()ที่ดึงวงกลมขนาดเล็กในตำแหน่งของคีย์พอยท์ ถ้าคุณส่งธงcv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTSไปจะวาดวงกลมที่มีขนาดของ keypoint และจะแสดงการวางแนว ดูตัวอย่างด้านล่าง

img=cv2.drawKeypoints(gray,kp,flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)

cv2.imwrite('sift\_keypoints.jpg',img)

ดูสองผลลัพธ์ด้านล่าง:



ตอนนี้เพื่อคำนวณ descriptor, OpenCV ให้สองวิธี

1. เนื่องจากคุณได้พบคีย์พอยน์แล้วคุณสามารถเรียกใช้sift.compute ()ซึ่งคำนวณตัวบ่งชี้จากคีย์พ็อคที่เราพบ เช่น: kp, des =sift.comp (สีเทา, kp)
2. หากคุณไม่พบ keypoints โดยตรงพบ keypoints และอธิบายในขั้นตอนเดียวกับฟังก์ชั่นsift.detectAndCompute ()

เราจะเห็นวิธีที่สอง:

sift = cv2.SIFT()

kp, des = sift.detectAndCompute(gray,None)

นี่ KP จะมีรายการของ keypoints และ des เป็นอาร์เรย์ numpy Number \ _of \ _Keypoints \ times 128ของรูปร่าง

ดังนั้นเราจึงมีคีย์พอยน์เตอร์อธิบาย ฯลฯ ตอนนี้เราต้องการดูวิธีจับคู่คีย์พอยน์ในภาพที่ต่างกัน ที่เราจะได้เรียนรู้ในบทที่จะมาถึง