**การไหลแบบออพติคอล**

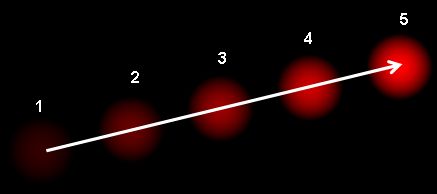
เป้าหมาย

ในบทนี้,

* เราจะเข้าใจแนวความคิดของการไหลของแสงและการประมาณค่าโดยใช้วิธี Lucas-Kanade
* เราจะใช้ฟังก์ชันเช่น**cv2.calcOpticalFlowPyrLK ()**เพื่อติดตามจุดสนใจในวิดีโอ

การไหลแบบออพติคอล

การไหลแบบออปติคัลเป็นรูปแบบการเคลื่อนที่ของวัตถุภาพระหว่างสองเฟรมต่อเนื่องที่เกิดจากการเคลื่อนย้ายวัตถุหรือกล้องถ่ายภาพ เป็นสนามเวกเตอร์ 2D ซึ่งแต่ละเวกเตอร์เป็นเวกเตอร์การเคลื่อนที่ที่แสดงการเคลื่อนที่ของจุดจากเฟรมแรกไปเป็นที่สอง ลองพิจารณาภาพด้านล่าง (รูปภาพมารยาท: [Wikipedia article on Optical Flow](http://en.wikipedia.org/wiki/Optical_flow) )



มันแสดงให้เห็นลูกบอลย้ายใน 5 เฟรมต่อเนื่อง ลูกศรแสดงเวคเตอร์การเคลื่อนที่ การไหลแบบออปติคัลมีแอพพลิเคชันมากมายในพื้นที่เช่น:

* โครงสร้างจาก Motion
* การบีบอัดวิดีโอ
* ความเสถียรของวิดีโอ ...

การไหลของแสงทำงานบนสมมติฐานหลายประการ:

1. ความเข้มของพิกเซลของวัตถุไม่เปลี่ยนแปลงระหว่างเฟรมติดต่อกัน
2. พิกเซลที่อยู่ใกล้เคียงมีการเคลื่อนไหวที่คล้ายคลึงกัน

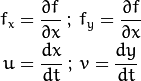
พิจารณาพิกเซลฉัน (x, y, t)ในเฟรมแรก (ตรวจสอบมิติใหม่เวลาจะถูกเพิ่มที่นี่ก่อนหน้านี้เรากำลังทำงานกับภาพเท่านั้นจึงไม่จำเป็นต้องใช้เวลา) มันเคลื่อนไปตามระยะทาง(DX, DY)ในกรอบถัดไปที่ถ่ายหลังจากdtเวลา ดังนั้นเนื่องจากพิกเซลเหล่านี้มีความเหมือนกันและความรุนแรงไม่เปลี่ยนแปลงเราสามารถพูดได้

I (x, y, t) = I (x + dx, y + dy, t + dt)

จากนั้นให้ใช้เทย์เลอร์ประมาณทางด้านขวาให้ลบคำทั่วไปและหารด้วยdtเพื่อให้ได้สมการต่อไปนี้:

f_x u + f_y v + f_t = 0 \;

ที่อยู่:



สมการข้างต้นเรียกว่าสมการ Flow แบบออฟไลน์ ในนั้นเราสามารถค้นหาได้f_xและf_yเป็นภาพไล่ระดับสี ในทำนองเดียวกันf_tคือการไล่ระดับสีตามเวลา แต่(U, V)ไม่เป็นที่รู้จัก เราไม่สามารถแก้สมการหนึ่งอันนี้ได้ด้วยสองตัวแปรที่ไม่รู้จัก ดังนั้นวิธีการต่างๆที่มีให้เพื่อแก้ปัญหานี้และหนึ่งในนั้นคือ Lucas-Kanade

วิธี Lucas-Kanade

เราได้เห็นข้อสมมติฐานก่อนหน้านี้ว่าพิกเซลที่อยู่ใกล้เคียงทั้งหมดจะมีการเคลื่อนไหวที่คล้ายคลึงกัน วิธี Lucas-Kanade ใช้แพทช์ 3x3 รอบจุด ดังนั้นทั้งหมด 9 จุดมีการเคลื่อนไหวเดียวกัน เราสามารถหา(f_x, f_y, f_t)จุดเหล่านี้ได้ 9 จุด ดังนั้นตอนนี้ปัญหาของเรากลายเป็นแก้ 9 สมการที่มีสองตัวแปรที่ไม่รู้จักซึ่งเกินกำหนด วิธีการแก้ปัญหาที่ดีกว่าจะได้รับโดยใช้วิธีการแบบรูปสี่เหลี่ยมผืนผ้าน้อยที่สุด ด้านล่างเป็นทางออกสุดท้ายซึ่งเป็นสมการสองปัญหาที่ไม่รู้จักและแก้ปัญหาเพื่อแก้ปัญหา

\ begin {bmatrix} u \\ v \ end {bmatrix} = \ begin {bmatrix} \ sum_ {i} {f_ {x_i}} ^ 2 & \ sum_ {i} {f_ {x_i} f_ {y_i}} \ \ \ sum_ {i} {f_ {x_i} f_ {y_i}} และ \ sum_ {i} {f_ {y_i}} ^ 2 \ end {bmatrix} ^ {- 1} \ begin {bmatrix} - \ sum_ {i } {f_ {x_i} f_ {t_i}} \\ - \ sum_ {i} {f_ {y_i} f_ {t_i}} \ end {bmatrix}

(ตรวจสอบความคล้ายคลึงกันของเมทริกซ์ผกผันกับแฮร์ริสมุมเครื่องตรวจจับ. มันหมายถึงว่ามุมเป็นจุดที่ดีกว่าที่จะติดตาม.)

ดังนั้นจากมุมมองของผู้ใช้ความคิดเป็นเรื่องง่ายเราจะให้บางประเด็นในการติดตามเราได้รับเวกเตอร์การไหลของจุดแสงเหล่านั้น แต่อีกครั้งมีปัญหาบางอย่าง จนถึงขณะนี้เรากำลังรับมือกับการเคลื่อนไหวเล็ก ๆ ดังนั้นจึงล้มเหลวเมื่อมีการเคลื่อนไหวขนาดใหญ่ อีกครั้งเราไปปิรามิด เมื่อเราขึ้นไปในพีระมิดการเคลื่อนที่เล็ก ๆ จะถูกลบออกและการเคลื่อนไหวใหญ่จะกลายเป็นภาพเคลื่อนไหวขนาดเล็ก ดังนั้นการใช้ลูคัส - คาเนดที่นั่นเราจึงได้รับกระแสแสงพร้อมกับขนาด

การไหลแบบออพติคอล Lucas-Kanade ใน OpenCV

OpenCV ให้ทุกเหล่านี้ในฟังก์ชั่นเดียวcv2.calcOpticalFlowPyrLK () ที่นี่เราสร้างแอปพลิเคชันง่ายๆซึ่งติดตามบางจุดในวิดีโอ การตัดสินใจจุดที่เราจะใช้cv2.goodFeaturesToTrack () เราใช้เฟรมแรกตรวจจับจุดมุมของ Shi-Tomasi ในนั้นจากนั้นเราจะติดตามจุดเหล่านั้นโดยใช้การไหลแบบออพติคอล Lucas-Kanade สำหรับฟังก์ชันcv2.calcOpticalFlowPyrLK ()เราจะผ่านเฟรมก่อนหน้าจุดก่อนหน้าและเฟรมถัดไป จะส่งกลับจุดถัดไปพร้อมด้วยหมายเลขสถานะบางอย่างที่มีค่าเป็น 1 ถ้าพบจุดถัดไปมิฉะนั้นศูนย์ เรานำคะแนนต่อไปนี้ไปใช้เป็นจุดก่อนหน้าในขั้นตอนต่อไป ดูรหัสด้านล่าง:

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

**import** **cv2**

cap = cv2.VideoCapture('slow.flv')

*# params for ShiTomasi corner detection*

feature\_params = dict( maxCorners = 100,

qualityLevel = 0.3,

minDistance = 7,

blockSize = 7 )

*# Parameters for lucas kanade optical flow*

lk\_params = dict( winSize = (15,15),

maxLevel = 2,

criteria = (cv2.TERM\_CRITERIA\_EPS | cv2.TERM\_CRITERIA\_COUNT, 10, 0.03))

*# Create some random colors*

color = np.random.randint(0,255,(100,3))

*# Take first frame and find corners in it*

ret, old\_frame = cap.read()

old\_gray = cv2.cvtColor(old\_frame, cv2.COLOR\_BGR2GRAY)

p0 = cv2.goodFeaturesToTrack(old\_gray, mask = None, \*\*feature\_params)

*# Create a mask image for drawing purposes*

mask = np.zeros\_like(old\_frame)

**while**(1):

ret,frame = cap.read()

frame\_gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

*# calculate optical flow*

p1, st, err = cv2.calcOpticalFlowPyrLK(old\_gray, frame\_gray, p0, None, \*\*lk\_params)

*# Select good points*

good\_new = p1[st==1]

good\_old = p0[st==1]

*# draw the tracks*

**for** i,(new,old) **in** enumerate(zip(good\_new,good\_old)):

a,b = new.ravel()

c,d = old.ravel()

mask = cv2.line(mask, (a,b),(c,d), color[i].tolist(), 2)

frame = cv2.circle(frame,(a,b),5,color[i].tolist(),-1)

img = cv2.add(frame,mask)

cv2.imshow('frame',img)

k = cv2.waitKey(30) & 0xff

**if** k == 27:

**break**

*# Now update the previous frame and previous points*

old\_gray = frame\_gray.copy()

p0 = good\_new.reshape(-1,1,2)

cv2.destroyAllWindows()

cap.release()

(รหัสนี้ไม่ได้ตรวจสอบว่า keypoints ถัดไปถูกต้องอย่างไรดังนั้นแม้ว่าจุดสนใจใด ๆ จะหายไปในภาพ แต่ก็มีโอกาสที่การไหลแบบออพติคัลจะพบจุดต่อ ๆ ไปซึ่งอาจดูใกล้เคียงกับมันดังนั้นจริงๆแล้วสำหรับการติดตามที่มีประสิทธิภาพมุม จุดควรจะตรวจพบในช่วงเวลาที่เฉพาะเจาะจงตัวอย่าง OpenCV มาขึ้นกับตัวอย่างเช่นที่พบจุดคุณลักษณะที่ทุก 5 เฟรมนอกจากนี้ยังเรียกใช้ย้อนกลับตรวจสอบของจุดไหลแสงได้เลือกคนดีเท่านั้นตรวจสอบตัวอย่าง / python2 / lk\_track.py )

ดูผลลัพธ์ที่เราได้รับ:



การไหลแบบออฟติคอลหนาแน่นใน OpenCV

วิธี Lucas-Kanade คำนวณการไหลของแสงสำหรับชุดคุณลักษณะเบาบาง (ในตัวอย่างของเรามุมที่ตรวจพบโดยใช้อัลกอริธึม Shi-Tomasi) OpenCV มีขั้นตอนอีกขั้นหนึ่งเพื่อหาการไหลของแสงแบบหนาแน่น จะคำนวณการไหลของแสงสำหรับทุกจุดในเฟรม มันขึ้นอยู่กับขั้นตอนของ Gunner Farneback ซึ่งอธิบายไว้ใน "การประมาณการการเคลื่อนไหวสองเฟรมตามการขยายตัวของพหุนาม" โดย Gunner Farneback ในปี 2003

ตัวอย่างด้านล่างแสดงวิธีการหาการไหลของแสงที่หนาแน่นโดยใช้ขั้นตอนข้างต้น เราได้รับอาร์เรย์ 2 (U, V)ช่องกับเวกเตอร์การไหลของแสง เราหาขนาดและทิศทางของพวกเขา เราทำโค้ดสีเพื่อให้ได้ภาพที่ดีขึ้น ทิศทางตรงกับค่าสีของภาพ ขนาดสอดคล้องกับระนาบ Value ดูรหัสด้านล่าง:

**import** **cv2**

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

cap = cv2.VideoCapture("vtest.avi")

ret, frame1 = cap.read()

prvs = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

hsv = np.zeros\_like(frame1)

hsv[...,1] = 255

**while**(1):

ret, frame2 = cap.read()

next = cv2.cvtColor(frame2,cv2.COLOR\_BGR2GRAY)

flow = cv2.calcOpticalFlowFarneback(prvs,next, None, 0.5, 3, 15, 3, 5, 1.2, 0)

mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])

hsv[...,0] = ang\*180/np.pi/2

hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM\_MINMAX)

rgb = cv2.cvtColor(hsv,cv2.COLOR\_HSV2BGR)

cv2.imshow('frame2',rgb)

k = cv2.waitKey(30) & 0xff

**if** k == 27:

**break**

**elif** k == ord('s'):

cv2.imwrite('opticalfb.png',frame2)

cv2.imwrite('opticalhsv.png',rgb)

prvs = next

cap.release()

cv2.destroyAllWindows()

ดูผลลัพธ์ด้านล่าง:



OpenCV มาพร้อมกับตัวอย่างที่สูงขึ้นในการไหลของแสงหนาแน่นโปรดดูตัวอย่าง / python2 / opt\_flow.py