El flujo óptico es el patrón de movimiento aparente de los objetos de la imagen entre dos cuadros consecutivos causado por el movimiento del objeto o la cámara. Es un campo vectorial 2D donde cada vector es un vector de desplazamiento que muestra el movimiento de puntos desde el primer cuadro al segundo.

Lucas-Kanade Optical Flow

OpenCV proporciona todo esto en una sola función, cv.calcOpticalFlowPyrLK(). Aquí, creamos una aplicación sencilla que rastrea algunos puntos de un vídeo. Para decidir los puntos usamos cv.goodFeaturesToTrack(). Tomamos el primer cuadro, detectamos algunos puntos de esquina de Shi-Tomasi en él y luego rastreamos iterativamente esos puntos utilizando el flujo óptico de Lucas-Kanade. Para la función cv.calcOpticalFlowPyrLK() pasamos el fotograma anterior, los puntos anteriores y el fotograma siguiente. Devuelve los siguientes puntos junto con algunos números de estado que tienen un valor de 1 si se encuentra el siguiente punto; en caso contrario, cero. Pasamos iterativamente estos siguientes puntos como puntos anteriores en el siguiente paso. Vea el código a continuación:

import numpy as np  
import cv2 as cv  
  
# Ruta al archivo de video  
video\_path = 'resources/slow\_traffic\_small.mp4'  
  
# Crear un objeto VideoCapture apuntando al archivo de video guardado  
cap = cv.VideoCapture(video\_path)  
  
# Parámetros para la detección de esquinas Shi-Tomasi  
feature\_params = dict(maxCorners=100,  
 qualityLevel=0.3,  
 minDistance=7,  
 blockSize=7)  
  
# Parámetros para el flujo óptico de Lucas-Kanade  
lk\_params = dict(winSize=(15, 15),  
 maxLevel=2,  
 criteria=(cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 10, 0.03))  
  
# Crear algunos colores aleatorios  
color = np.random.randint(0, 255, (100, 3))  
  
# Leer el primer fotograma y encontrar las esquinas en él  
ret, old\_frame = cap.read()  
old\_gray = cv.cvtColor(old\_frame, cv.COLOR\_BGR2GRAY)  
p0 = cv.goodFeaturesToTrack(old\_gray, mask=None, \*\*feature\_params)  
  
# Crear una máscara de imagen para propósitos de dibujo  
mask = np.zeros\_like(old\_frame)  
  
while True:  
 ret, frame = cap.read()  
 if not ret:  
 print('¡No se han capturado fotogramas!')  
 break  
 frame\_gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY)  
 # Calcular el flujo óptico  
 p1, st, err = cv.calcOpticalFlowPyrLK(old\_gray, frame\_gray, p0, None, \*\*lk\_params)  
 # Seleccionar buenos puntos  
 if p1 is not None:  
 good\_new = p1[st == 1]  
 good\_old = p0[st == 1]  
 # Dibujar las pistas  
 for i, (new, old) in enumerate(zip(good\_new, good\_old)):  
 a, b = new.ravel()  
 c, d = old.ravel()  
 mask = cv.line(mask, (int(a), int(b)), (int(c), int(d)), color[i].tolist(), 2)  
 frame = cv.circle(frame, (int(a), int(b)), 5, color[i].tolist(), -1)  
 img = cv.add(frame, mask)  
 cv.imshow('frame', img)  
 k = cv.waitKey(30) & 0xff  
 if k == 27:  
 break  
 # Actualizar el fotograma anterior y los puntos anteriores  
 old\_gray = frame\_gray.copy()  
 p0 = good\_new.reshape(-1, 1, 2)  
  
cv.destroyAllWindows()

Dense Optical Flow in OpenCV

El método Lucas-Kanade calcula el flujo óptico para un conjunto de características escasas (en nuestro ejemplo, esquinas detectadas mediante el algoritmo Shi-Tomasi). OpenCV proporciona otro algoritmo para encontrar el flujo óptico denso. Calcula el flujo óptico para todos los puntos del marco. Se basa en el algoritmo de Gunnar Farneback

import numpy as np  
import cv2 as cv  
cap = cv.VideoCapture(cv.samples.findFile("resources/vtest.avi"))  
ret, frame1 = cap.read()  
prvs = cv.cvtColor(frame1, cv.COLOR\_BGR2GRAY)  
hsv = np.zeros\_like(frame1)  
hsv[..., 1] = 255  
while(1):  
 ret, frame2 = cap.read()  
 if not ret:  
 print('No frames grabbed!')  
 break  
 next = cv.cvtColor(frame2, cv.COLOR\_BGR2GRAY)  
 flow = cv.calcOpticalFlowFarneback(prvs, next, None, 0.5, 3, 15, 3, 5, 1.2, 0)  
 mag, ang = cv.cartToPolar(flow[..., 0], flow[..., 1])  
 hsv[..., 0] = ang\*180/np.pi/2  
 hsv[..., 2] = cv.normalize(mag, None, 0, 255, cv.NORM\_MINMAX)  
 bgr = cv.cvtColor(hsv, cv.COLOR\_HSV2BGR)  
 cv.imshow('frame2', bgr)  
 k = cv.waitKey(30) & 0xff  
 if k == 27:  
 break  
 elif k == ord('s'):  
 cv.imwrite('opticalfb.png', frame2)  
 cv.imwrite('opticalhsv.png', bgr)  
 prvs = next  
cv.destroyAllWindows()