MeanShift

Se le proporciona una ventana pequeña (puede ser un círculo) y debe mover esa ventana al área de máxima densidad de píxeles (o número máximo de puntos).

La ventana inicial se muestra en un círculo azul con el nombre "C1". Su centro original está marcado en un rectángulo azul, denominado "C1\_o". Pero si encuentras el centroide de los puntos dentro de esa ventana, obtendrás el punto "C1\_r" (marcado en un pequeño círculo azul) que es el centroide real de la ventana. Seguramente no coinciden. Así que mueva su ventana de manera que el círculo de la nueva ventana coincida con el centroide anterior. Encuentre nuevamente el nuevo centroide. Lo más probable es que no coincida. Así que muévalo nuevamente y continúe las iteraciones de modo que el centro de la ventana y su centroide caigan en la misma ubicación (o dentro de un pequeño error deseado). Finalmente lo que obtienes es una ventana con la máxima distribución de píxeles. Está marcado con un círculo verde, denominado "C2". Como puedes ver en la imagen, tiene número máximo de puntos.

import cv2 as cv  
import numpy as np  
  
# Crear un objeto VideoCapture apuntando al archivo de video guardado  
cap = cv.VideoCapture('resources/slow\_traffic\_small.mp4')  
  
# Leer el primer fotograma del video  
ret, frame = cap.read()  
  
# Configurar la ubicación inicial de la ventana  
x, y, w, h = 300, 200, 100, 50 # simplemente valores codificados  
track\_window = (x, y, w, h)  
  
# Configurar la ROI para el seguimiento  
roi = frame[y:y+h, x:x+w]  
hsv\_roi = cv.cvtColor(roi, cv.COLOR\_BGR2HSV)  
mask = cv.inRange(hsv\_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.)))  
roi\_hist = cv.calcHist([hsv\_roi],[0],mask,[180],[0,180])  
cv.normalize(roi\_hist, roi\_hist, 0, 255, cv.NORM\_MINMAX)  
  
# Configurar el criterio de terminación, ya sea 10 iteraciones o moverse al menos 1 punto  
term\_crit = (cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 10, 1)  
  
while True:  
 # Leer un nuevo fotograma del video  
 ret, frame = cap.read()  
   
 # Si no se lee el fotograma correctamente, salir del bucle  
 if not ret:  
 break  
  
 hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)  
 dst = cv.calcBackProject([hsv],[0],roi\_hist,[0,180],1)  
   
 # Aplicar Mean Shift para obtener la nueva ubicación  
 ret, track\_window = cv.meanShift(dst, track\_window, term\_crit)  
   
 # Dibujar la ventana rastreada en la imagen  
 x, y, w, h = track\_window  
 img2 = cv.rectangle(frame, (x,y), (x+w,y+h), 255, 2)  
   
 # Mostrar la imagen con la ventana rastreada  
 cv.imshow('img2', img2)  
   
 # Esperar y verificar si se presiona la tecla 'Esc' para salir  
 k = cv.waitKey(30) & 0xff  
 if k == 27:  
 break  
  
# Liberar el objeto VideoCapture y cerrar todas las ventanas  
cap.release()  
cv.destroyAllWindows()

Camshift

Hay un problema. Nuestra ventana siempre tiene el mismo tamaño tanto si el coche está muy lejos como muy cerca de la cámara. Eso no es bueno. Necesitamos adaptar el tamaño de la ventana con el tamaño y la rotación del objetivo. Una vez más, la solución provino de "OpenCV Labs" y se llama CAMshift (Continuously Adaptive Meanshift)

Es similar a meanshift, pero devuelve un rectángulo rotado (ese es nuestro resultado) y parámetros de cuadro (que solían pasarse como ventana de búsqueda en la siguiente iteración). Vea el código a continuación:

import numpy as np  
import cv2 as cv  
  
# Crear un objeto VideoCapture apuntando al archivo de video guardado  
cap = cv.VideoCapture('resources/slow\_traffic\_small.mp4')  
  
# Leer el primer fotograma del video  
ret, frame = cap.read()  
  
# Configurar la ubicación inicial de la ventana  
x, y, w, h = 300, 200, 100, 50 # simplemente valores codificados  
track\_window = (x, y, w, h)  
  
# Configurar la ROI para el seguimiento  
roi = frame[y:y+h, x:x+w]  
hsv\_roi = cv.cvtColor(roi, cv.COLOR\_BGR2HSV)  
mask = cv.inRange(hsv\_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.)))  
roi\_hist = cv.calcHist([hsv\_roi],[0],mask,[180],[0,180])  
cv.normalize(roi\_hist, roi\_hist, 0, 255, cv.NORM\_MINMAX)  
  
# Configurar el criterio de terminación, ya sea 10 iteraciones o moverse al menos 1 punto  
term\_crit = (cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 10, 1)  
  
while True:  
 # Leer un nuevo fotograma del video  
 ret, frame = cap.read()  
   
 # Si no se lee el fotograma correctamente, salir del bucle  
 if not ret:  
 break  
  
 hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)  
 dst = cv.calcBackProject([hsv],[0],roi\_hist,[0,180],1)  
   
 # Aplicar CamShift para obtener la nueva ubicación  
 ret, track\_window = cv.CamShift(dst, track\_window, term\_crit)  
   
 # Obtener los puntos del rectángulo rotado  
 pts = cv.boxPoints(ret)  
 pts = np.int0(pts)  
   
 # Dibujar el rectángulo rotado en la imagen  
 img2 = cv.polylines(frame,[pts],True, 255, 2)  
   
 # Mostrar la imagen con el rectángulo rotado  
 cv.imshow('img2', img2)  
   
 # Esperar y verificar si se presiona la tecla 'Esc' para salir  
 k = cv.waitKey(30) & 0xff  
 if k == 27:  
 break  
  
# Liberar el objeto VideoCapture y cerrar todas las ventanas  
cap.release()  
cv.destroyAllWindows()