In [2]:
```
import cv2
import numpy as np

#path to source video
source_path = '/Users/Avlonitis/Desktop/R Datafiles'
#out path to save tracking data
output_path = '/Users/Avlonitis/Desktop/R Datafiles/SIS.csv'

#parameter of lk optical flow
www = 15 #www is the diameter of the object in pixles
lk_params = dict(winSize = (www,www),
                 maxLevel = 20000,
                 criteria = (cv2.TERM_CRITERIA_EPS |
                             cv2.TERM_CRITERIA_COUNT, 20, 0.03))

#create list to capture mouse clicks
click_list = []


#create list to store tracked positions
#positions[x] = the location of the tracked object in frame x
positions = []
for i in range(100000): positions.append((0,0))

#make mouse callback function
def callback(event, x, y, flags, param):
    #if the event is a left button click
    if event == 1: click_list.append((x,y))
cv2.namedWindow('img')
cv2.setMouseCallback('img', callback)

#capture the source video
cap = cv2.VideoCapture('/Users/Avlonitis/Desktop/R Datafiles/SISVid.mp4')

#start at frame number 0
frame_number = 0

#define wait time
wait_time = 0

#create a value for the tracker to track
#this will be reset by the user after the first frame
#p0 is the location of the tracked object in the previous frame
p0 = np.array([[[0,0]]], np.float32)

#print instructions
print ('Click on the object that you want to track.')
print ('Press any key (other than s, esc, or t) to advance one frame.')
print ('Press "s" to start and stop optical flow tracking.')
print ('Press "t" to track quickly.'
)
#main loop
while True:
    #check the length of the click list,
    #later this will be used to determine if a new click has been made
    click_length = len(click_list)

    #read frame of video
    _, img = cap.read()

    #try/except to see if video is over, and convert frame to gray
    try: old_gray = img_gray.copy()
    except: old_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    try: img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    except: break

    #try and track using opticalk flow
    p1, error, _ = cv2.calcOpticalFlowPyrLK(old_gray, img_gray,
                                            p0, None, **lk_params)
    if error==[0][0]: wait_time = 0

    #convert the coordinates to two intigers
    xy = int(p1[0][0][0]), int(p1[0][0][1])
    #update p0 to p1
    p0 = p1
    #make a circle around the object being tracked
    cv2.circle(img, xy, www, (244,4,4), 1)
    cv2.circle(img, xy, www+5, (244,4,4), 1)

    #show the frame and wait
    cv2.imshow('img', img)
    k = cv2.waitKey(wait_time)

    #if the keypress is "t"
    if k == 116: wait_time=38
    if k ==117: wait_time=1

    #if the keypress is "s"
    if k == 115:
        if wait_time == 0:
            print ('press "s" to pause')
            wait_time = 200
        else:
            wait_time = 0
            print ('press "s" to being optical flow tracking'
)
    #if keypress is "esc"
    if k == 27: break

    #check and see if there has been a new click
    if len(click_list) == click_length:
        #no new click: user likes optical flow value
        positions[frame_number] = xy
    else:
        #new click, update p0
        p0 = np.array([[click_list[-1]]], np.float32)
        xy = click_list[-1]
        positions[frame_number] = xy

    #increment frame number
    frame_number += 1

#close the window
cv2.destroyAllWindows()

#reduce length of the positions list to include only the data collected
positions = positions[:frame_number]

print ('finished tracking')

#write data
import csv
with open(output_path, 'w') as csvfile:
    fieldnames = ['x_position', 'y_position']
    writer = csv.DictWriter(csvfile, fieldnames = fieldnames)
    writer.writeheader()
    for position in positions:
        x, y = position[0], position[1]
        writer.writerow({'x_position': x, 'y_position': y})

print ('finished writing data'
    )
```

```
Click on the object that you want to track.
Press any key (other than s, esc, or t) to advance one frame.
Press "s" to start and stop optical flow tracking.
Press "t" to track quickly.
finished tracking
finished writing data
```

In [ ]:

In [ ]: