7-optical-flow-estimation

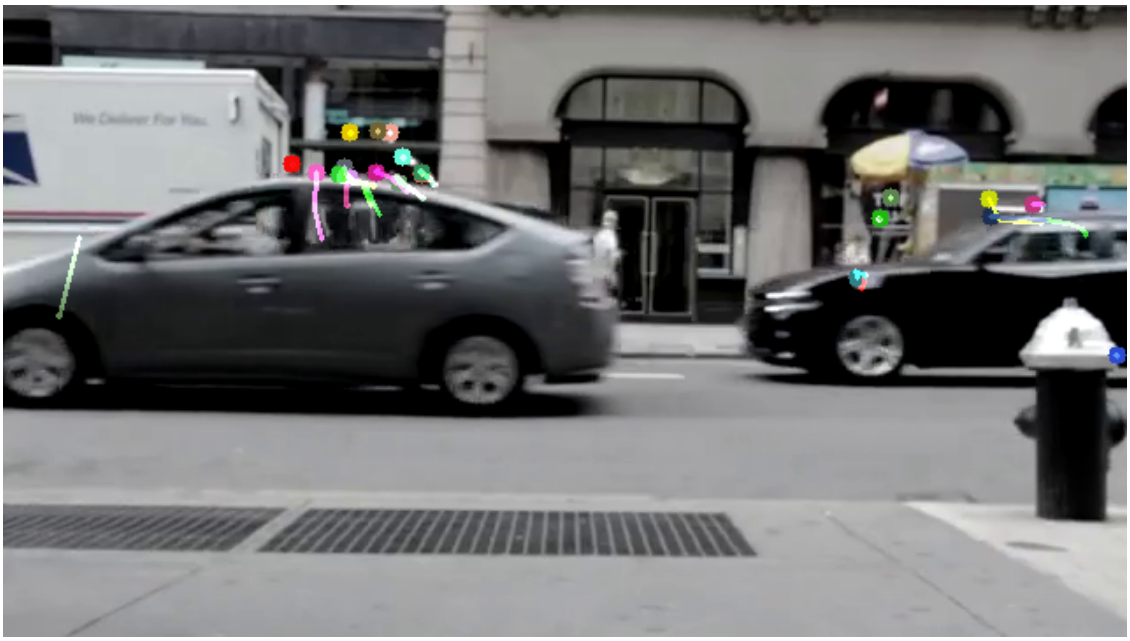
November 28, 2024

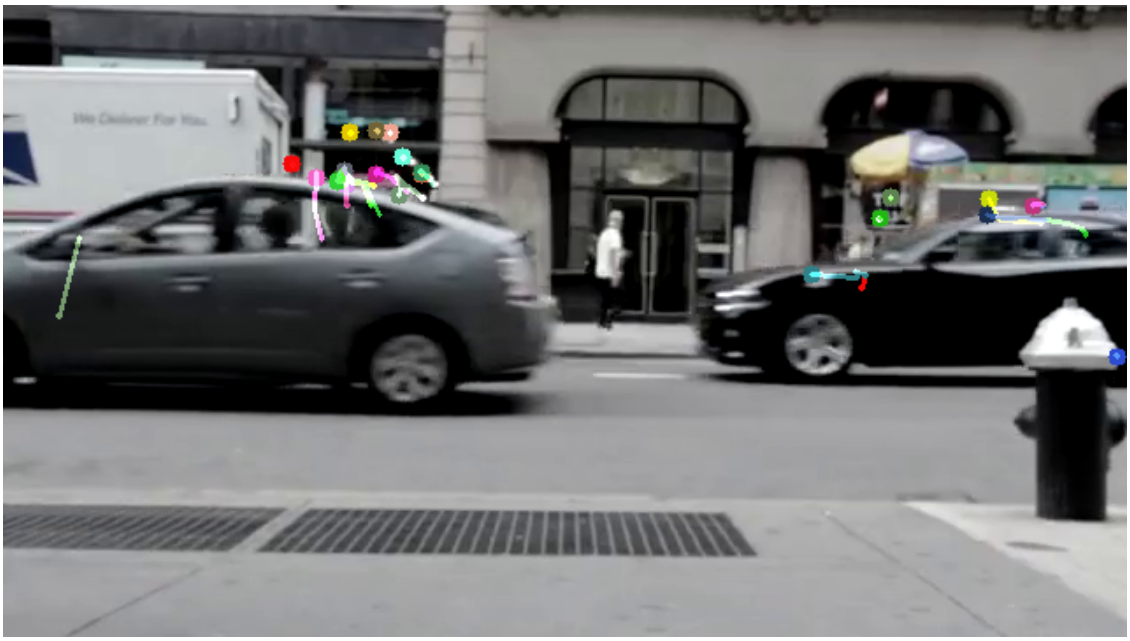
0.1 Lab Excerise 7:Optical Flow Estimation

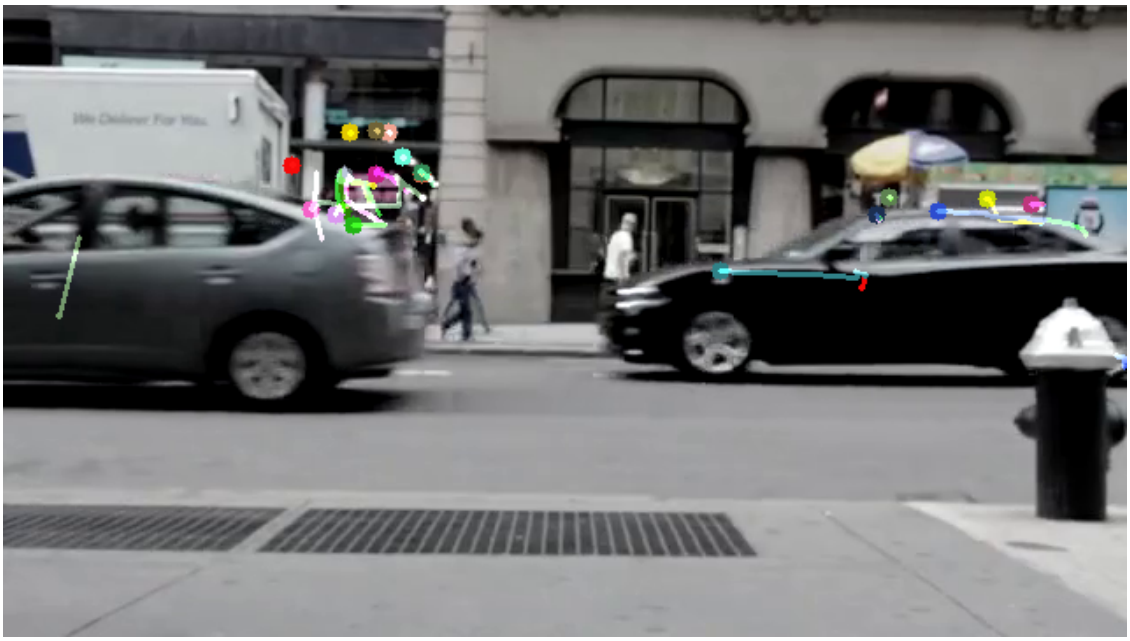
- **Objective:**Estimate optical flow between consecutive frames.
- **Task:** Implement optical flow algorithms (Horn-Schunck or Lucas-Kanade) to track the movement of objects between two consecutive frames in a video.

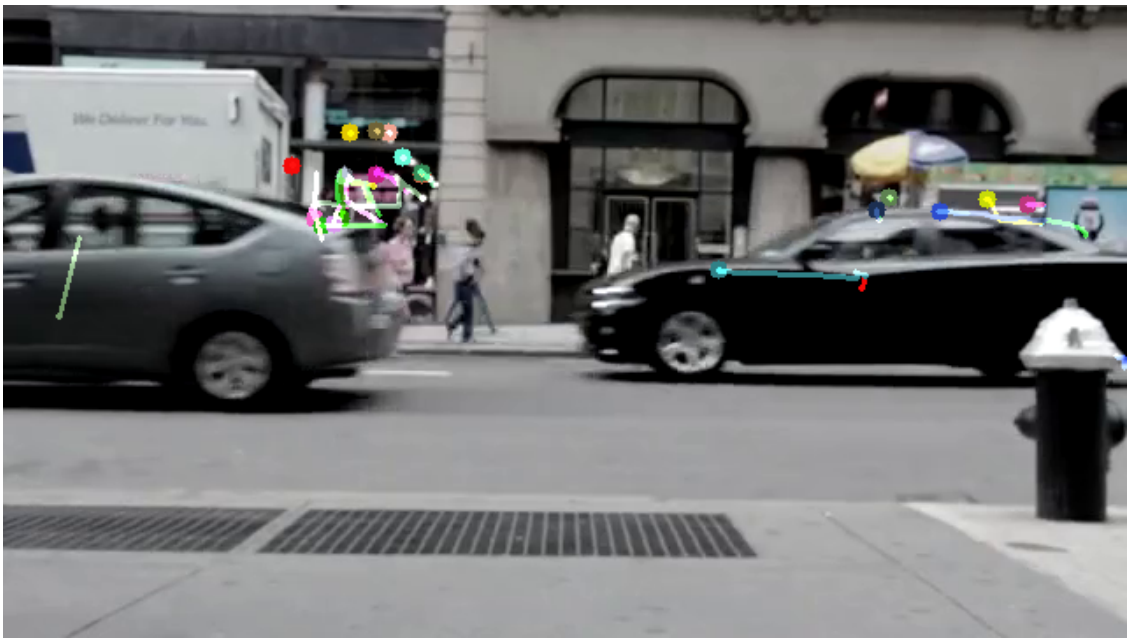
[1]:
```
import cv2
from google.colab.patches import cv2_imshow
import numpy as np

def lucas_kanade_method(video_path):
    cap = cv2.VideoCapture(video_path)
    # Parameters 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 True:
        ret, frame = cap.read()
        if not ret:
            break
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Calculate optical flow
        p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p0, None,␣
 ↪**lk_params)
```

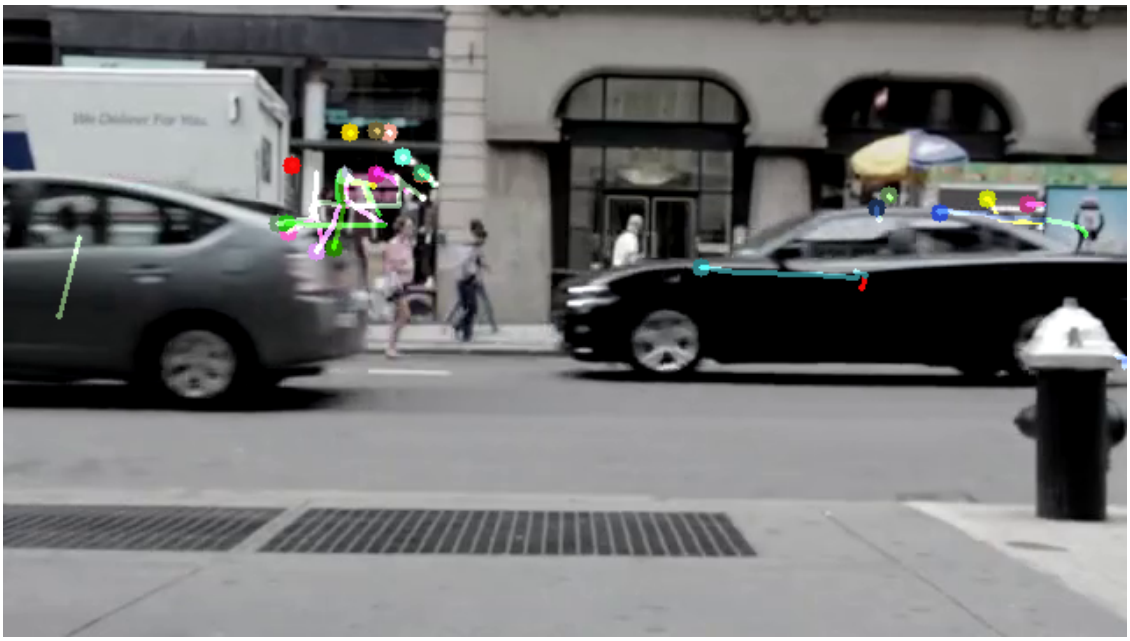
```
        # Select good points
        good_new = p1[st == 1]
        good_old = p0[st == 1]

        # Draw the tracks
        for i, (new, old) in enumerate(zip(good_new, good_old)):
            a, b = new.ravel()
            c, d = old.ravel()
             # Convert coordinates to integers
            a, b = int(a), int(b)
            c, d = int(c), int(d)
            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(img)

        k = cv2.waitKey(25) & 0xFF
        if k == 27:  # ESC key to stop
            break
        if k == ord("c"):  # 'c' key to clear the mask
            mask = np.zeros_like(old_frame)

        # Update the previous frame and previous points
        old_gray = frame_gray.copy()
        p0 = good_new.reshape(-1, 1, 2)

    # Release the capture and close windows
    cap.release()
    cv2.destroyAllWindows()

# Call the function with the path to your video
lucas_kanade_method("people.mp4")
```

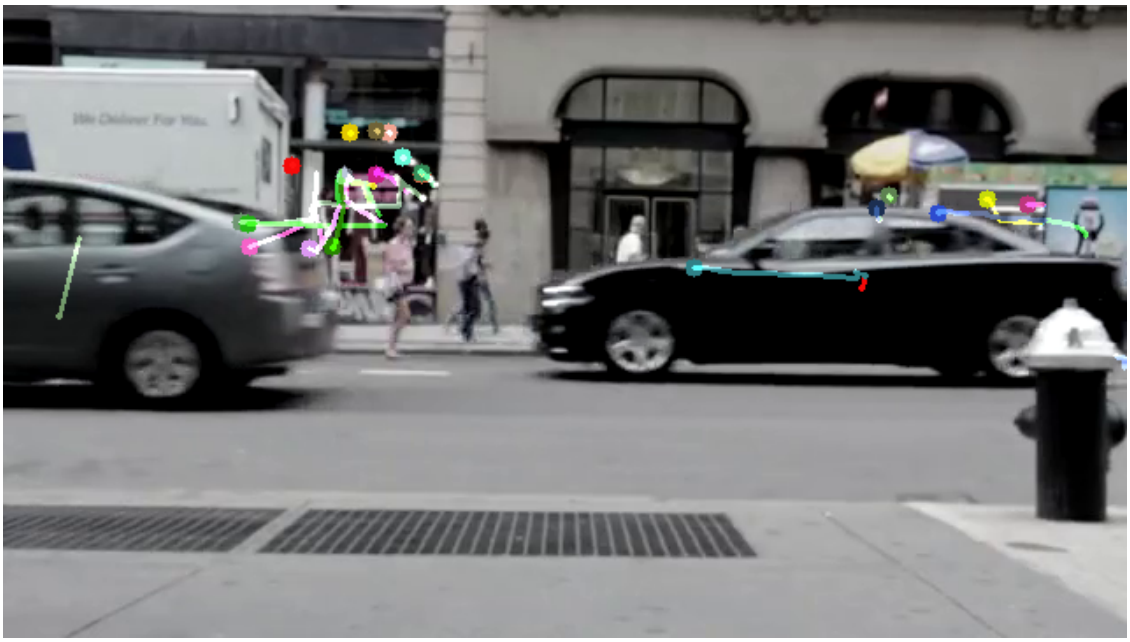


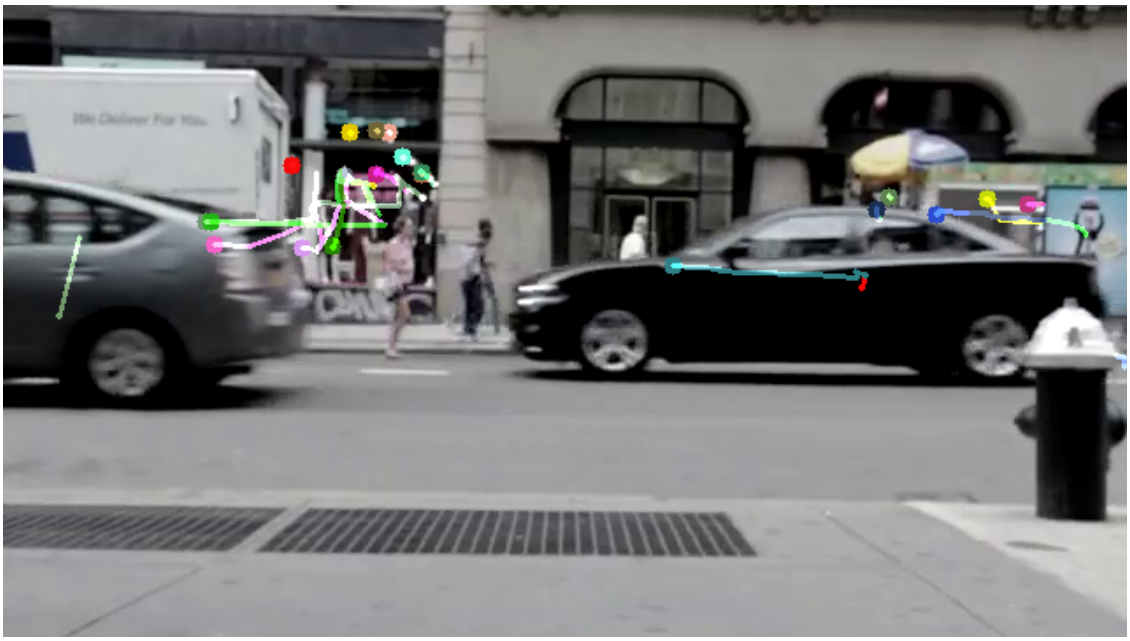


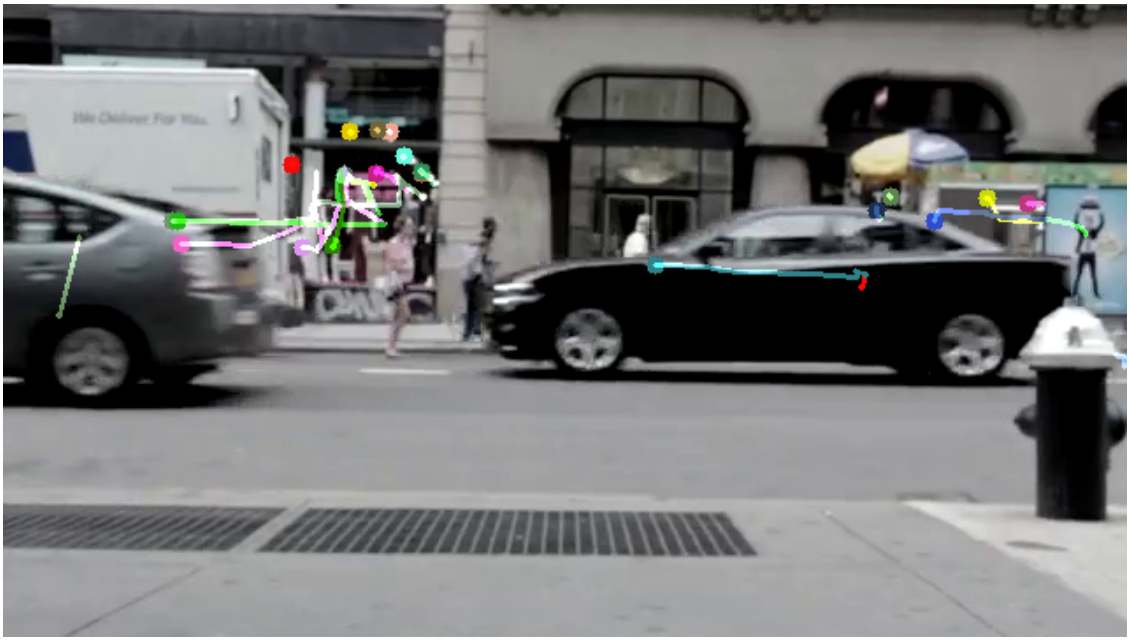


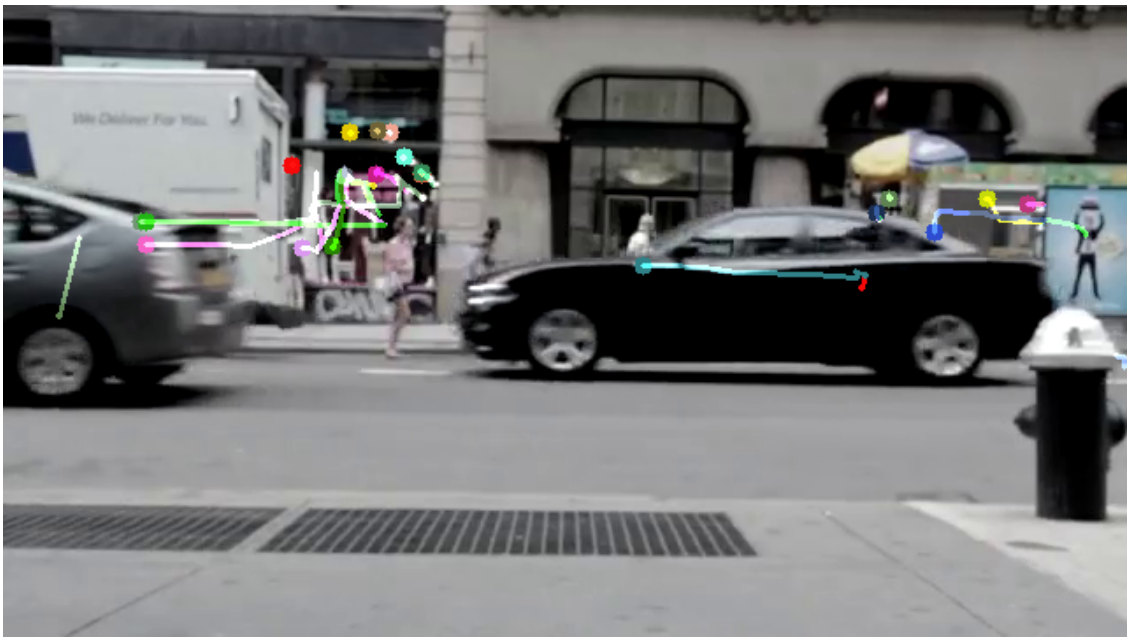


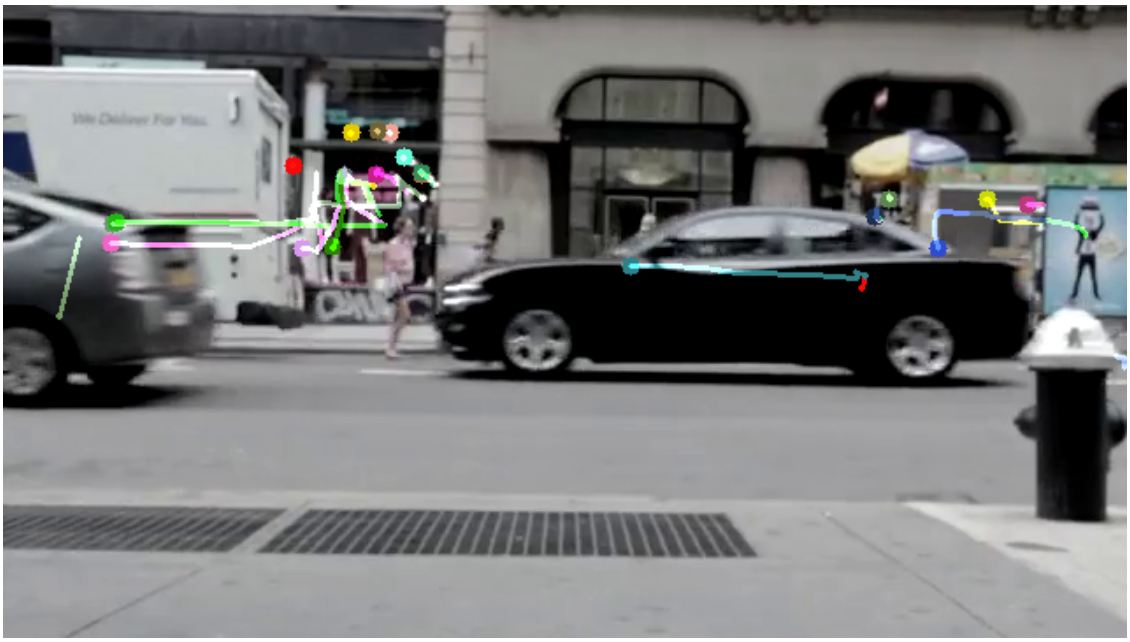


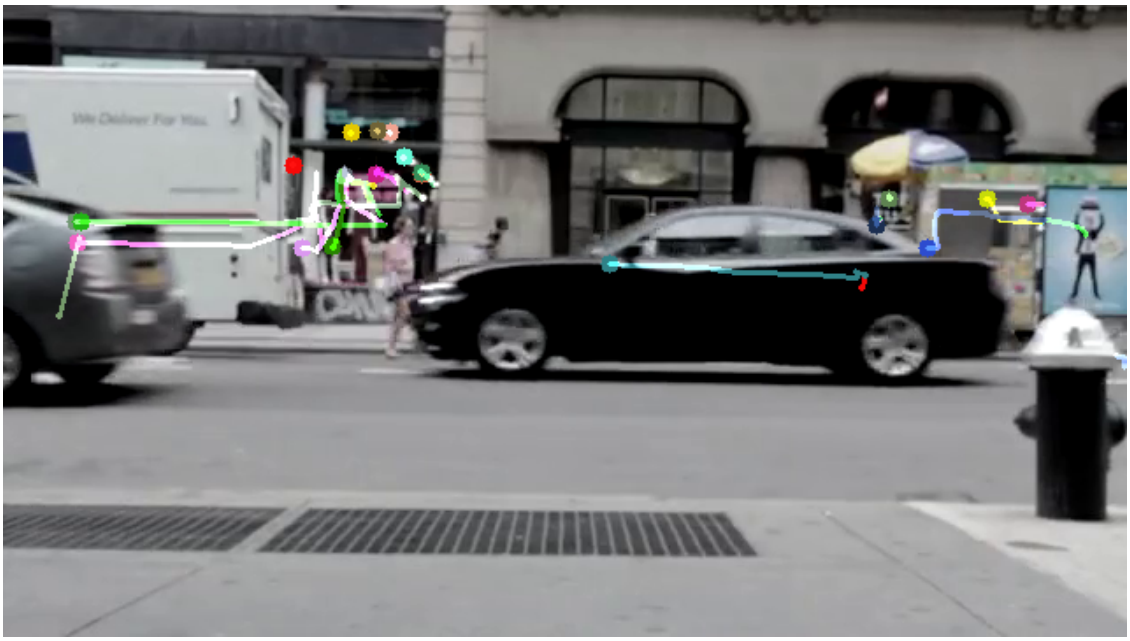


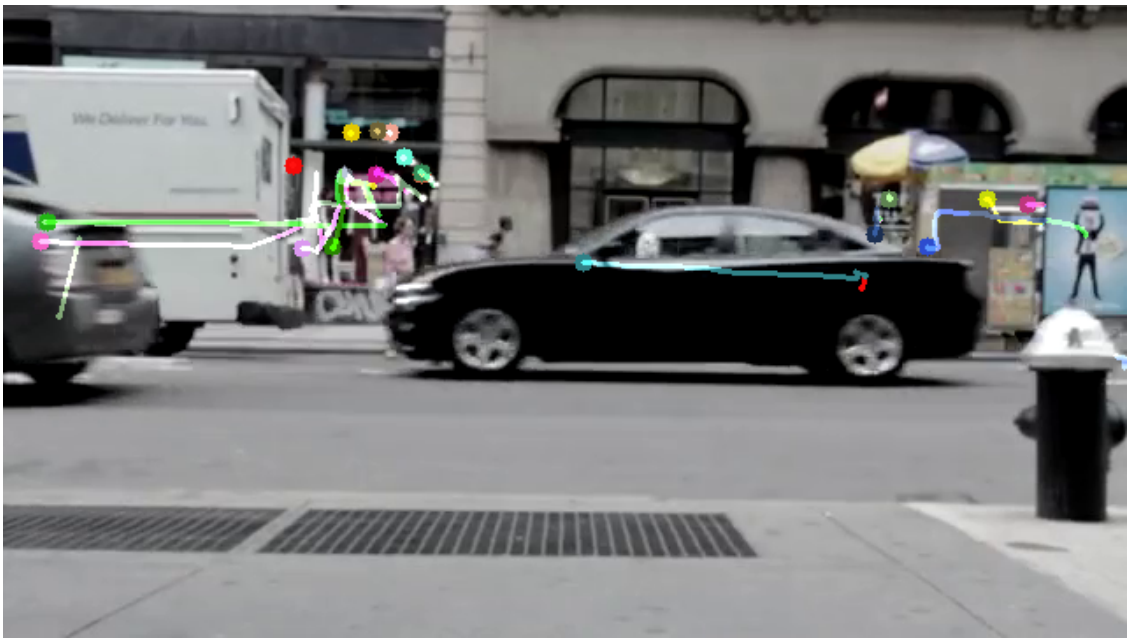


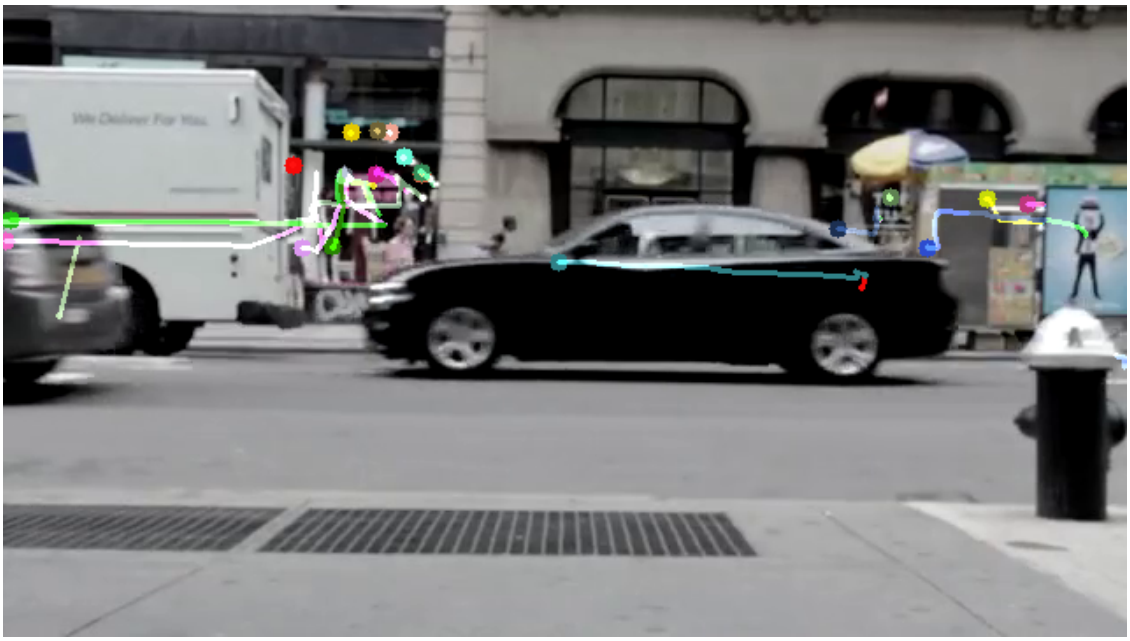


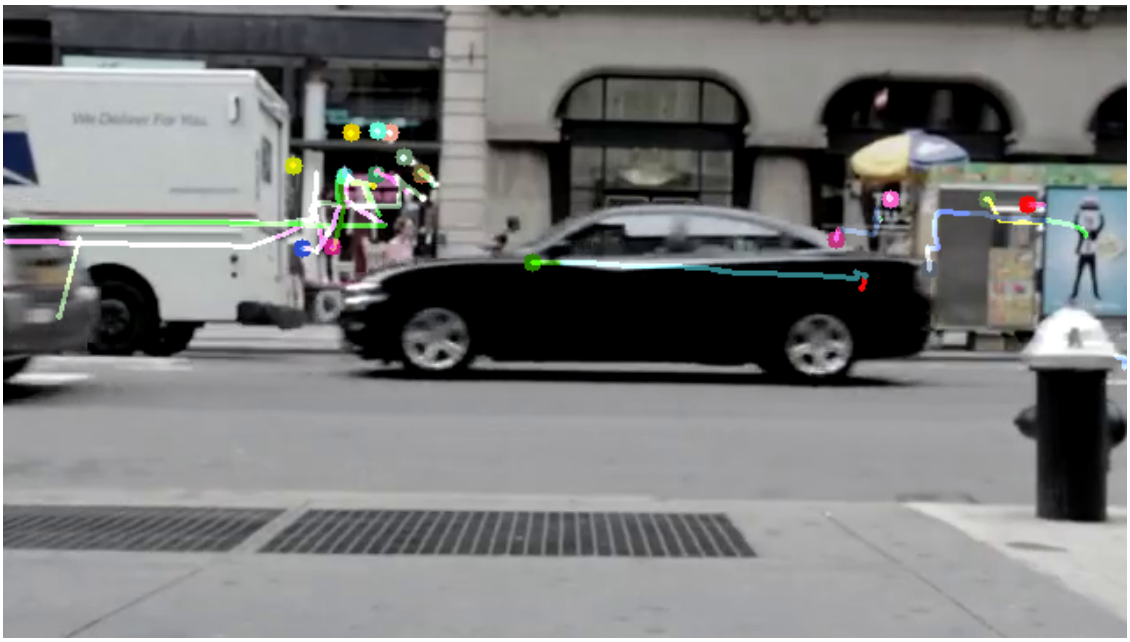


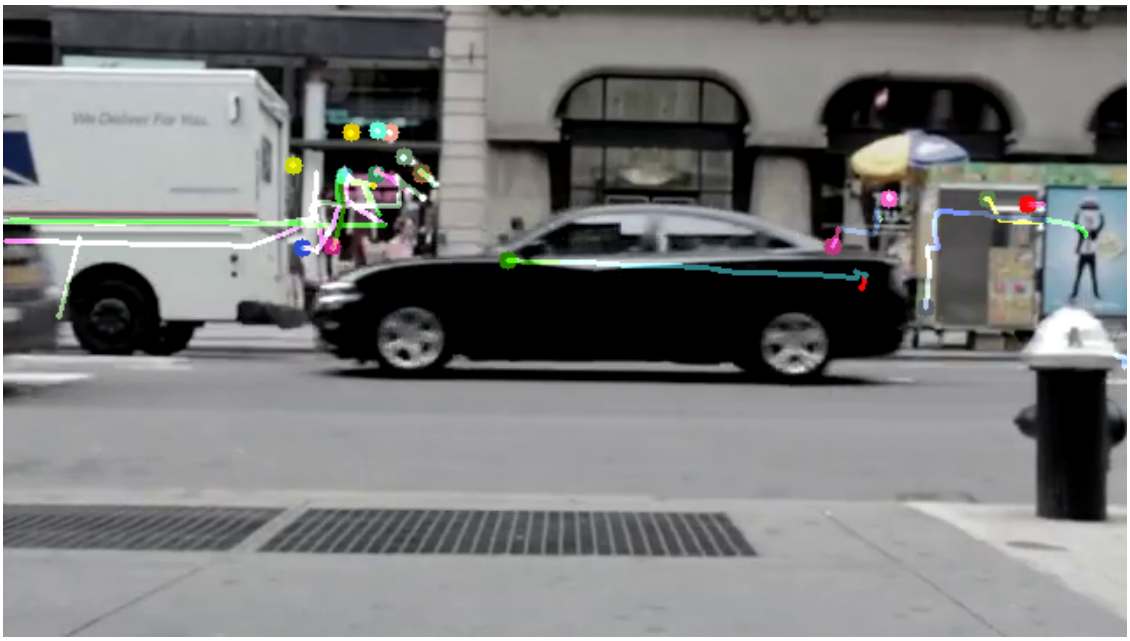


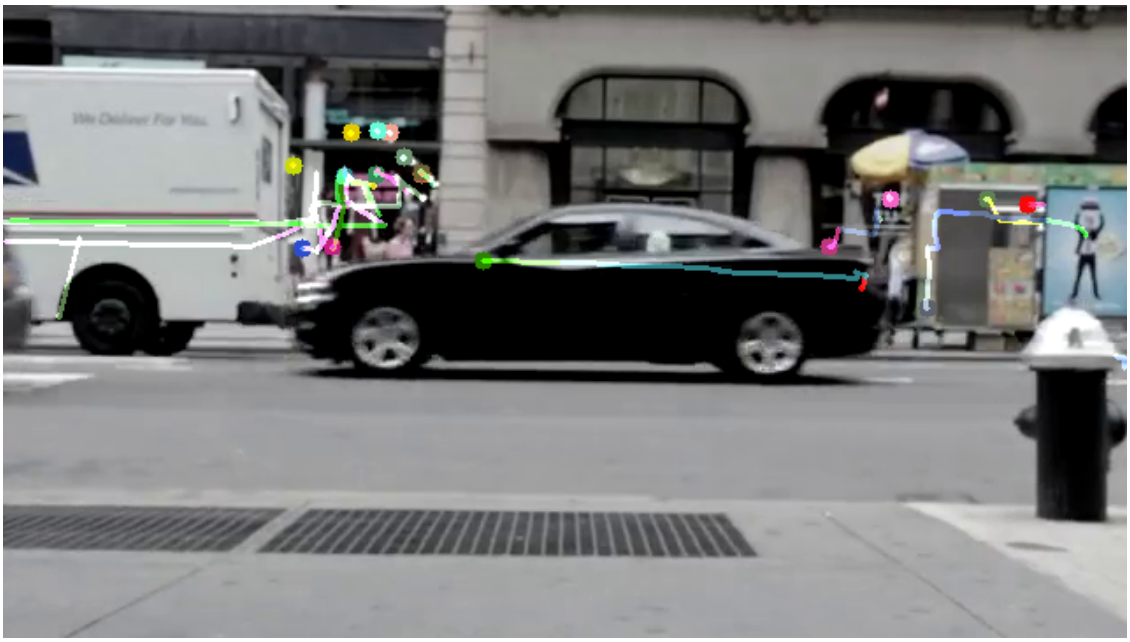


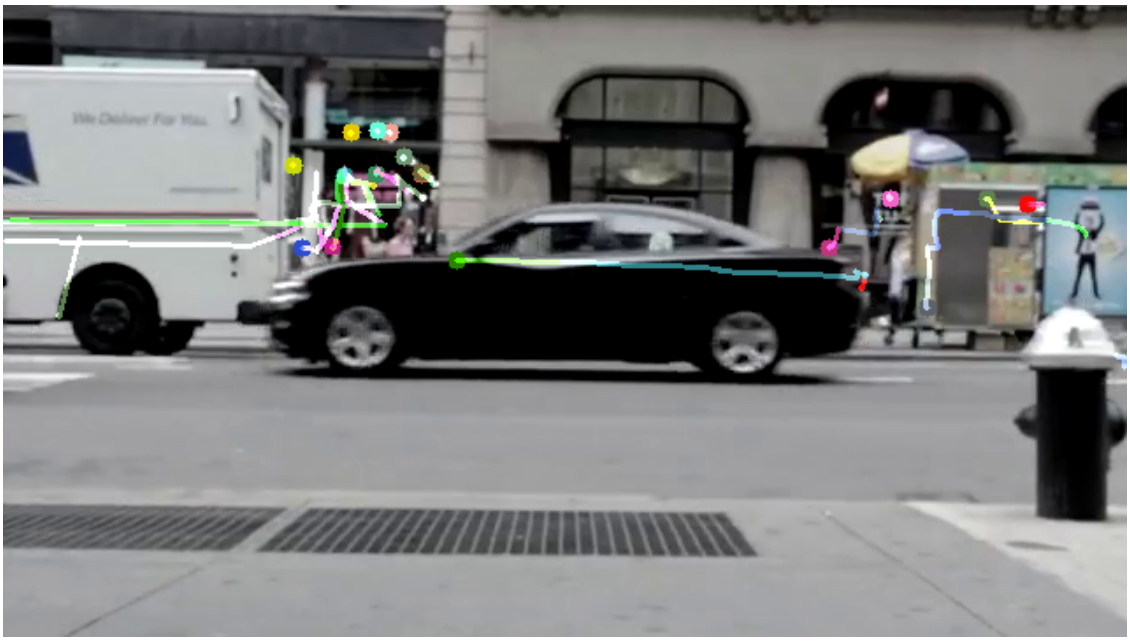


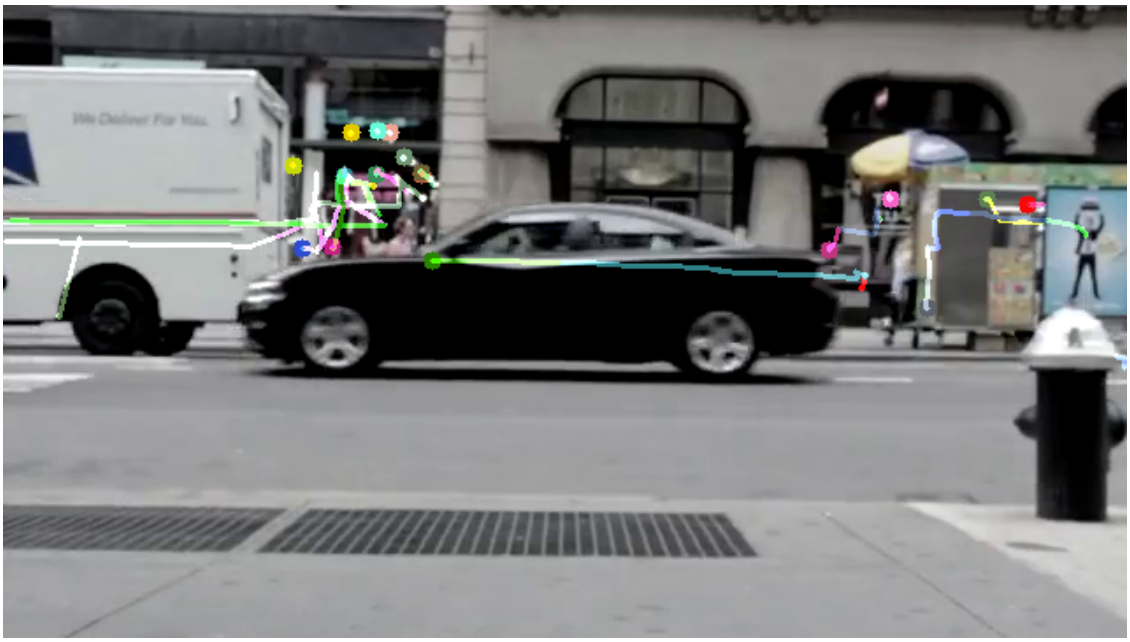


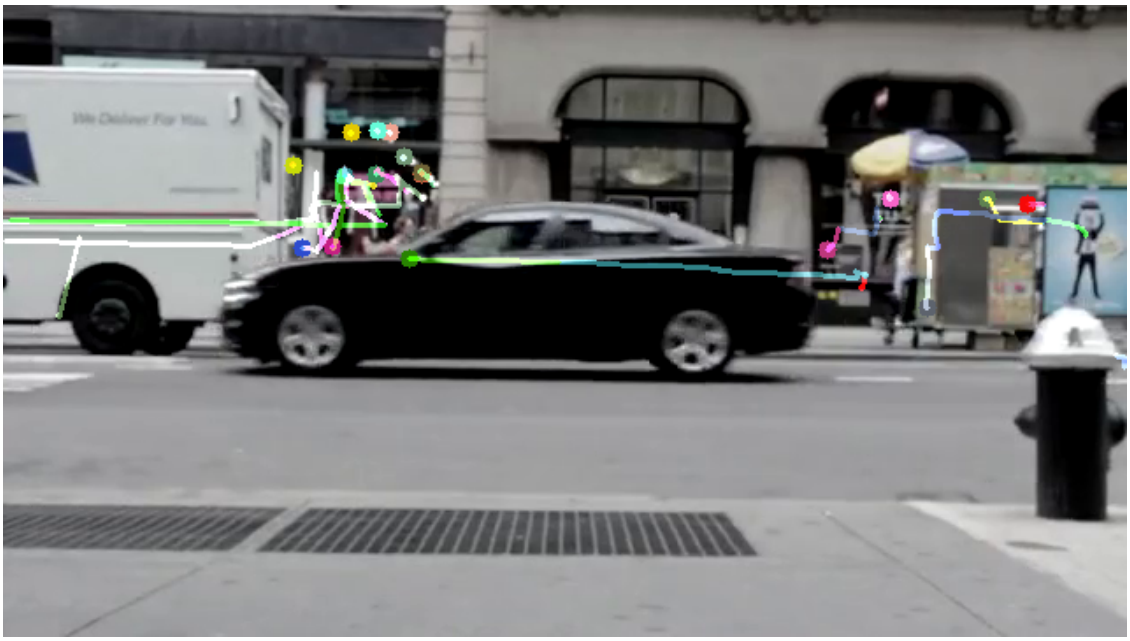


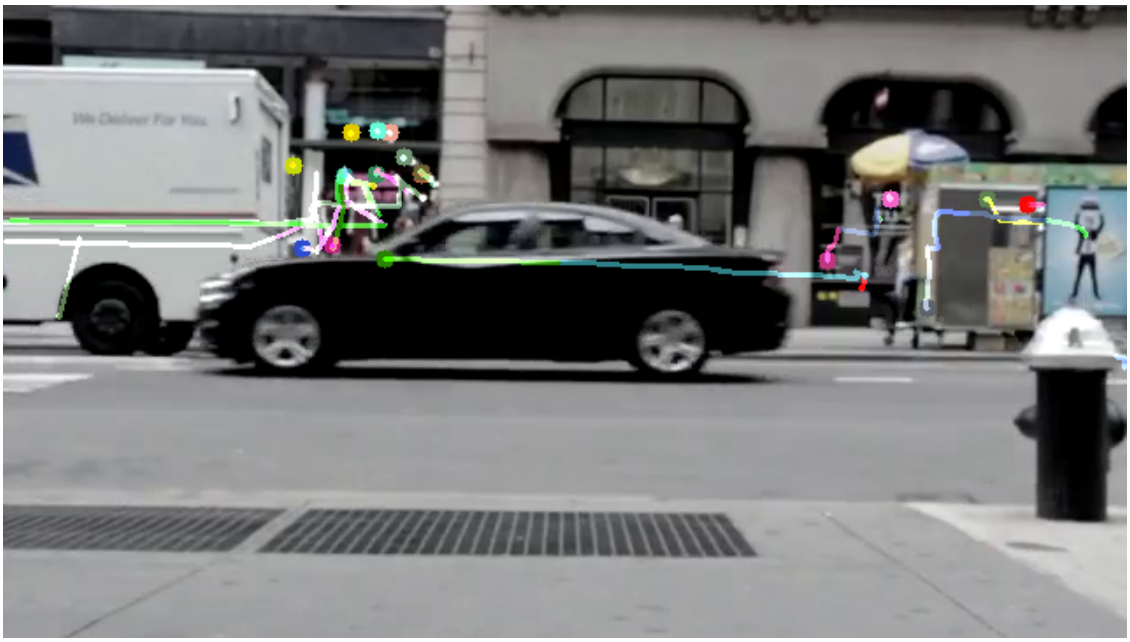


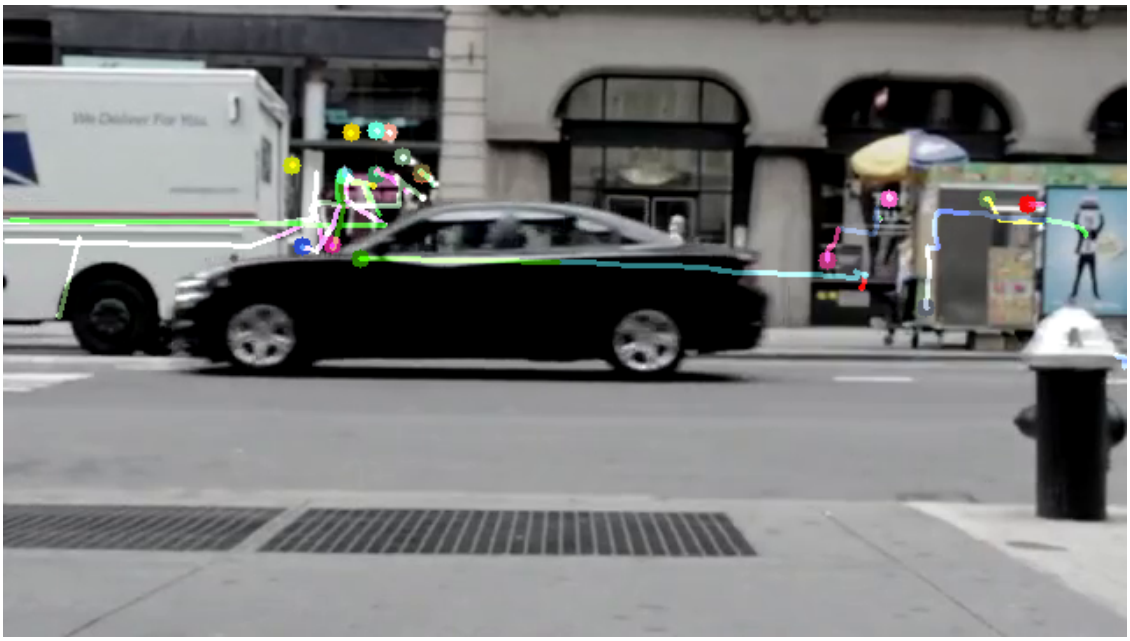


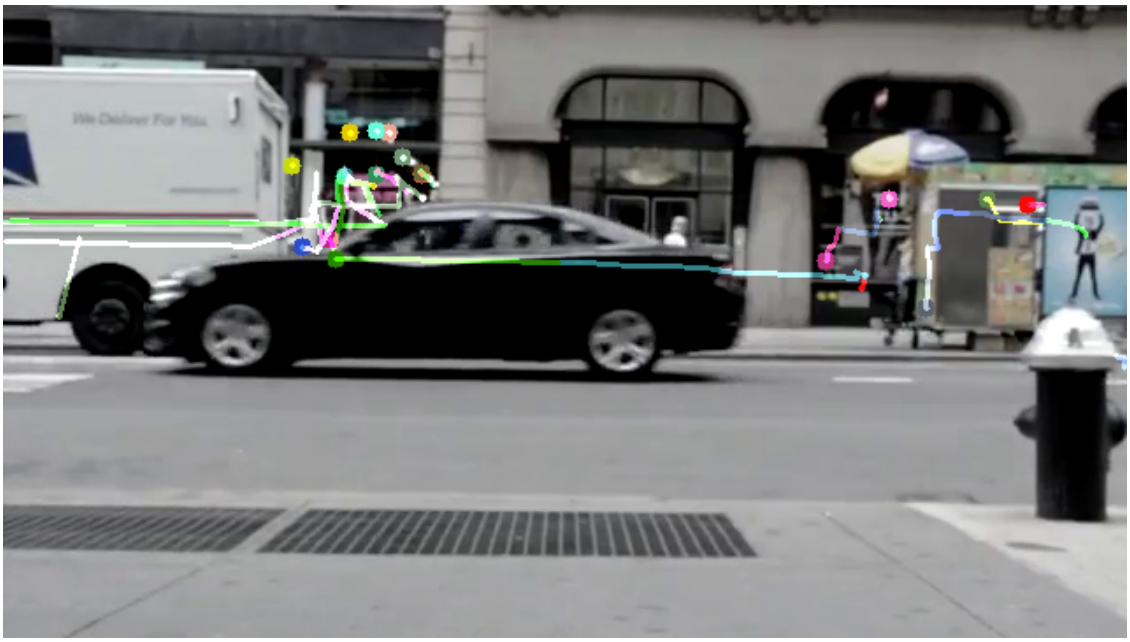


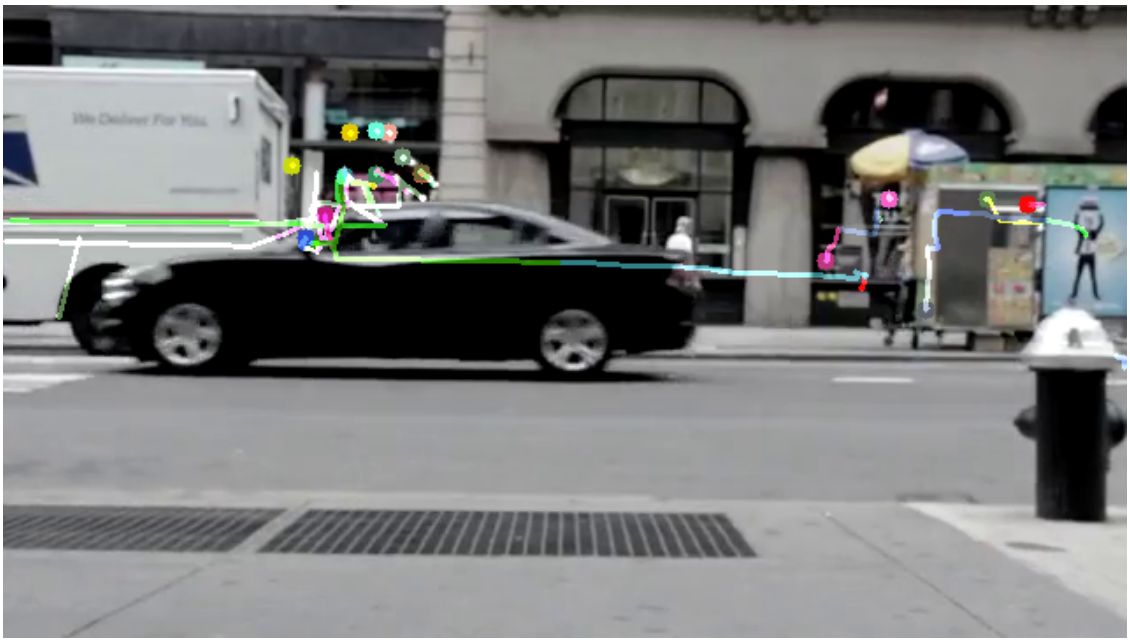


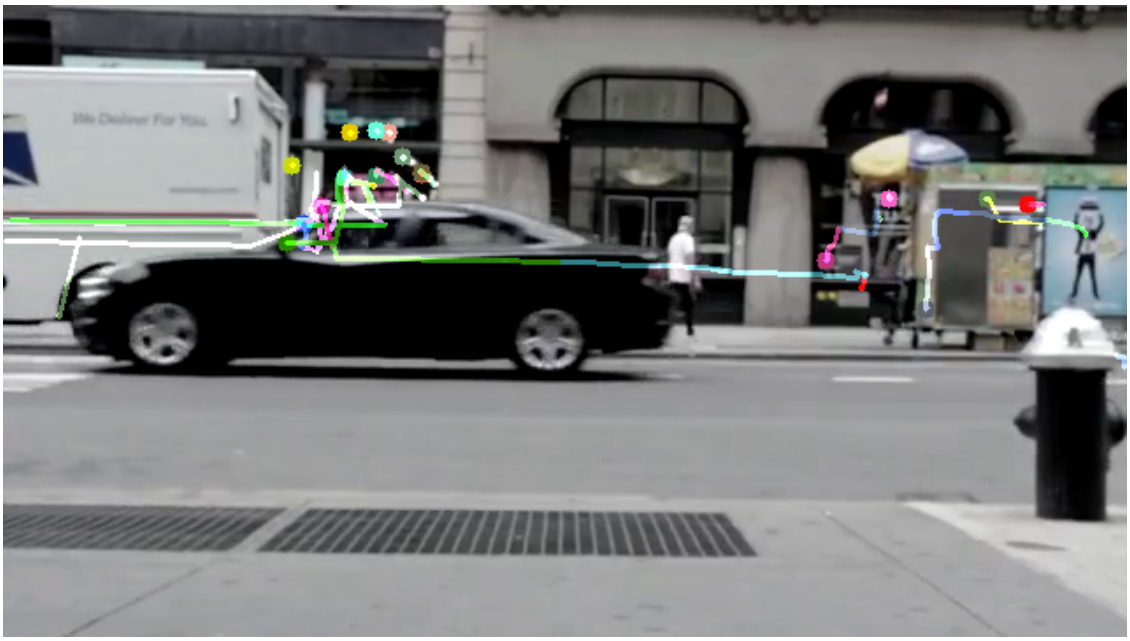


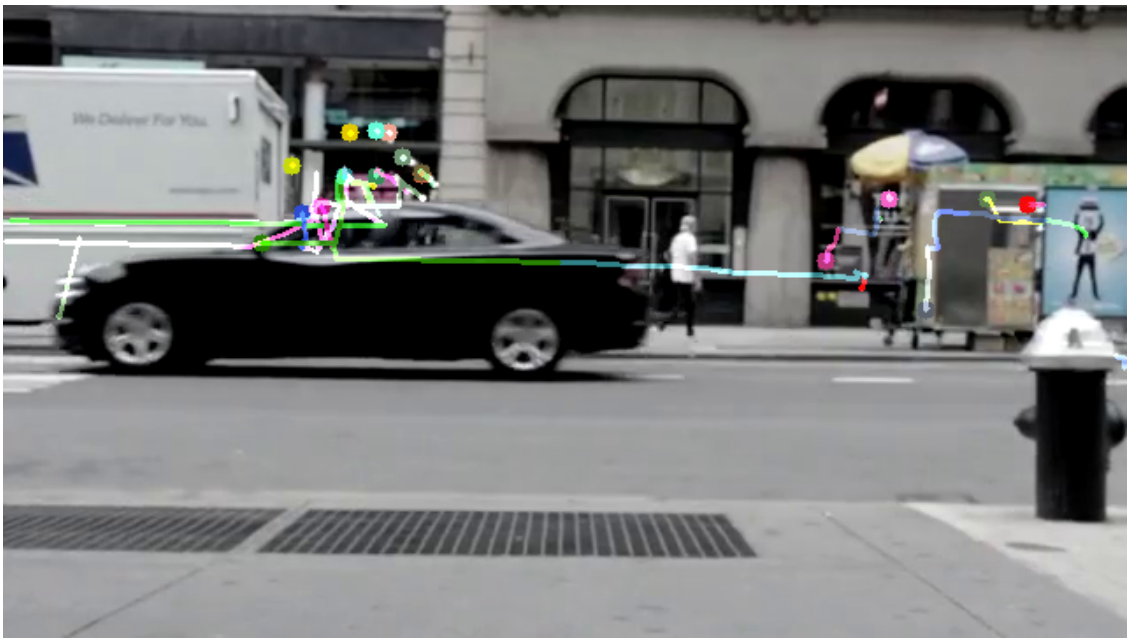


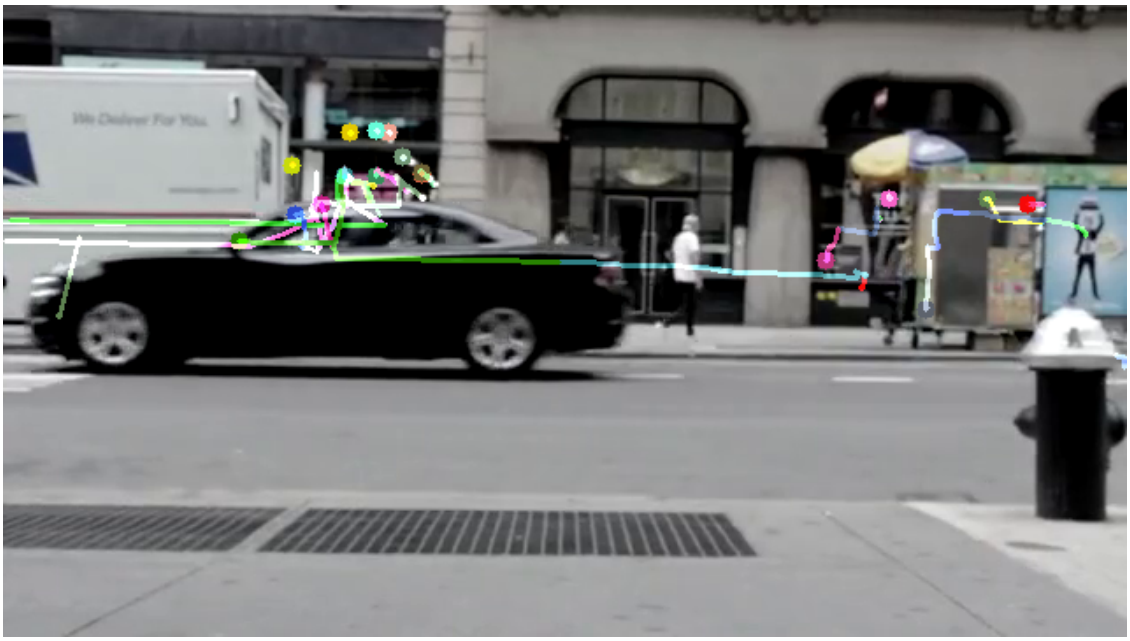


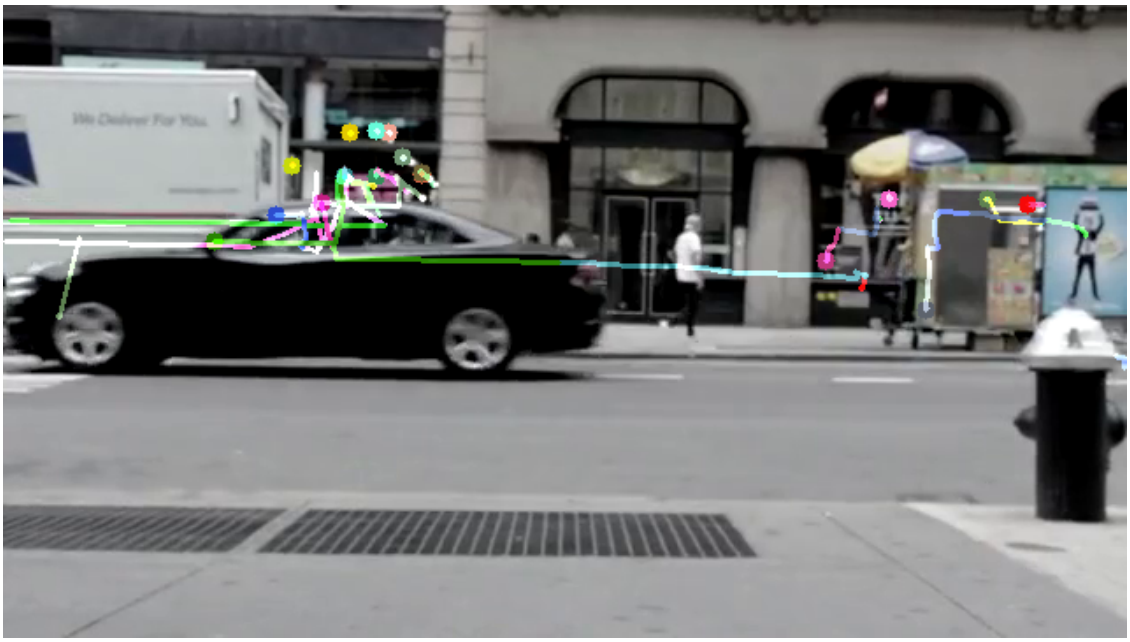


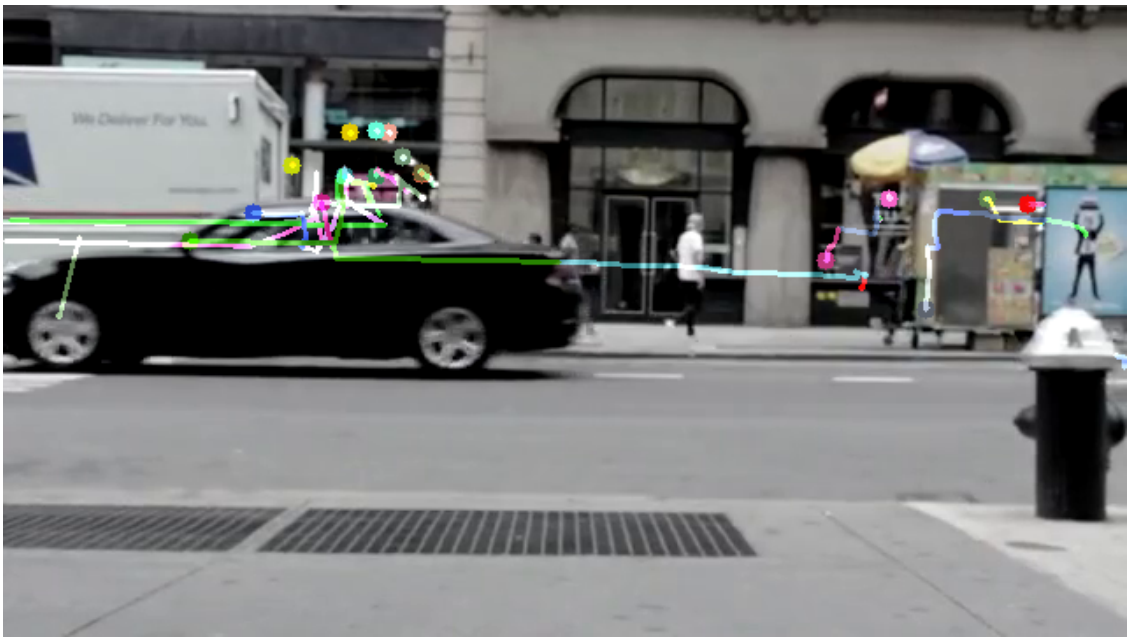


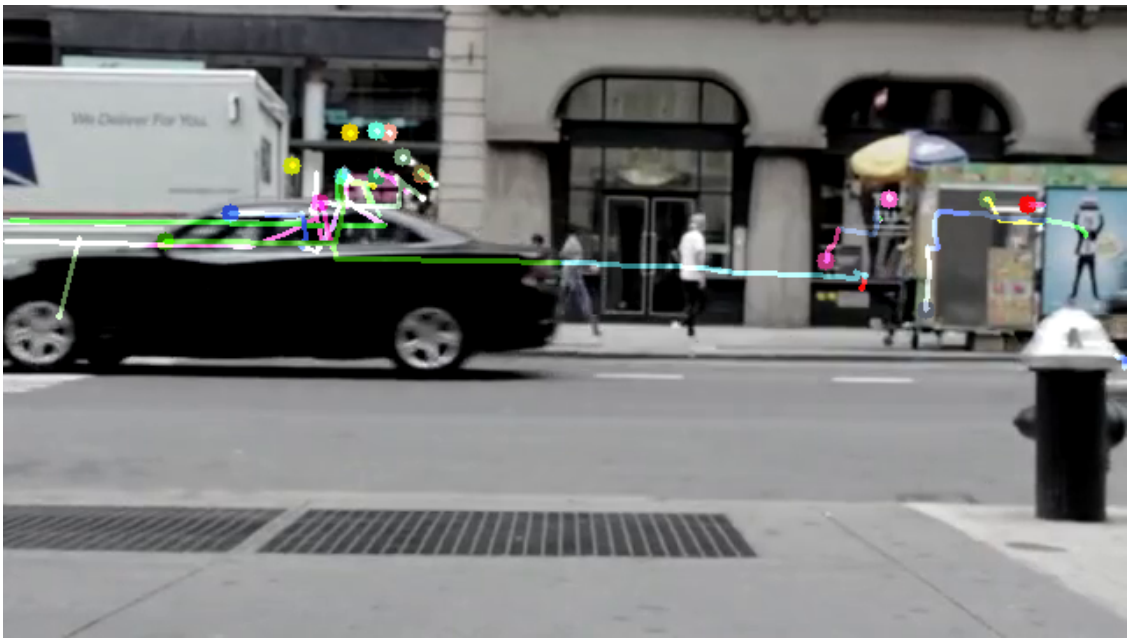


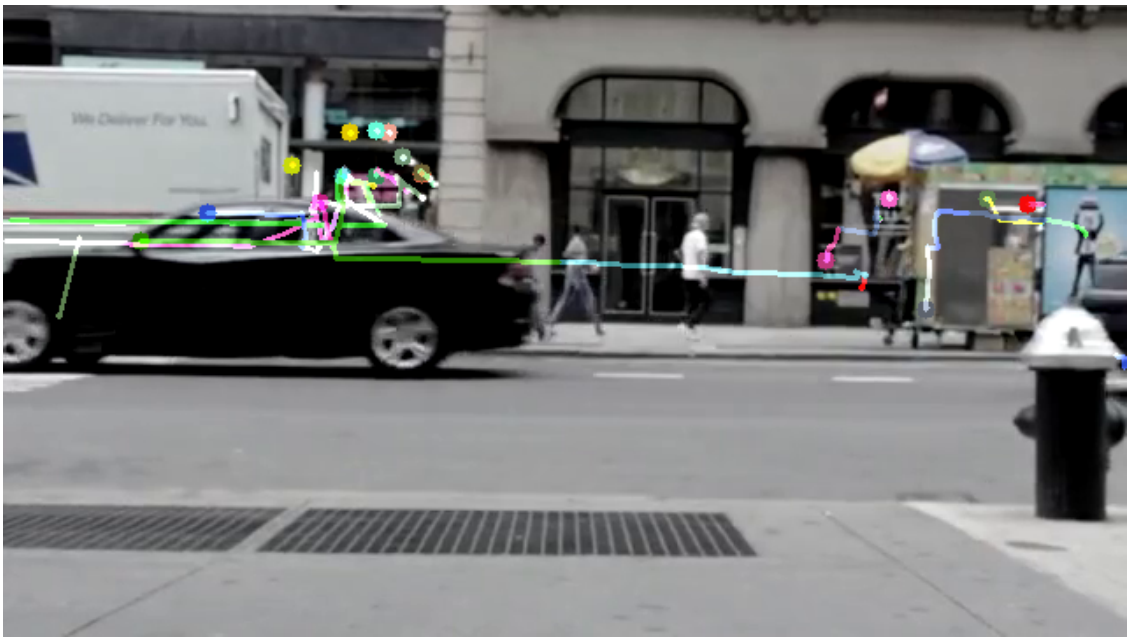


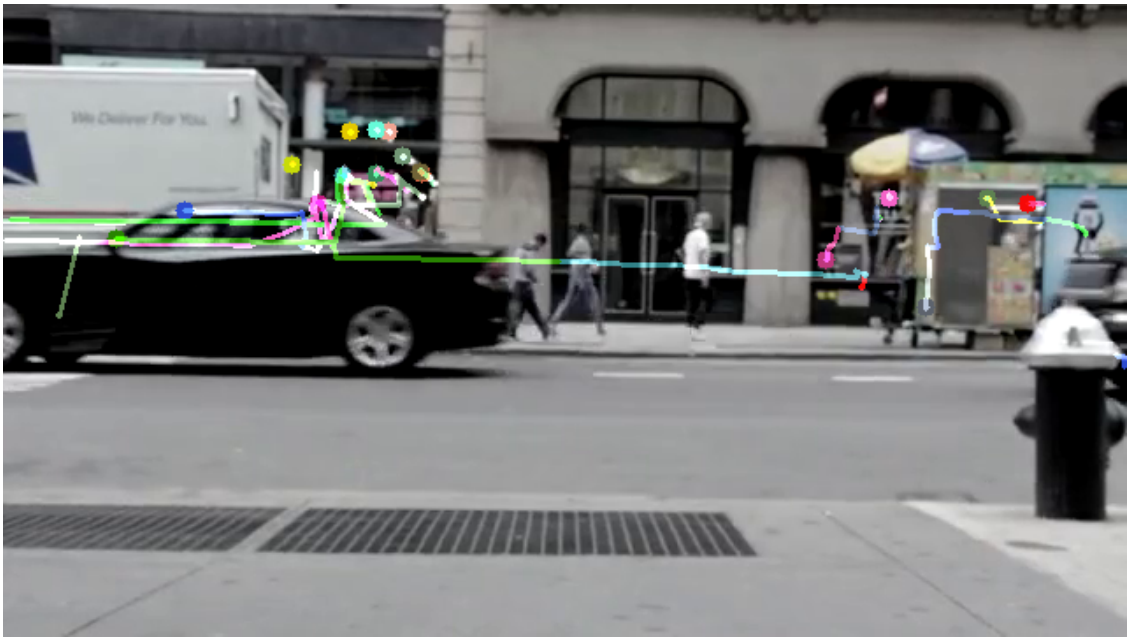


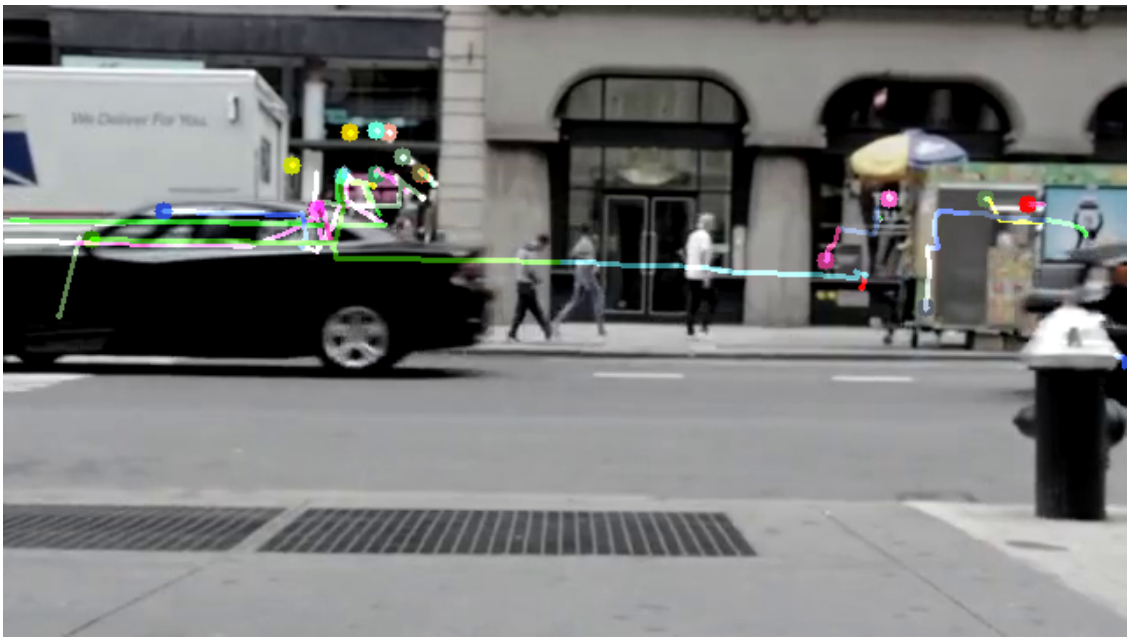


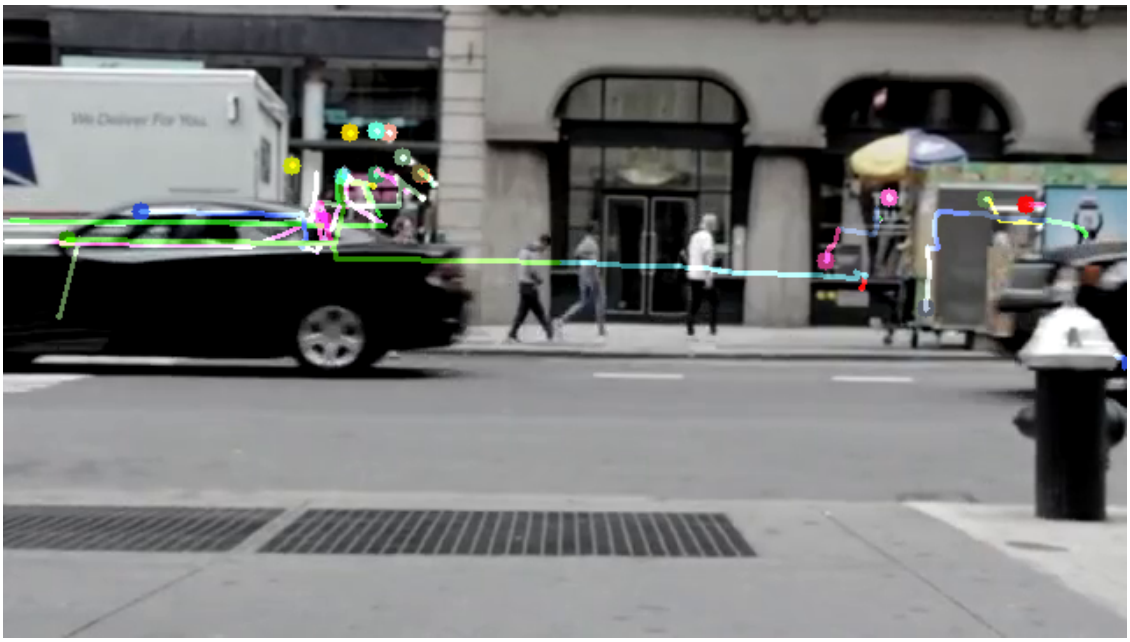


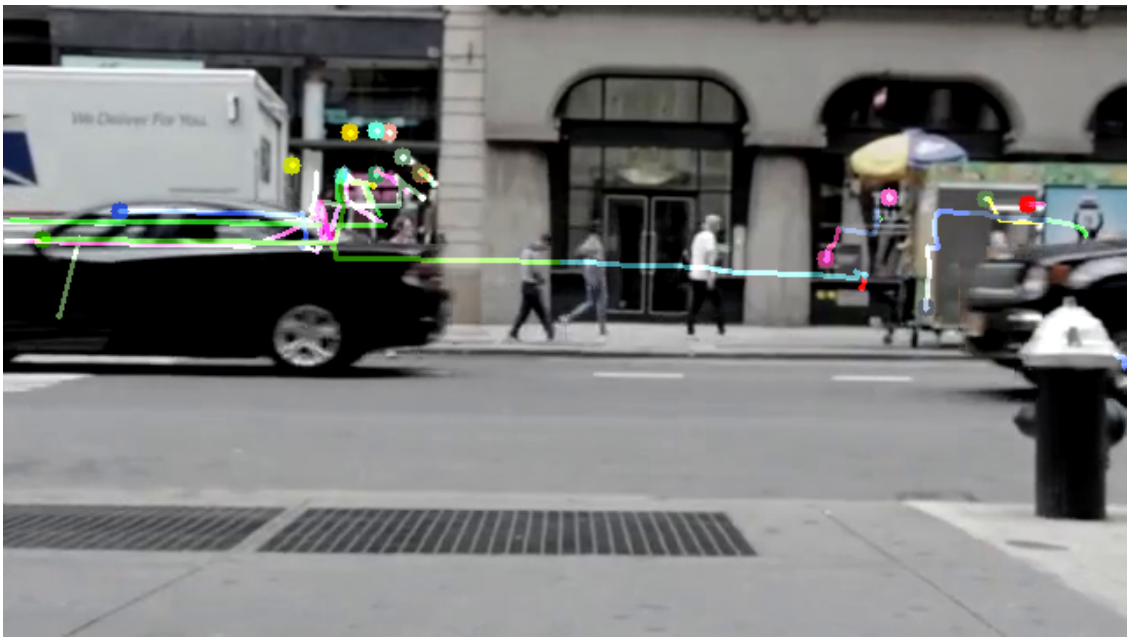


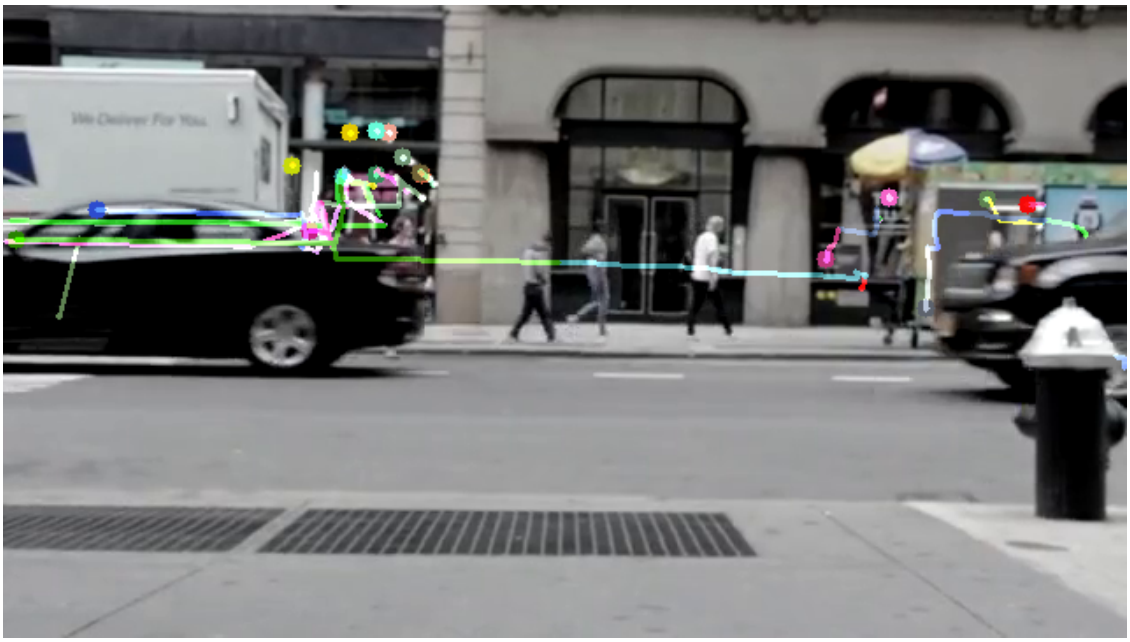


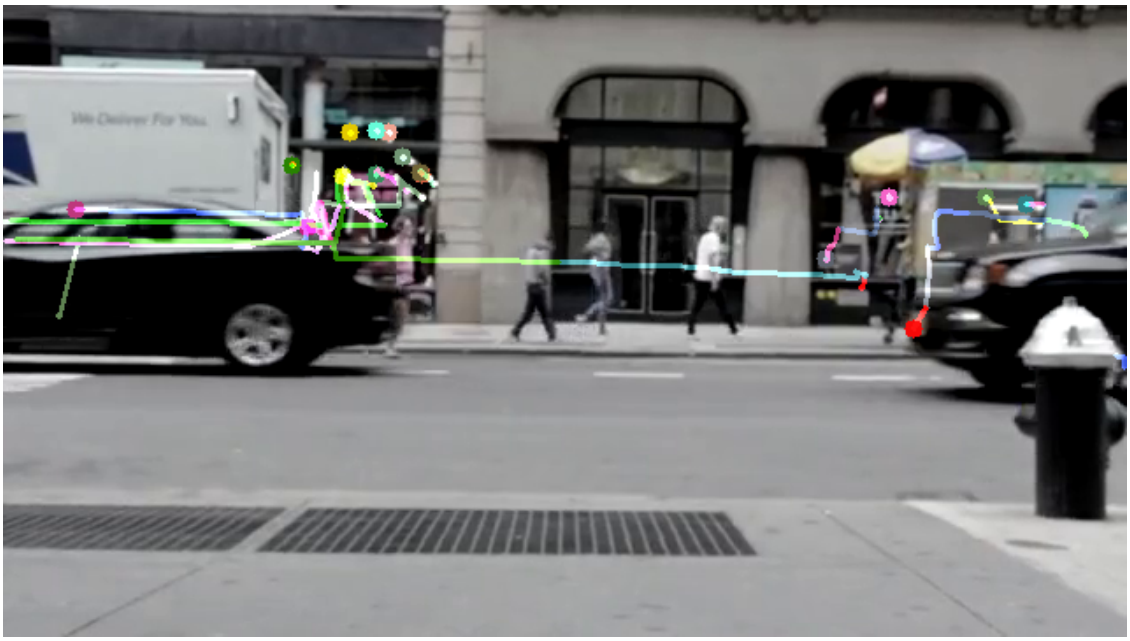


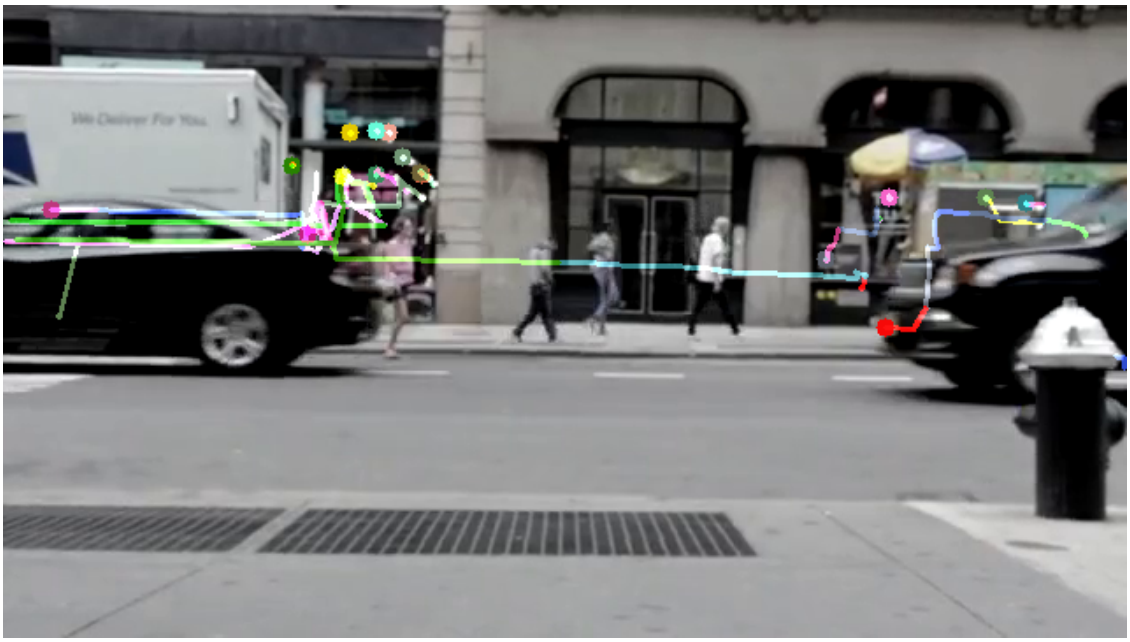


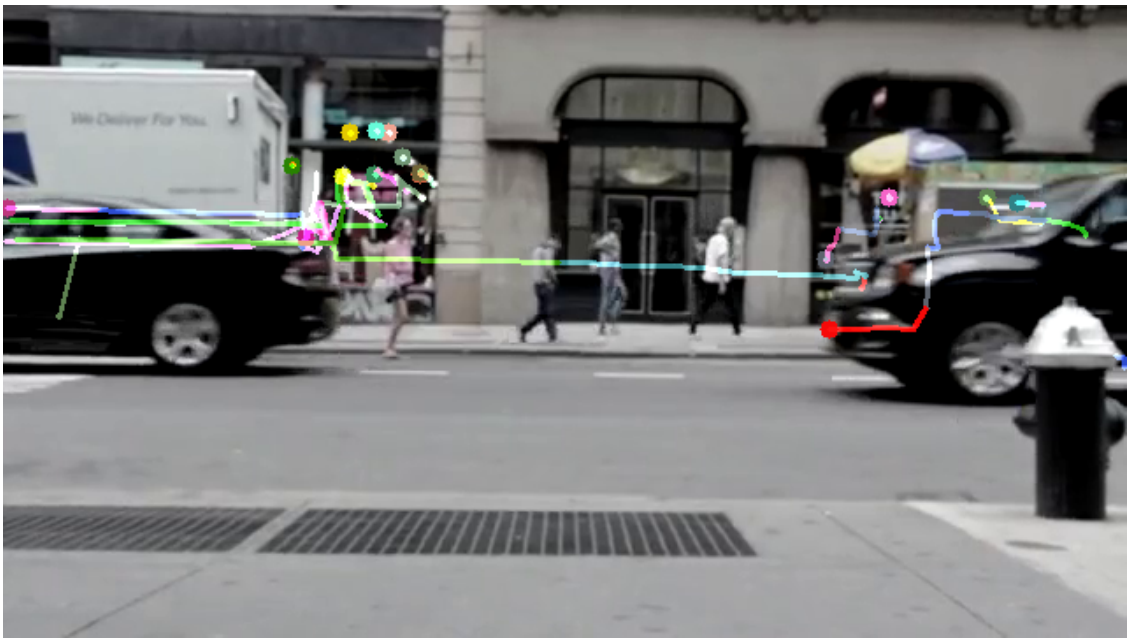


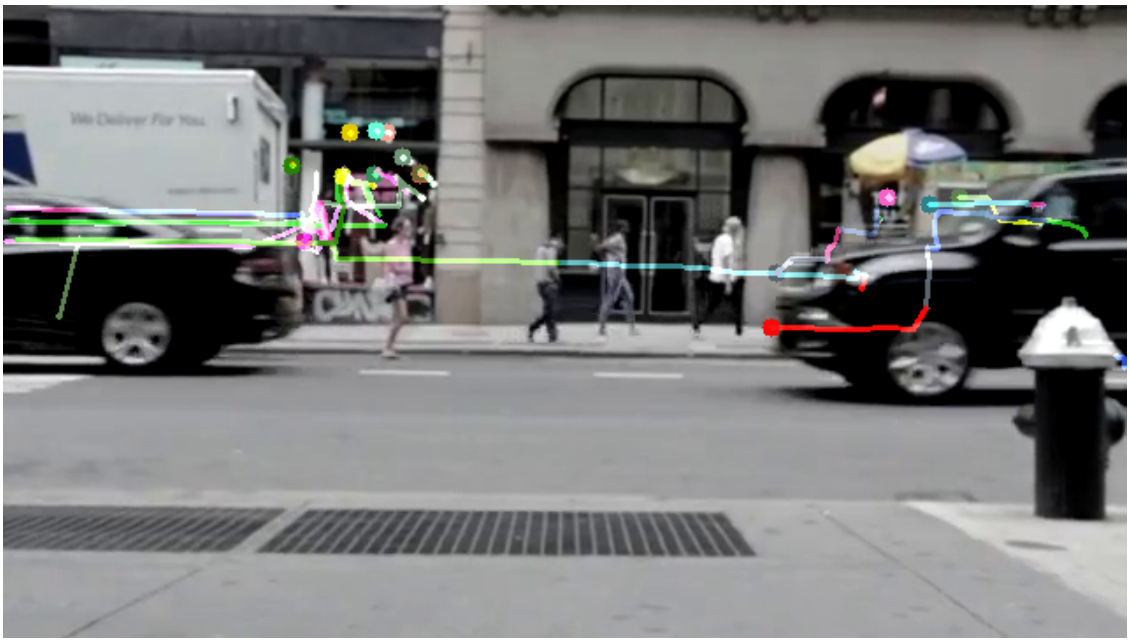


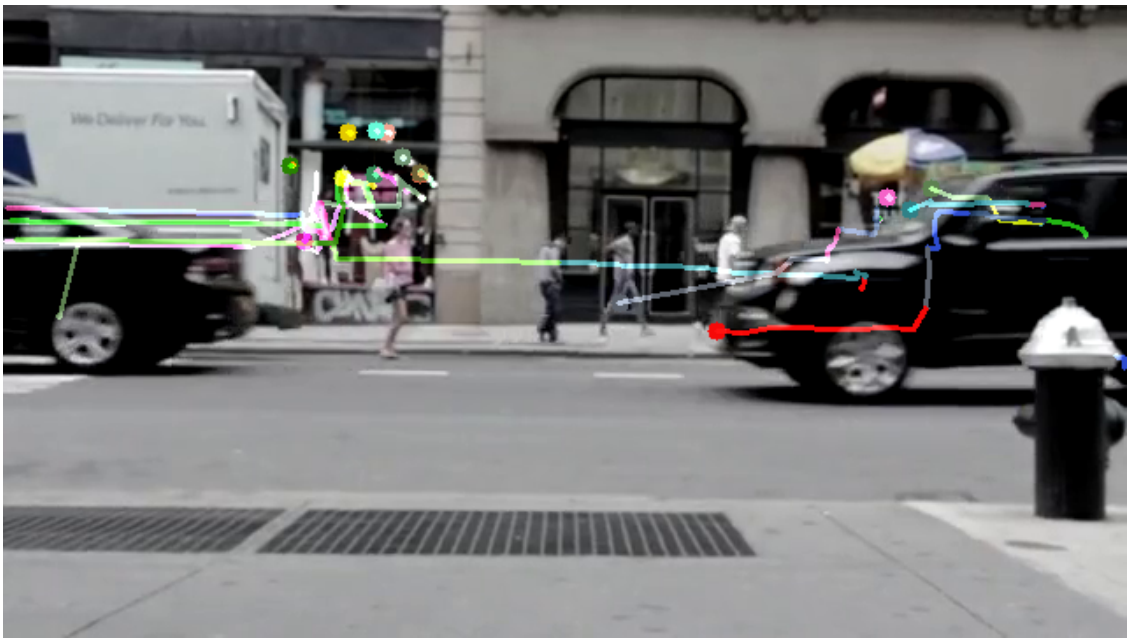


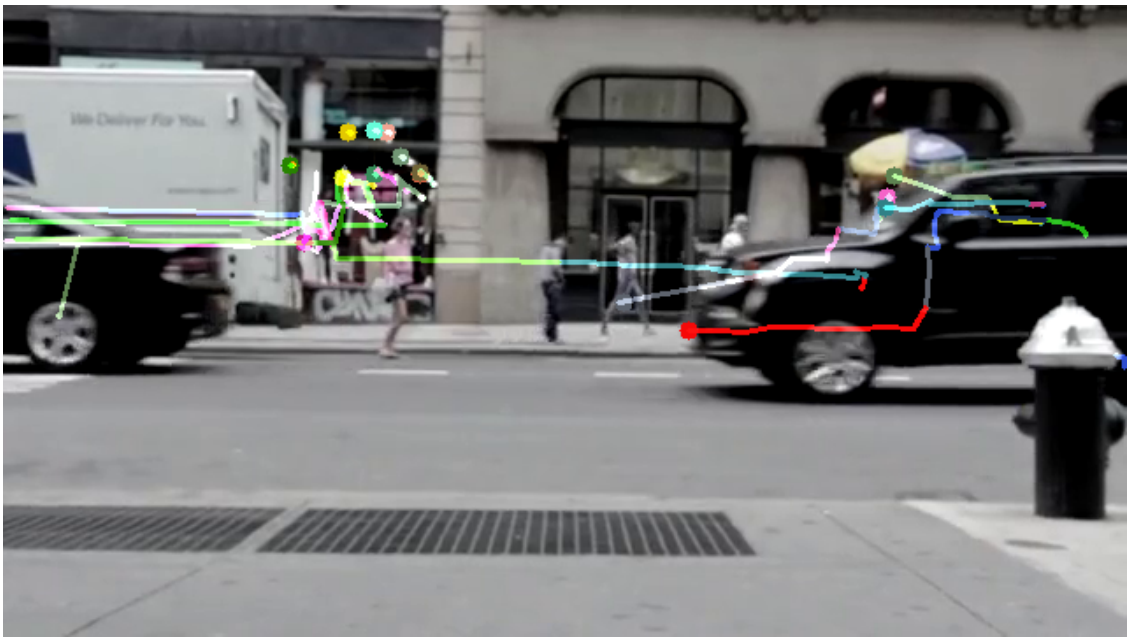


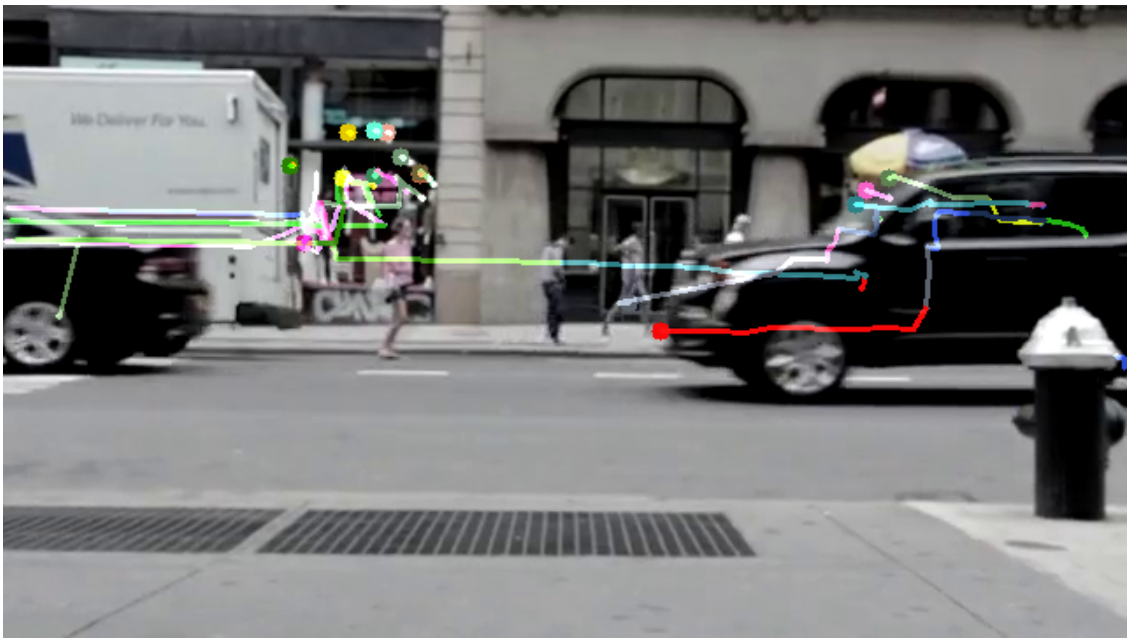


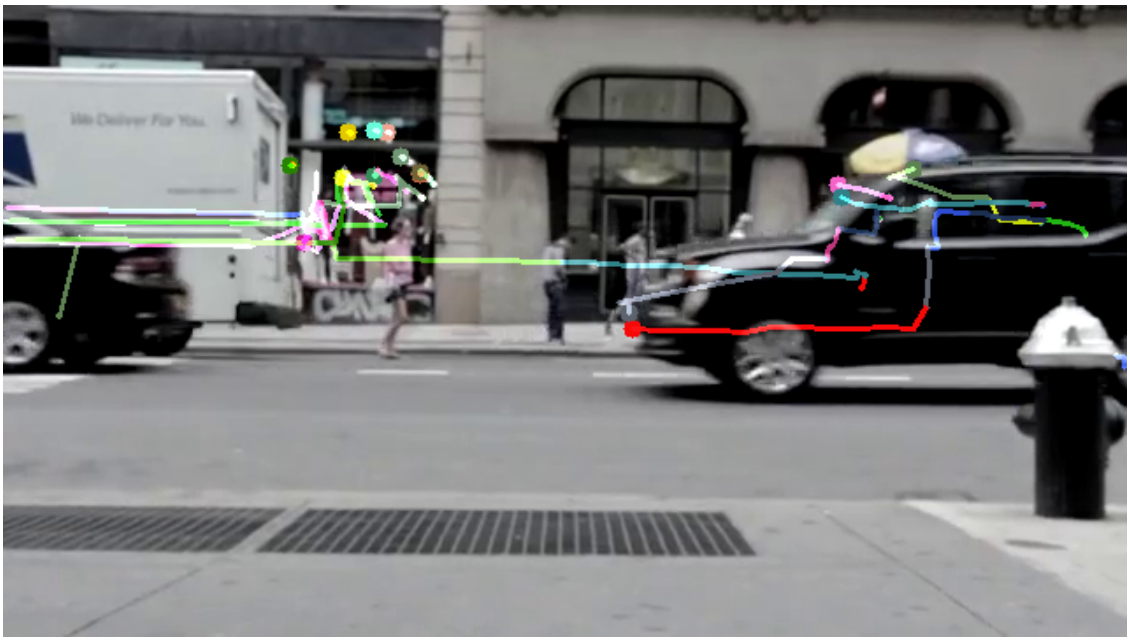


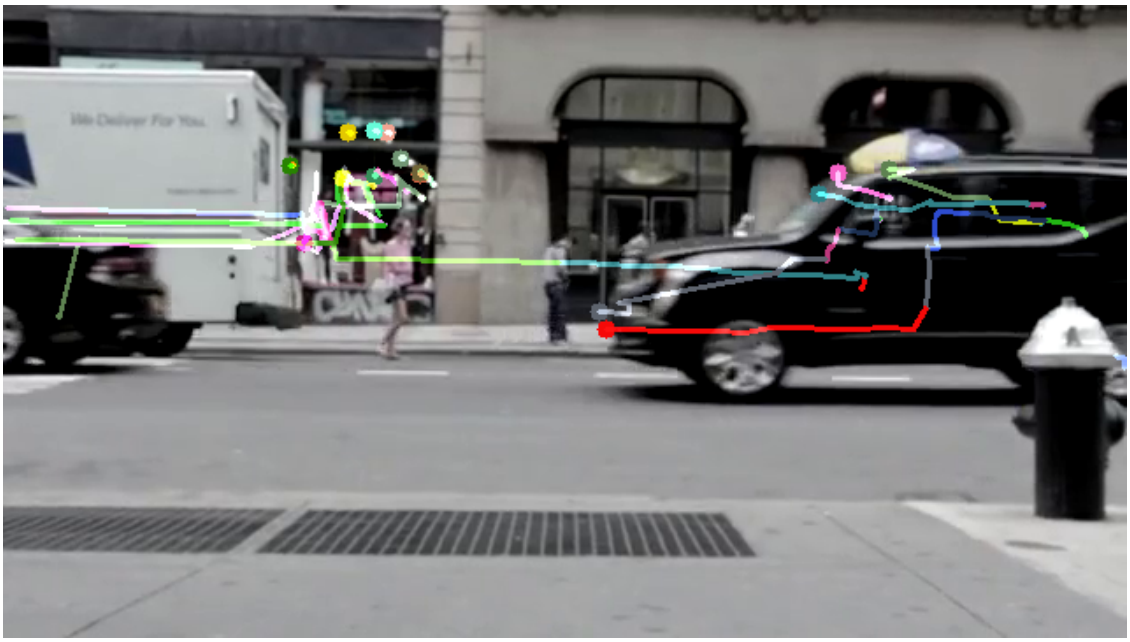


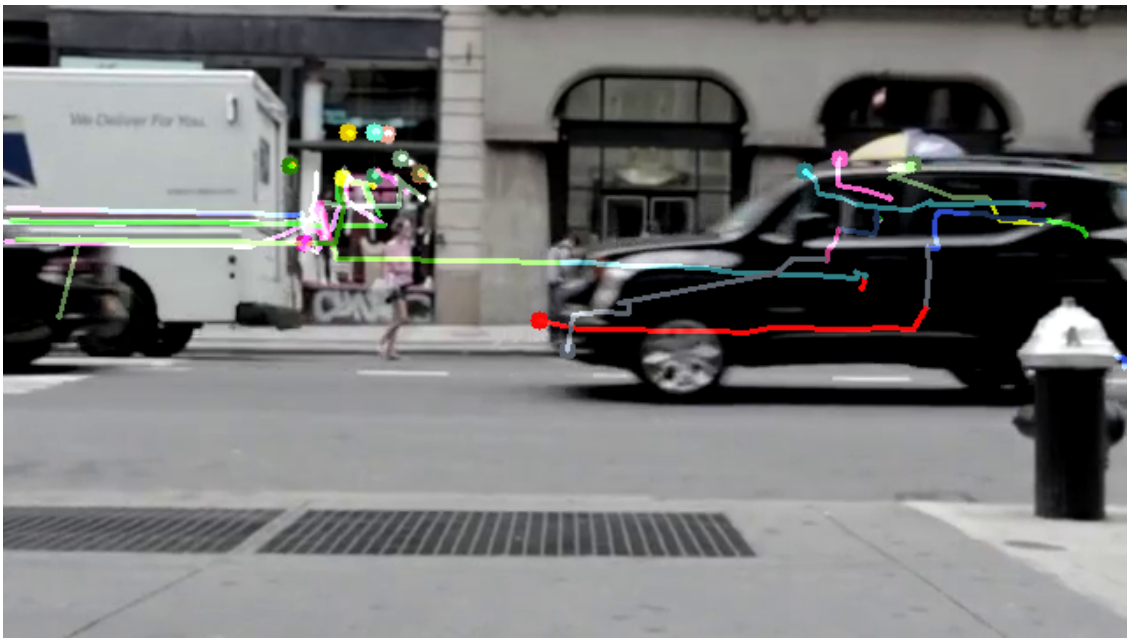


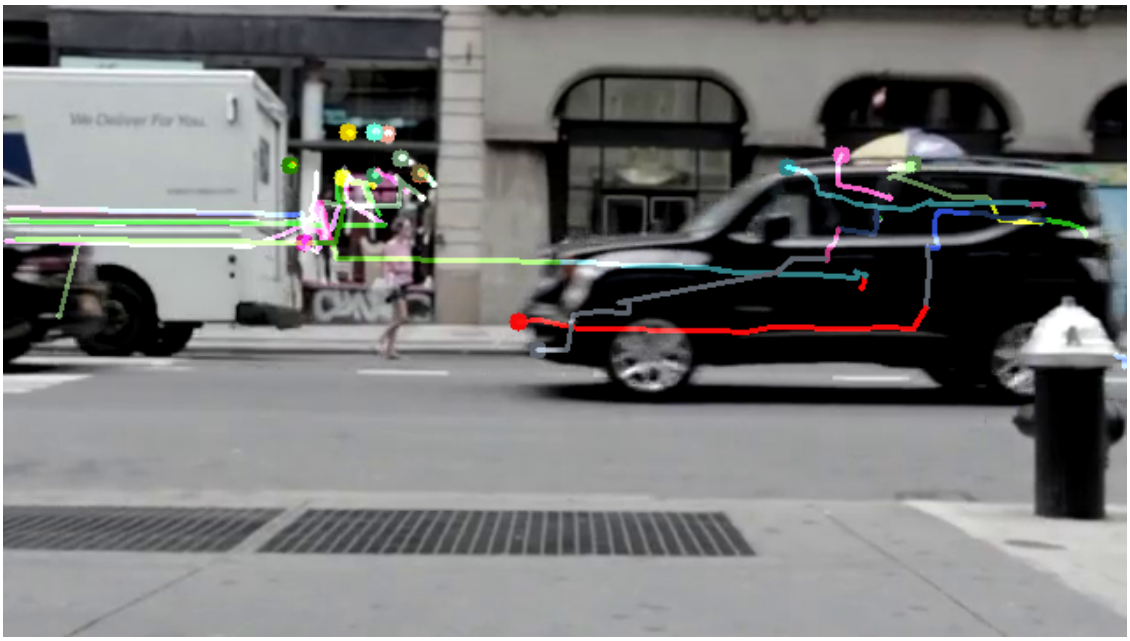


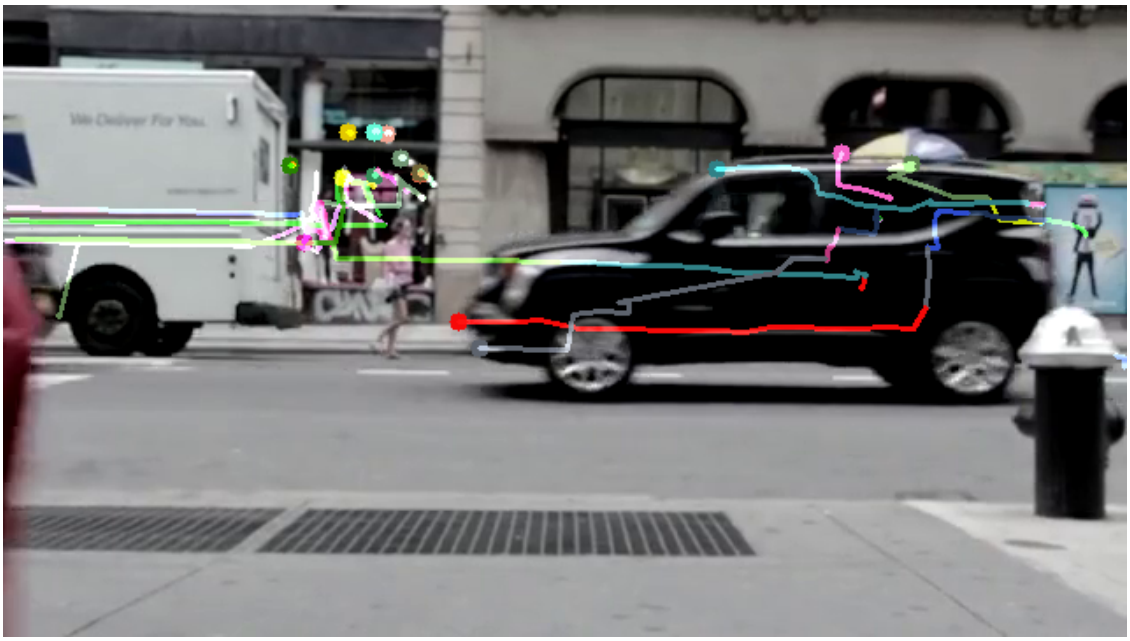


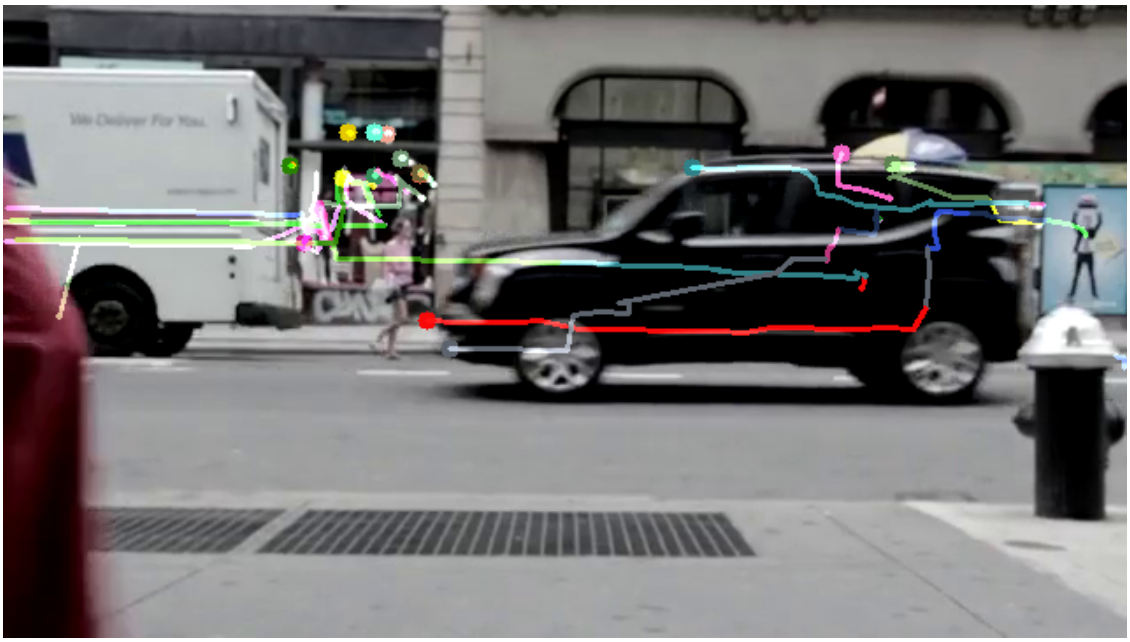


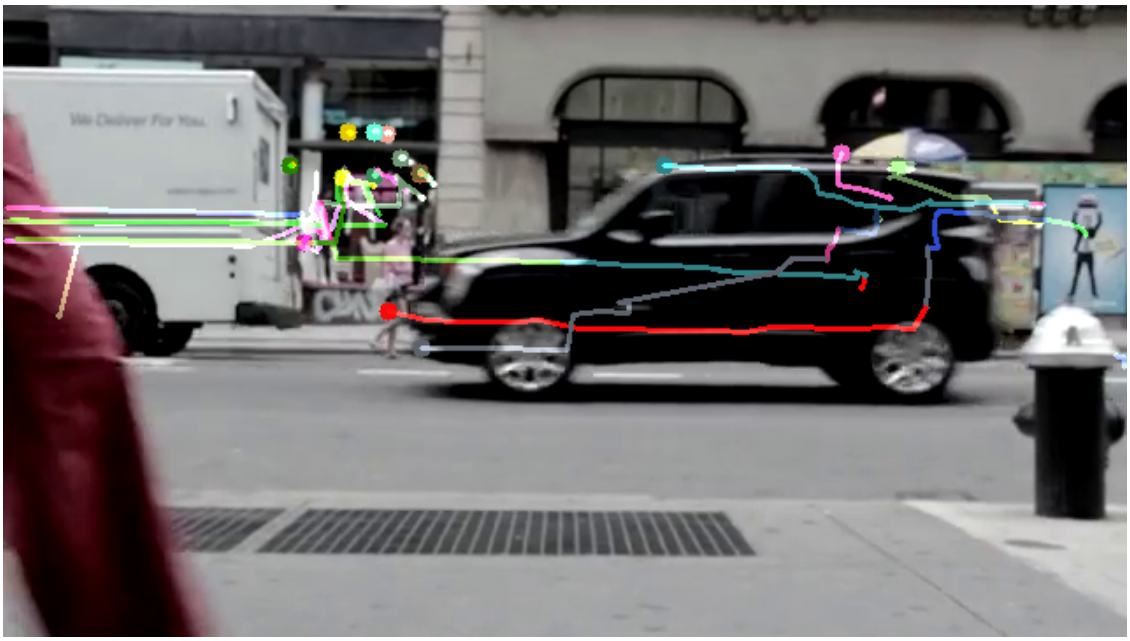


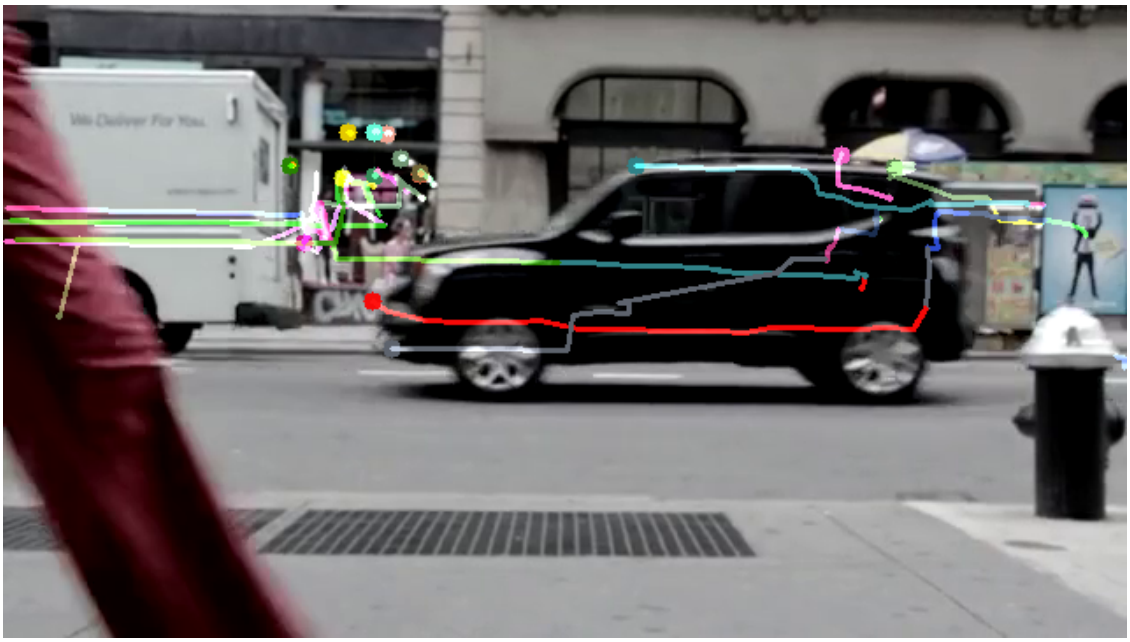


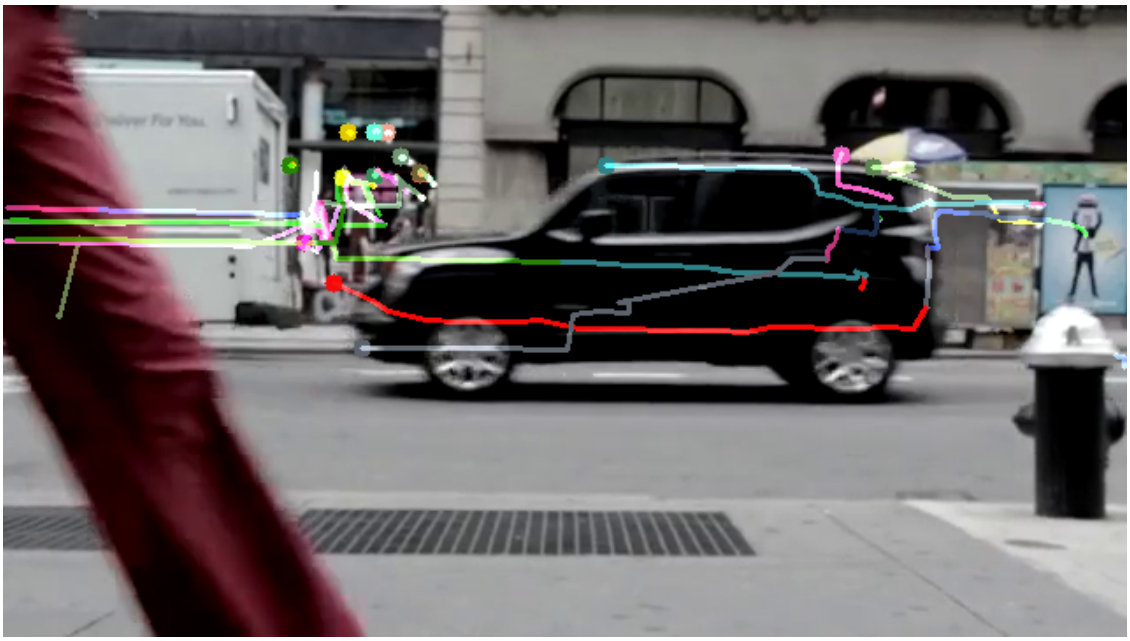


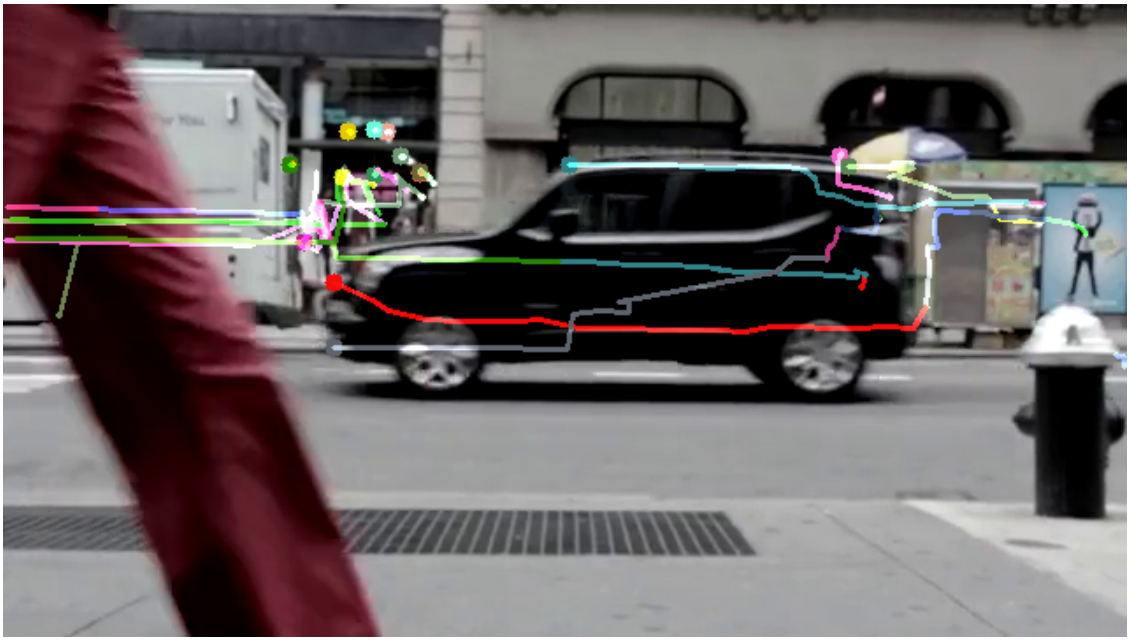


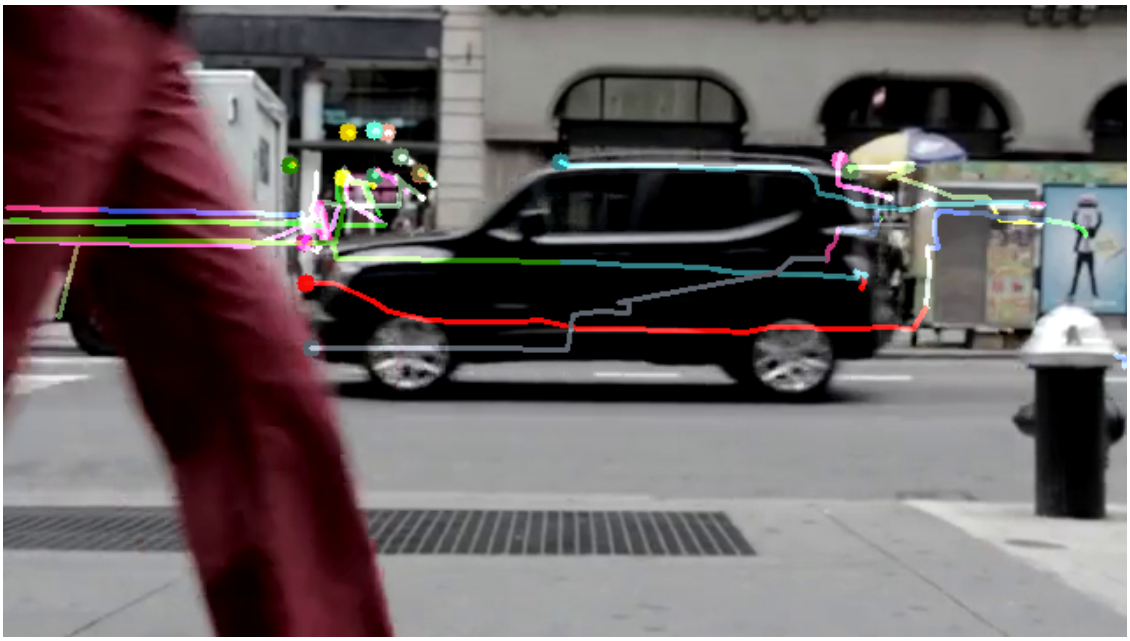


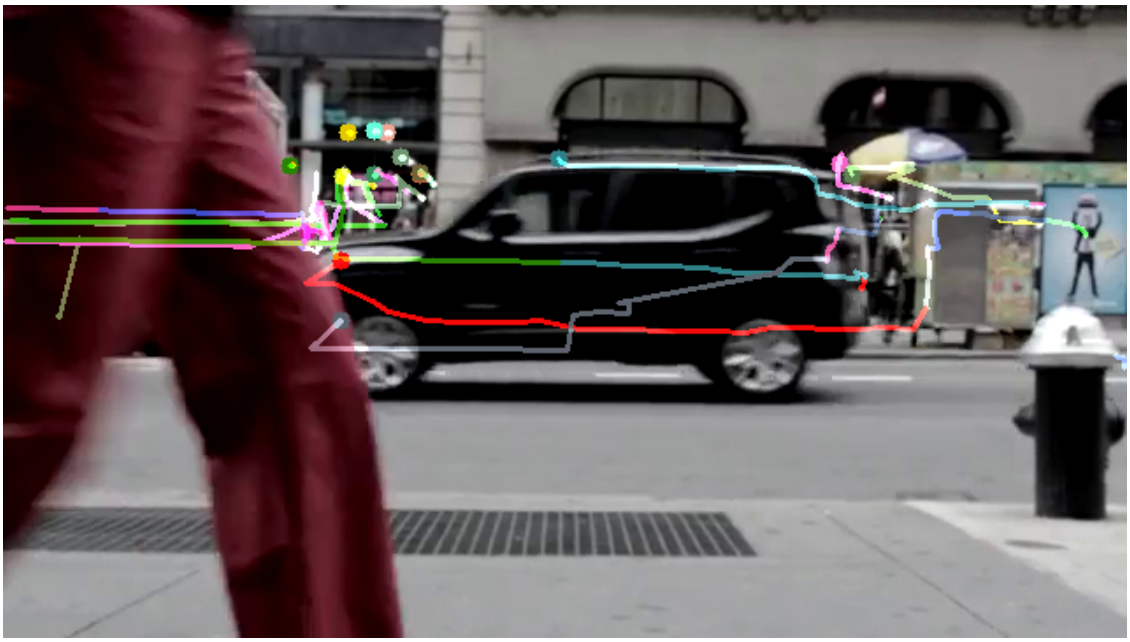


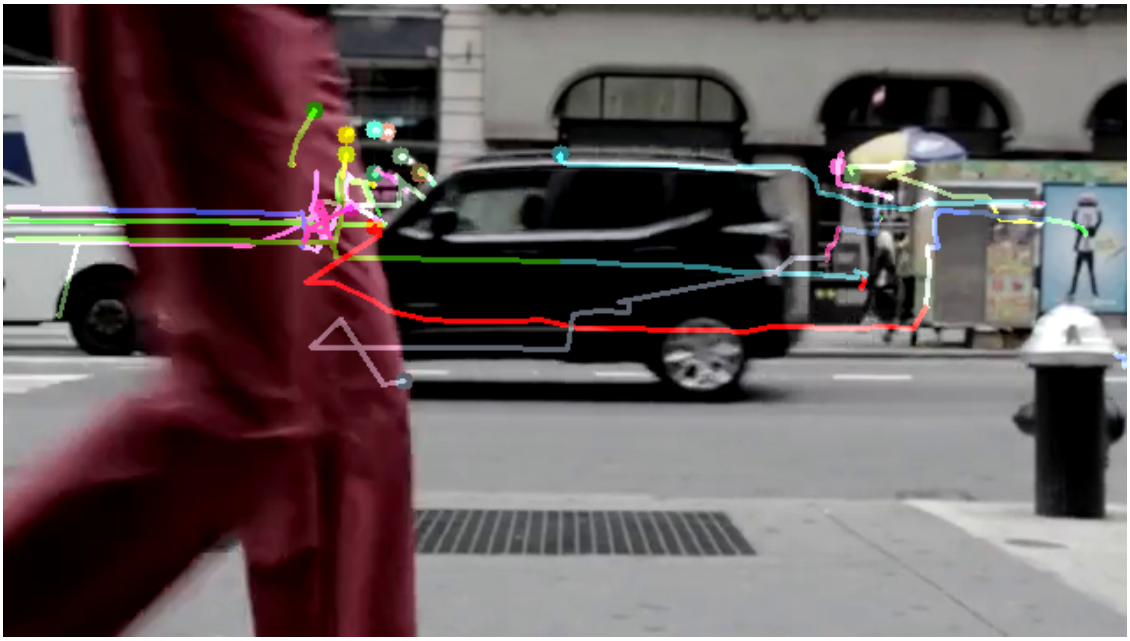


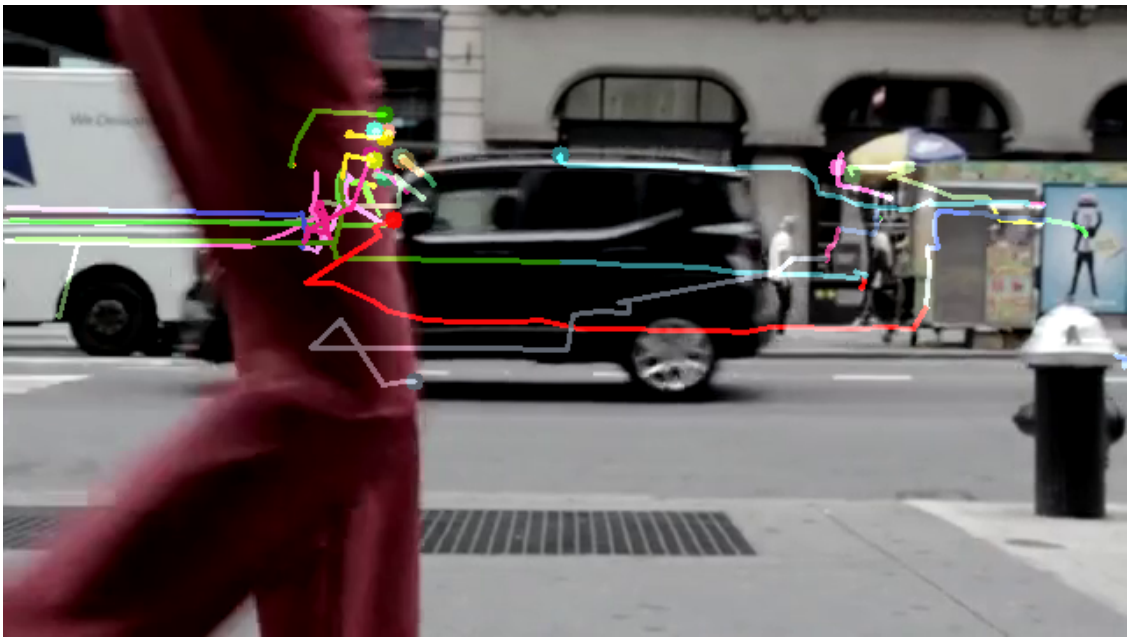


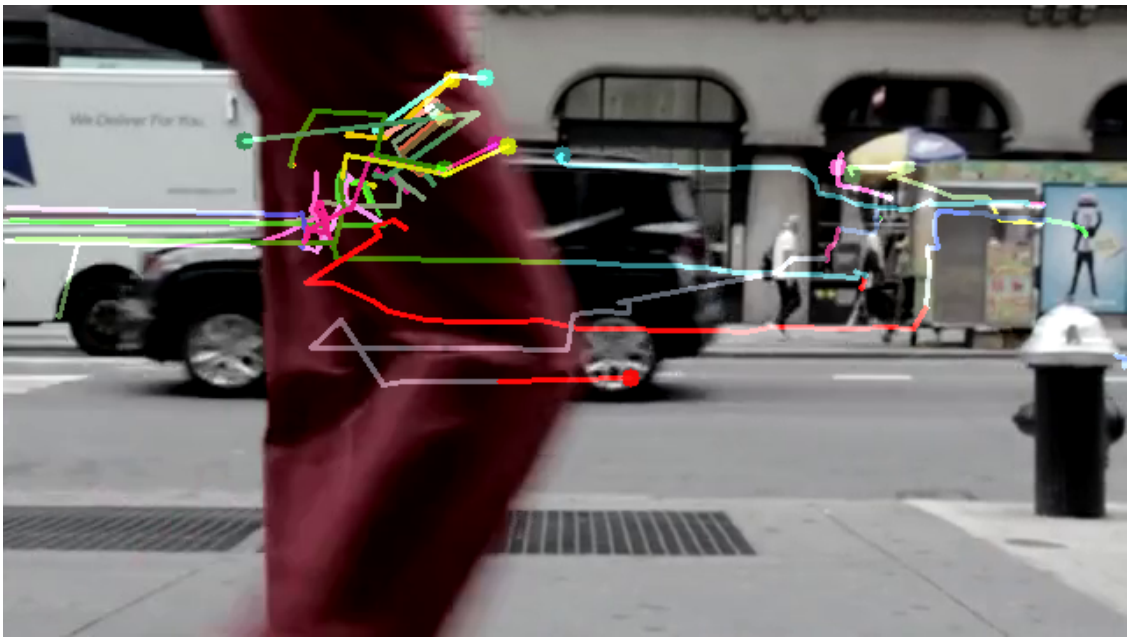


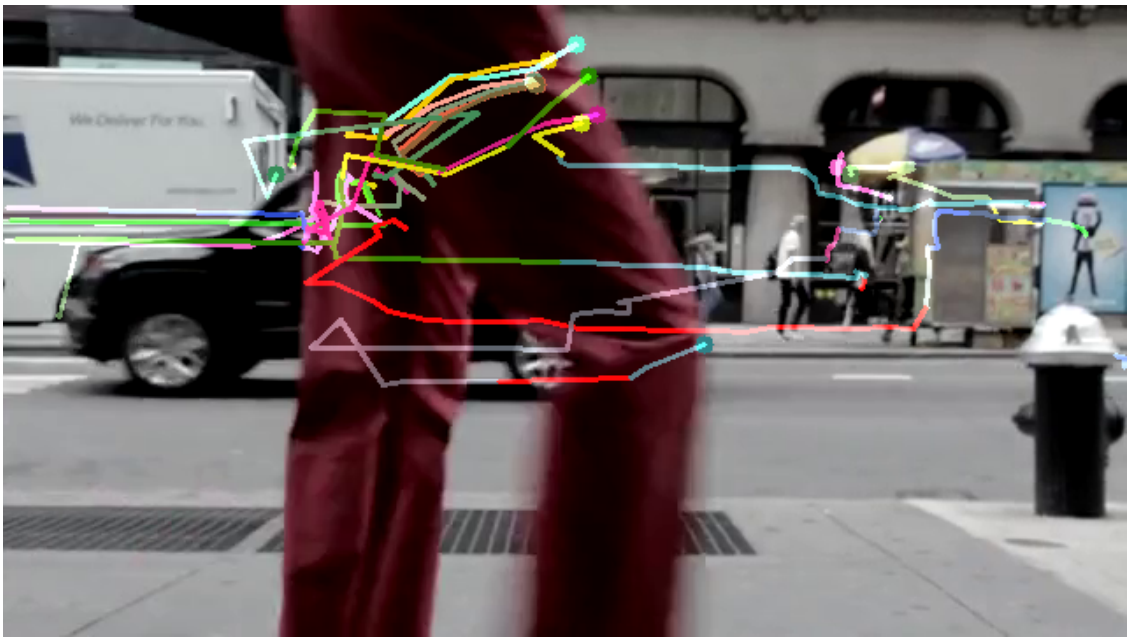


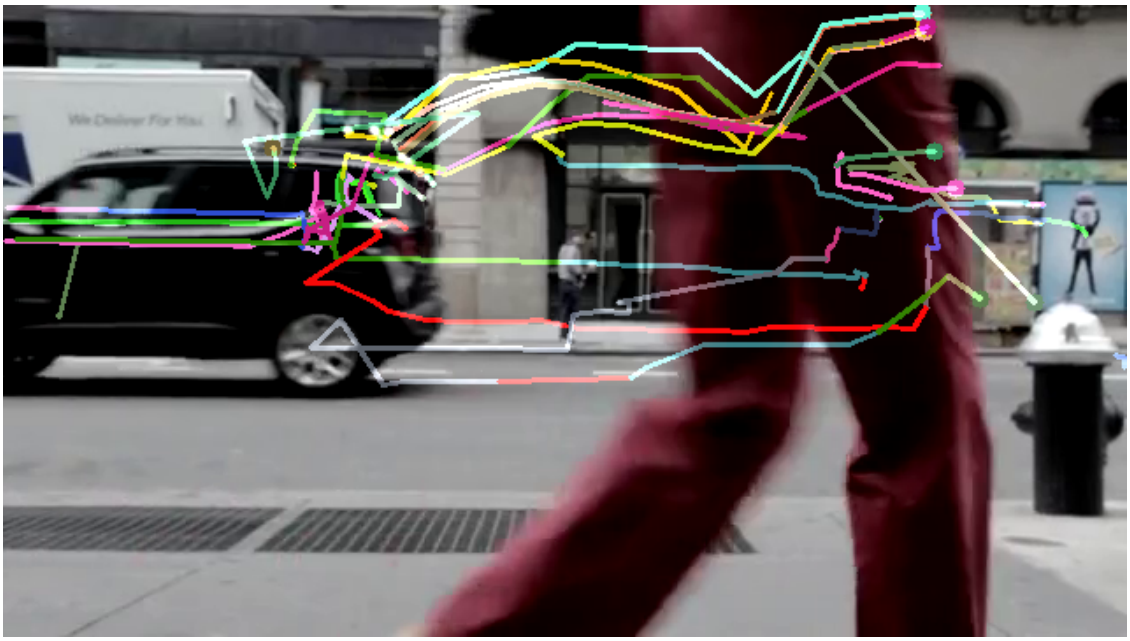


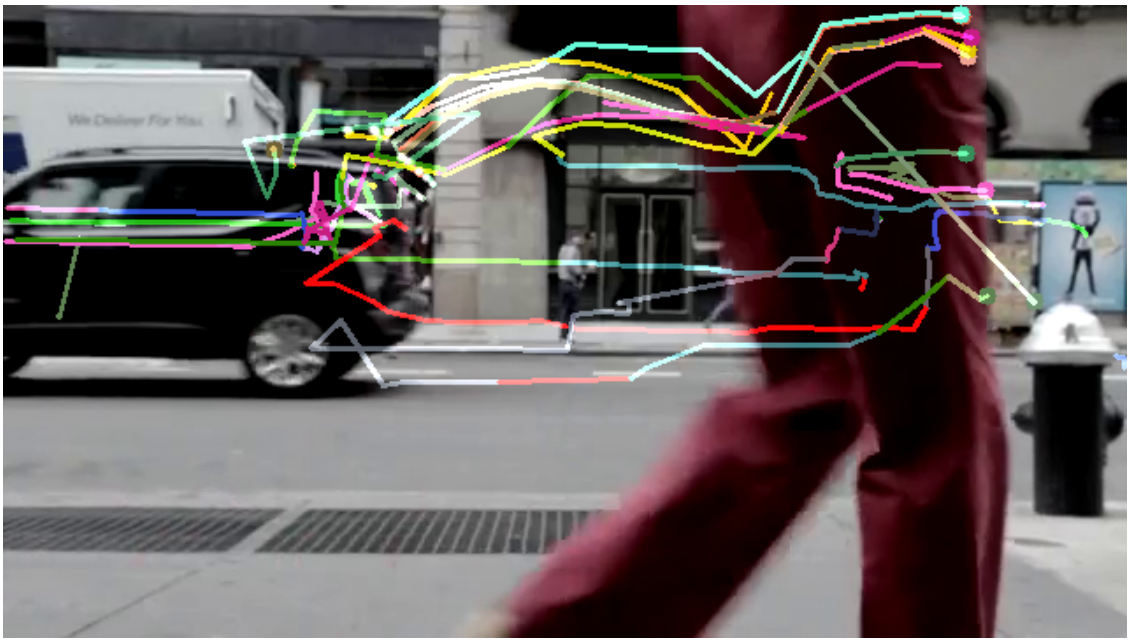


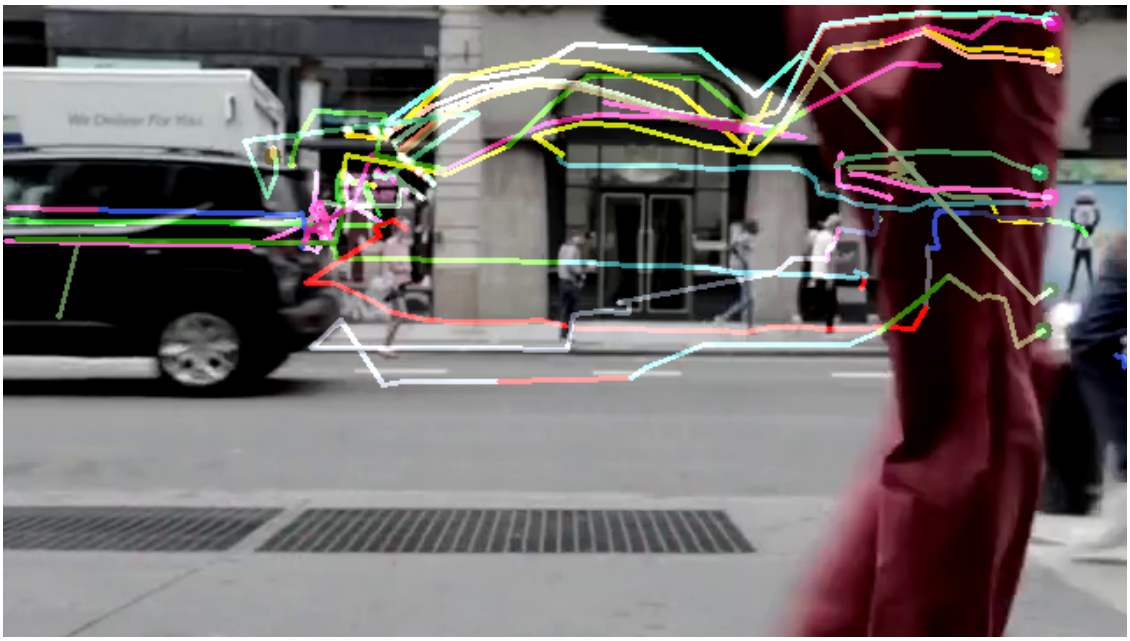


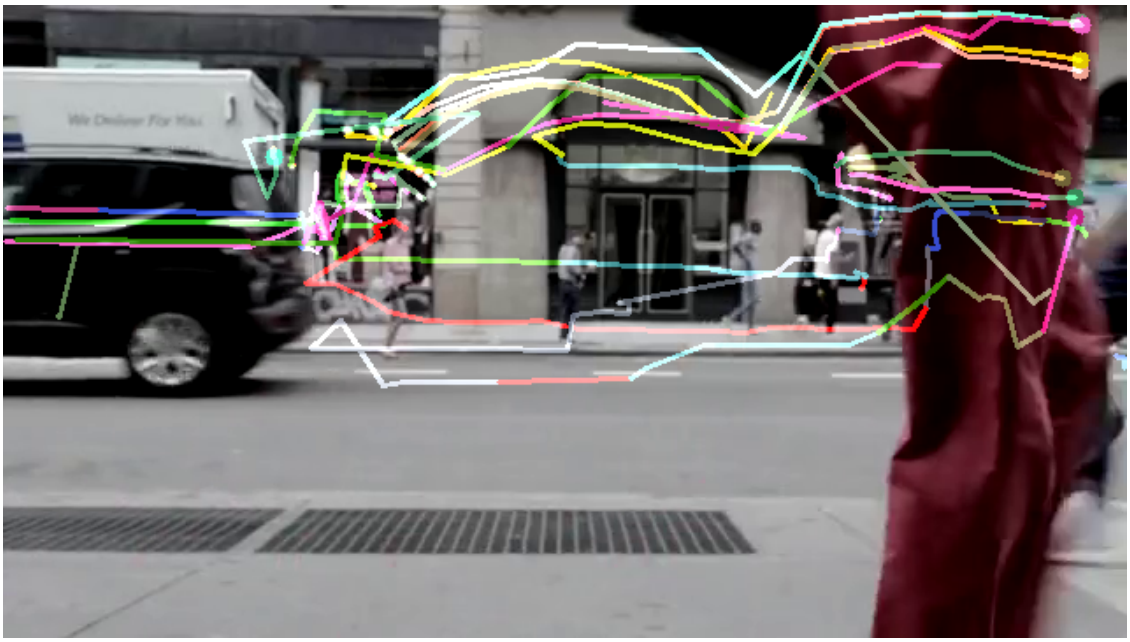


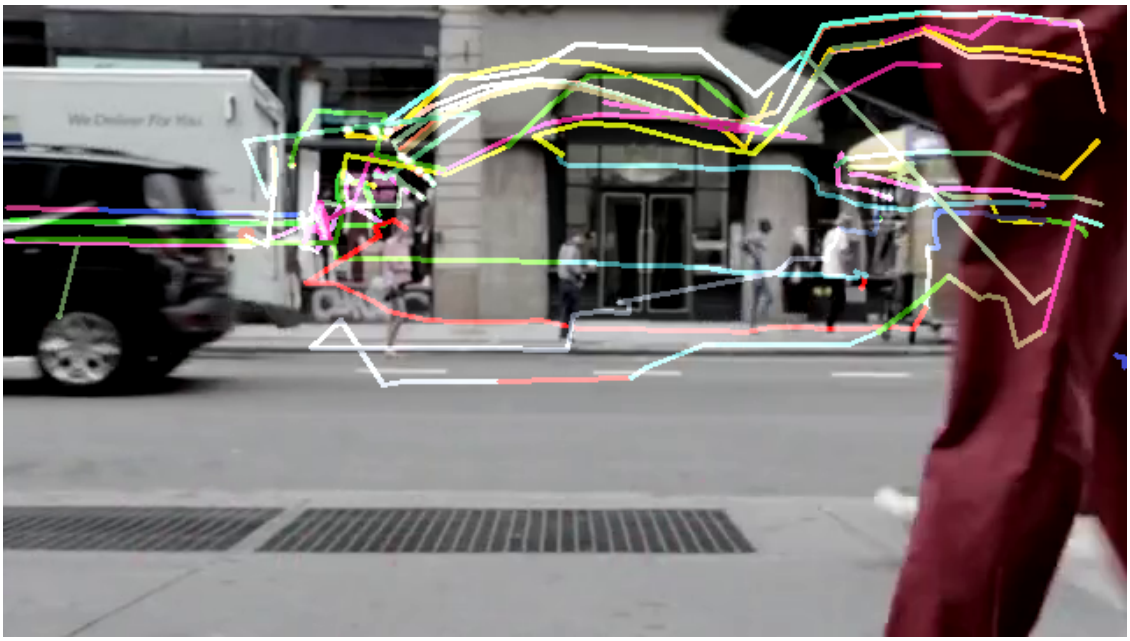


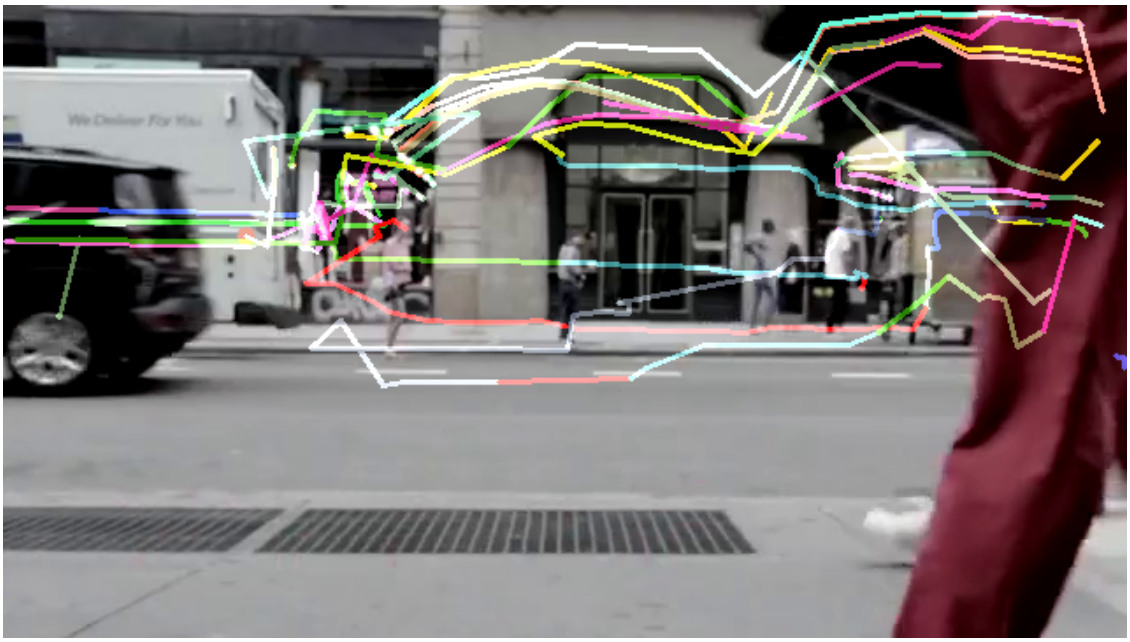


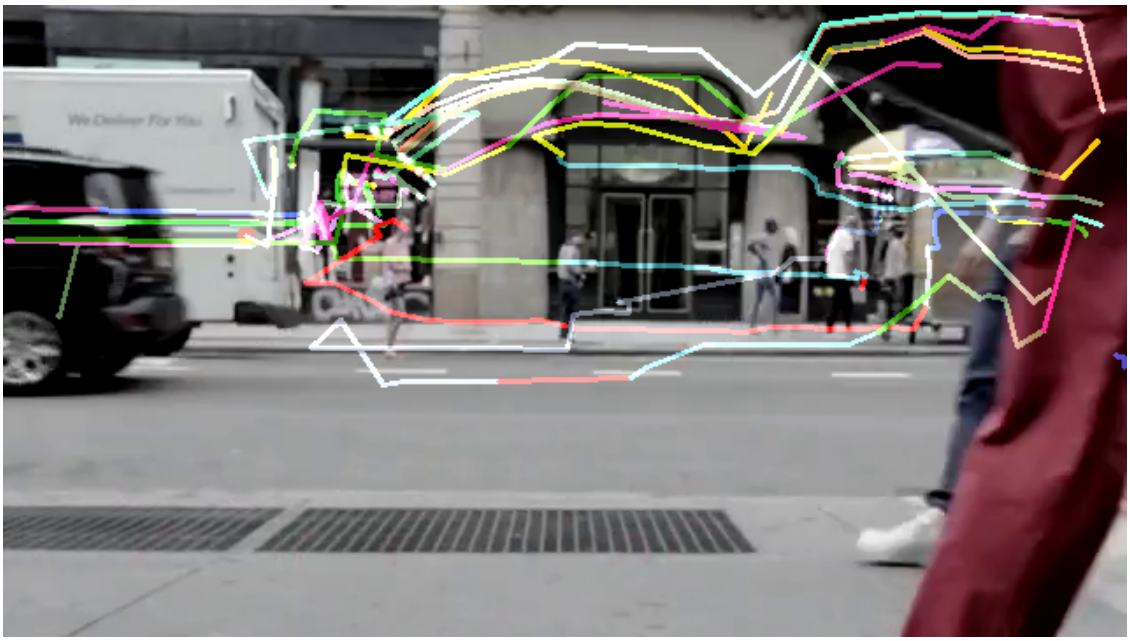


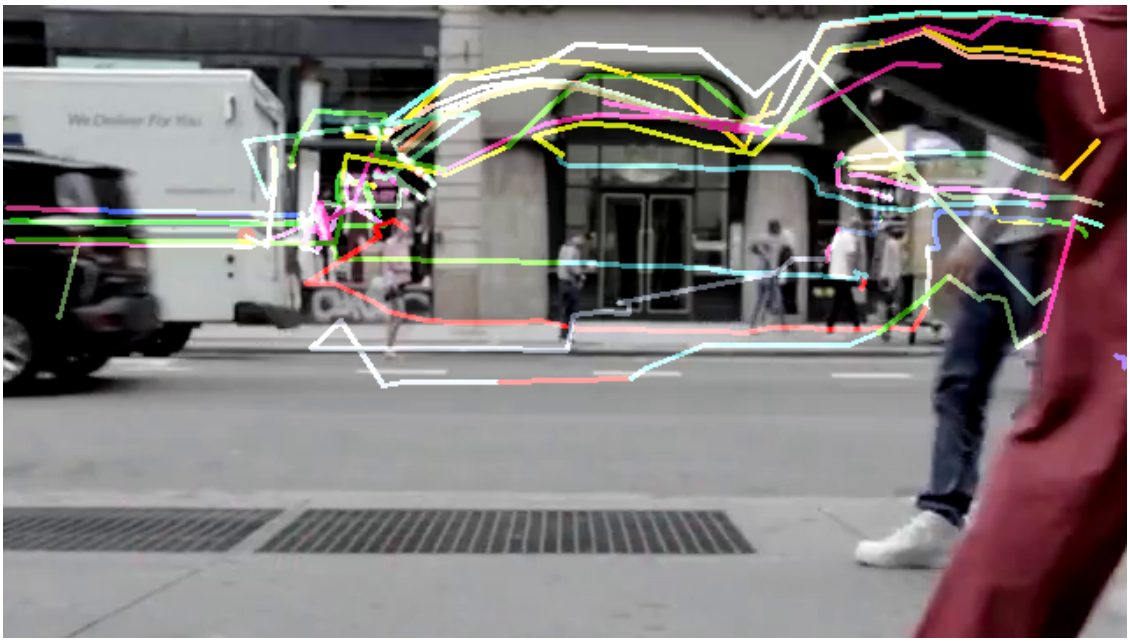


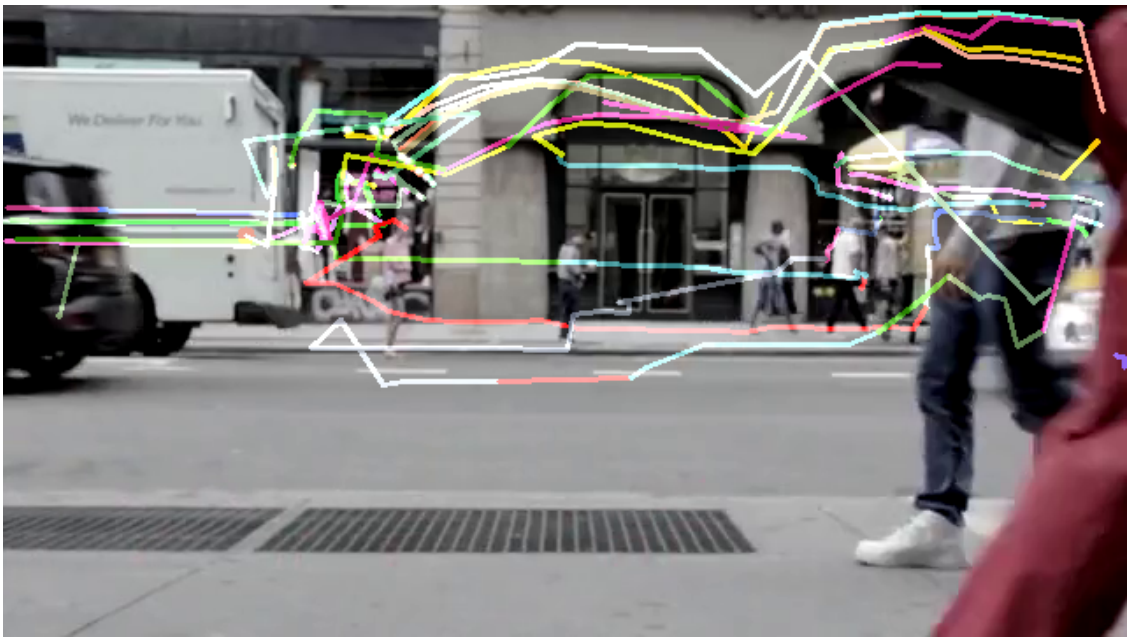


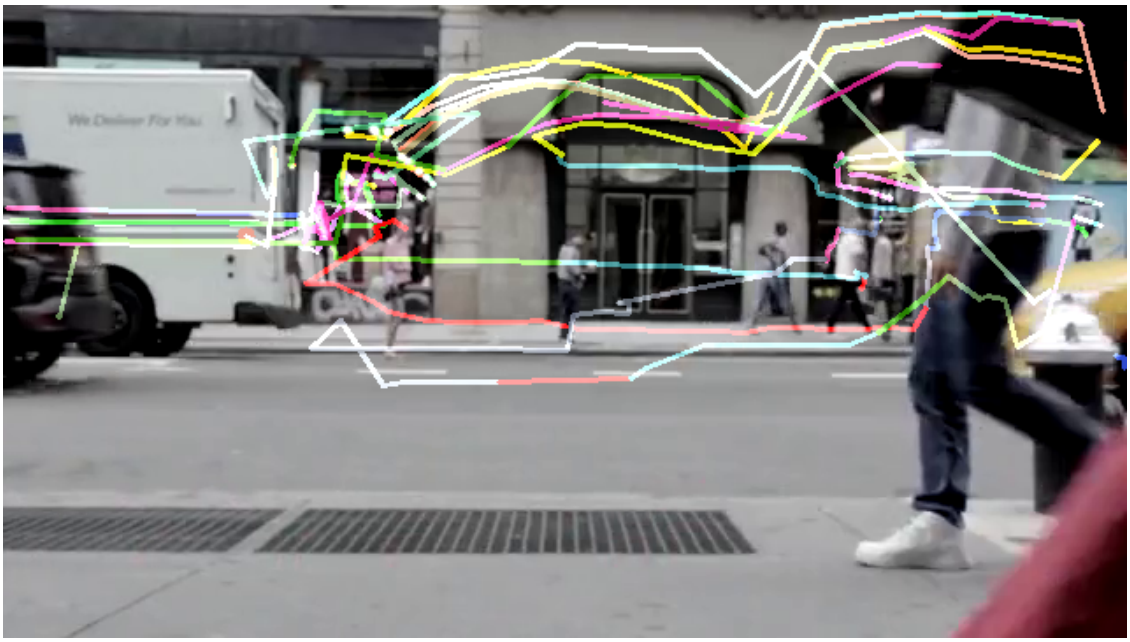


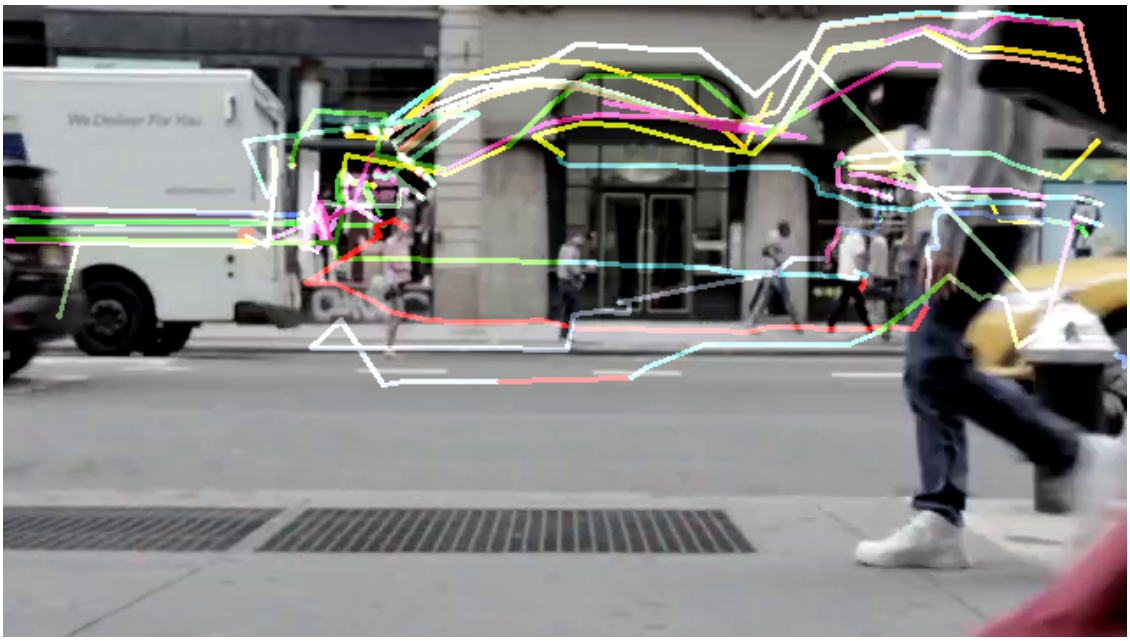


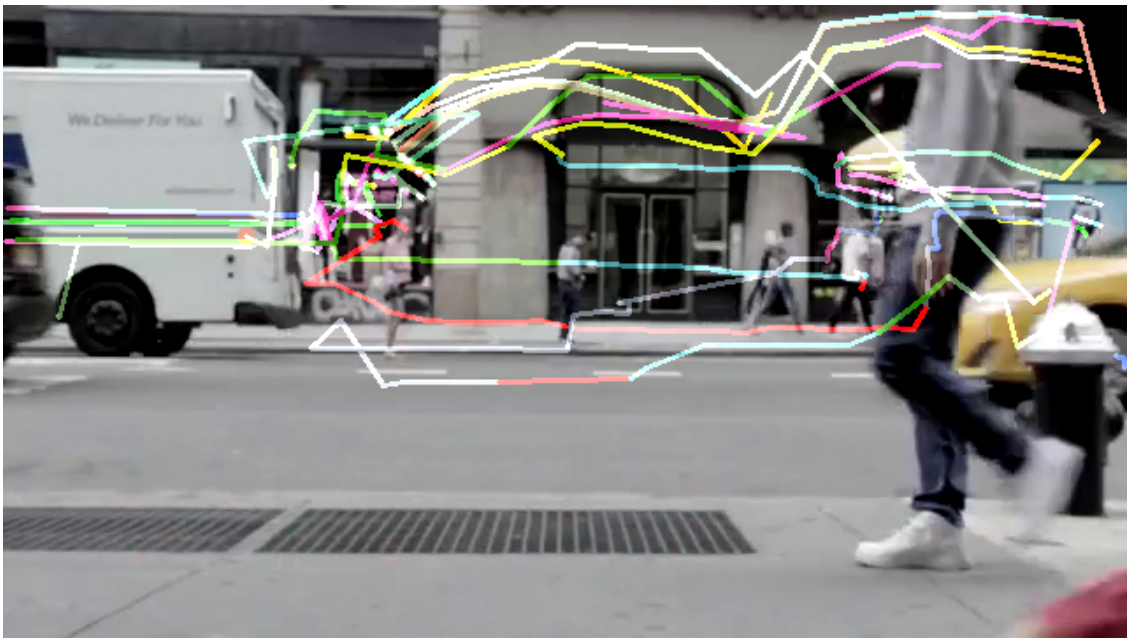


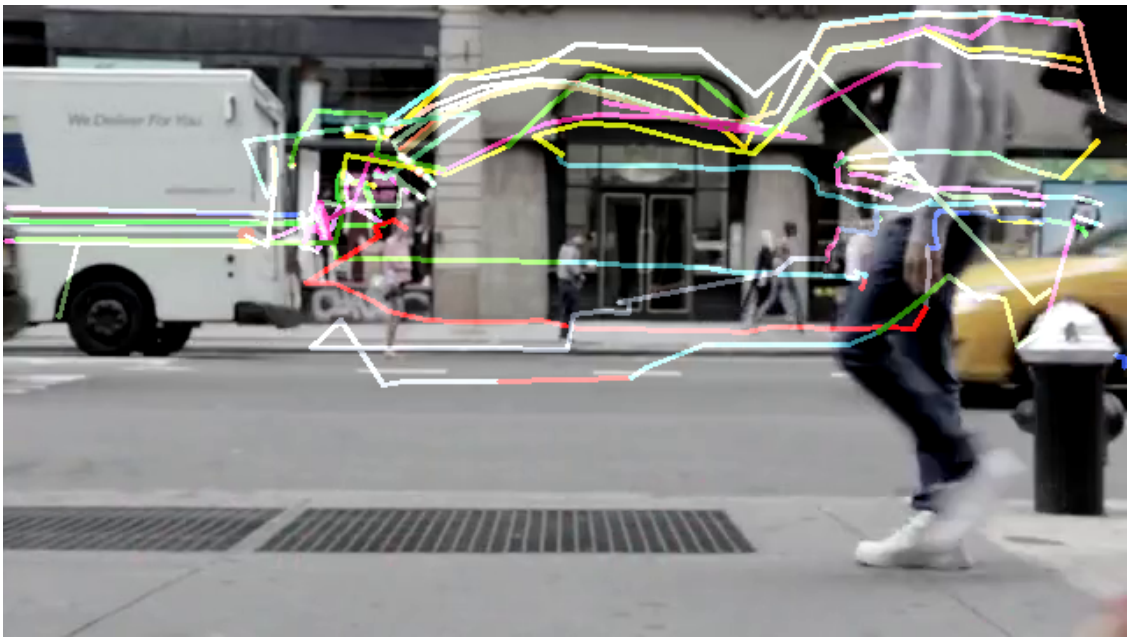


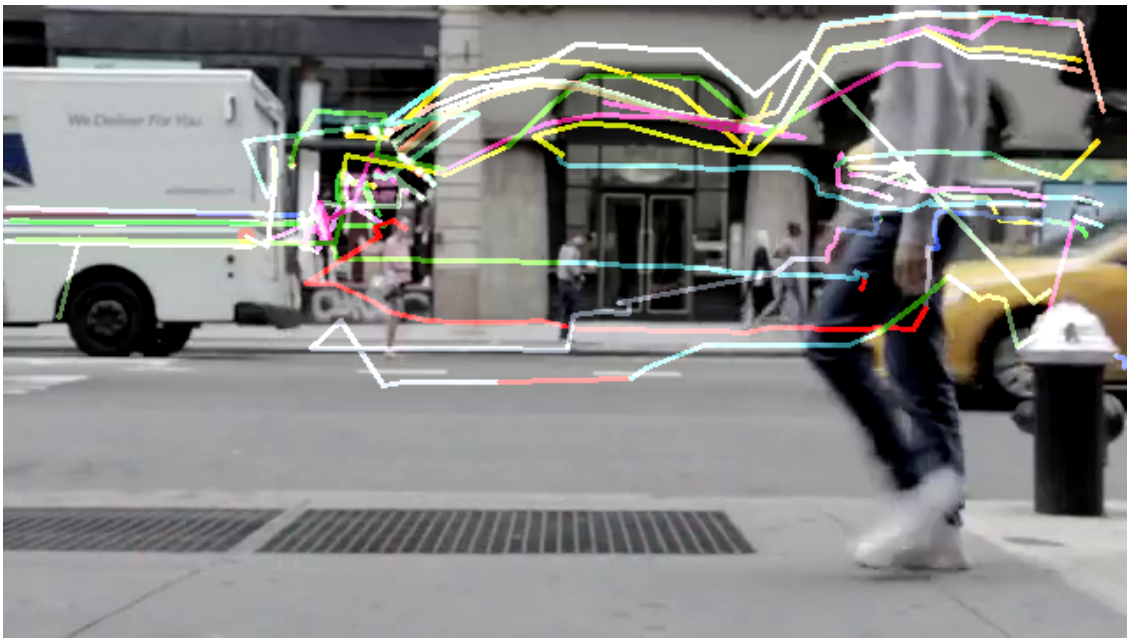


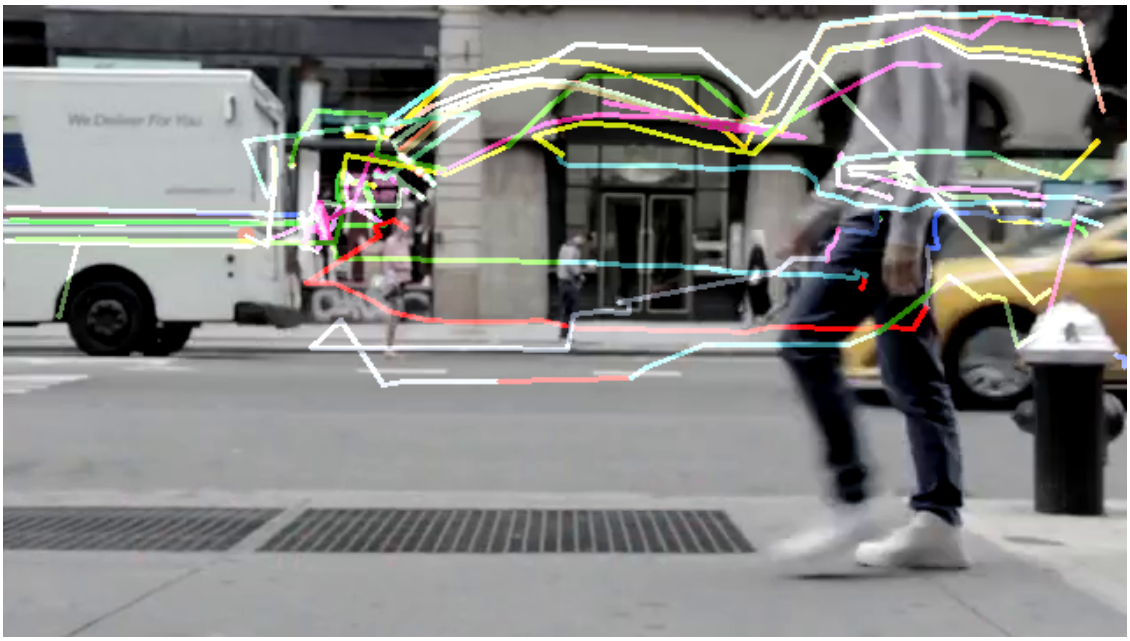


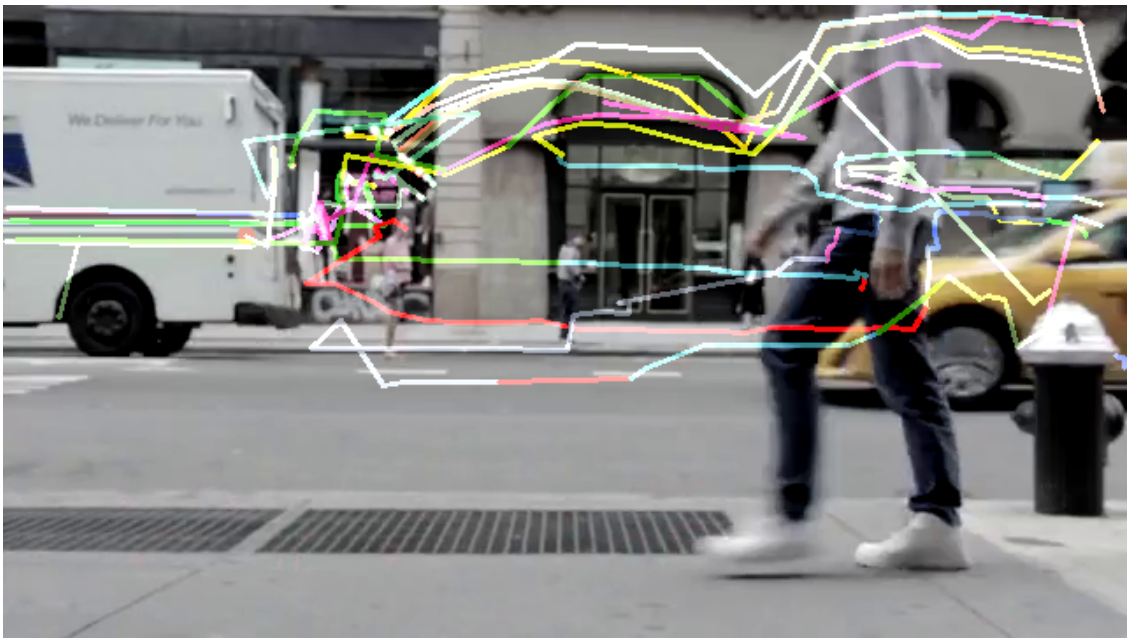


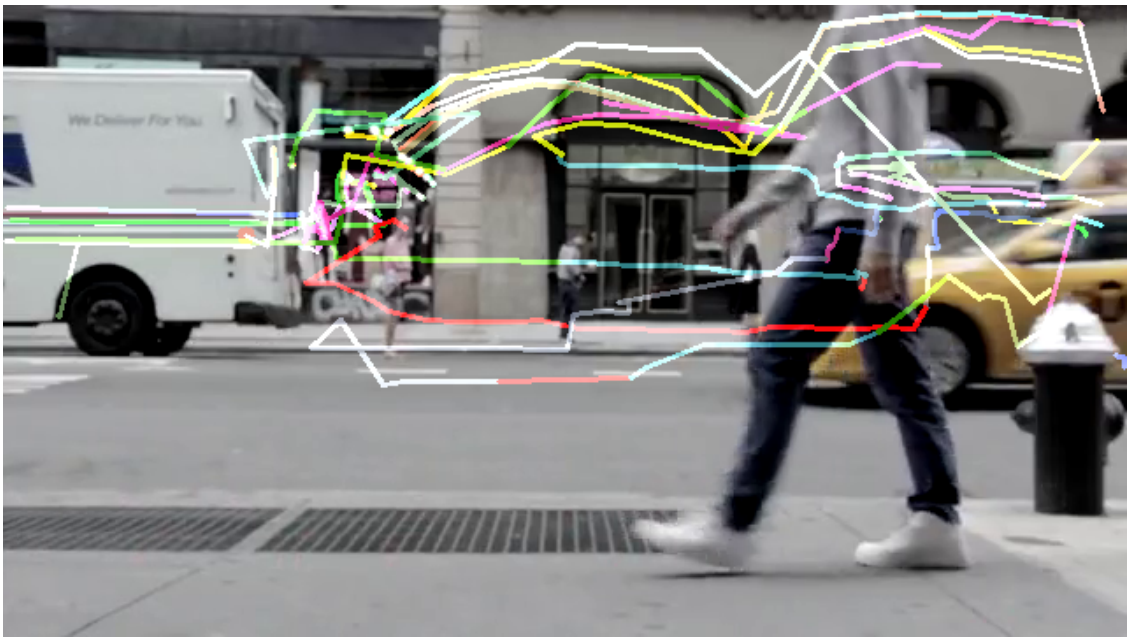


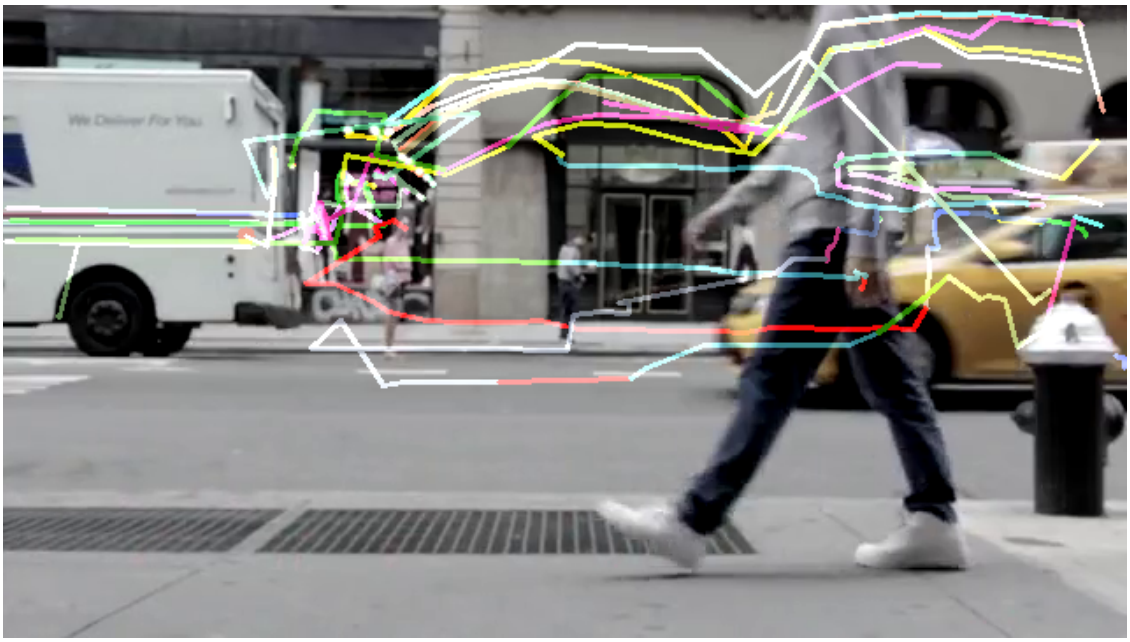


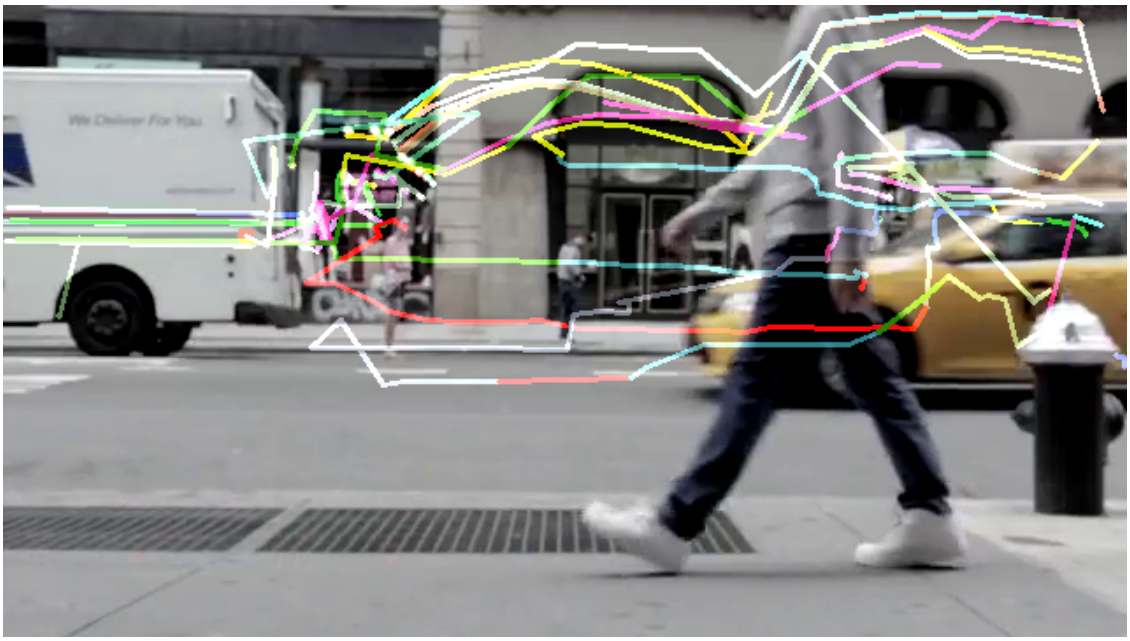


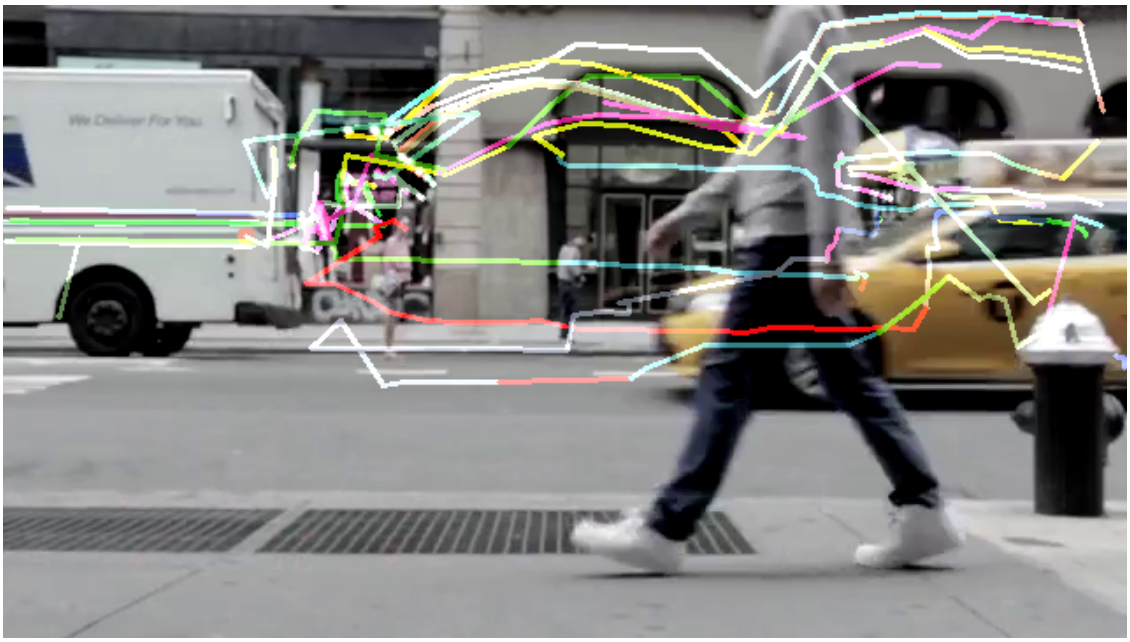


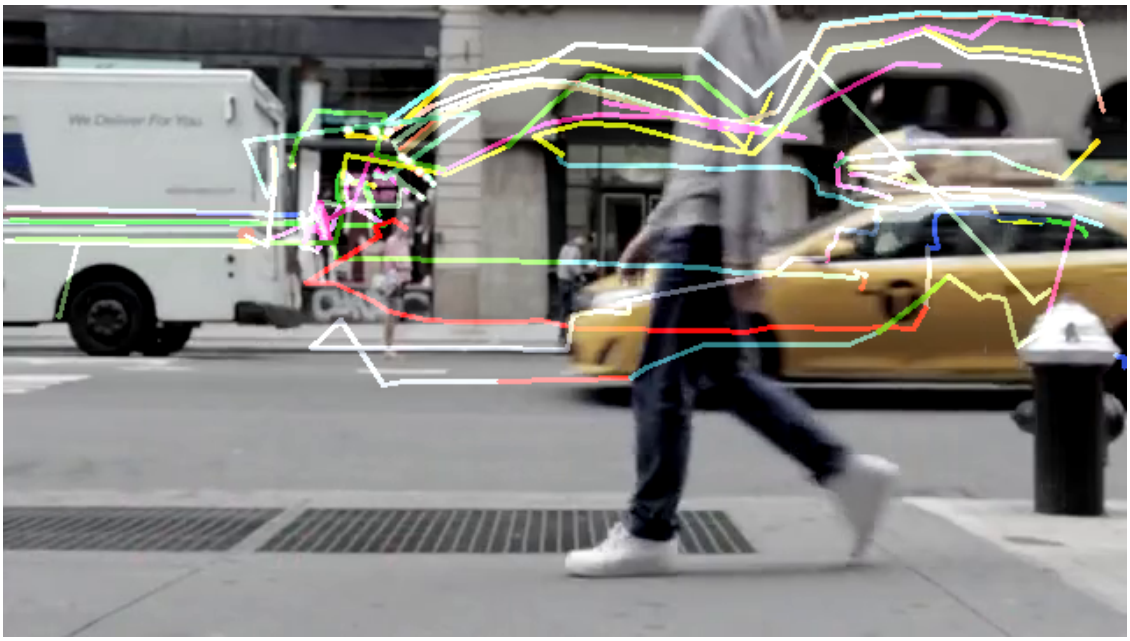


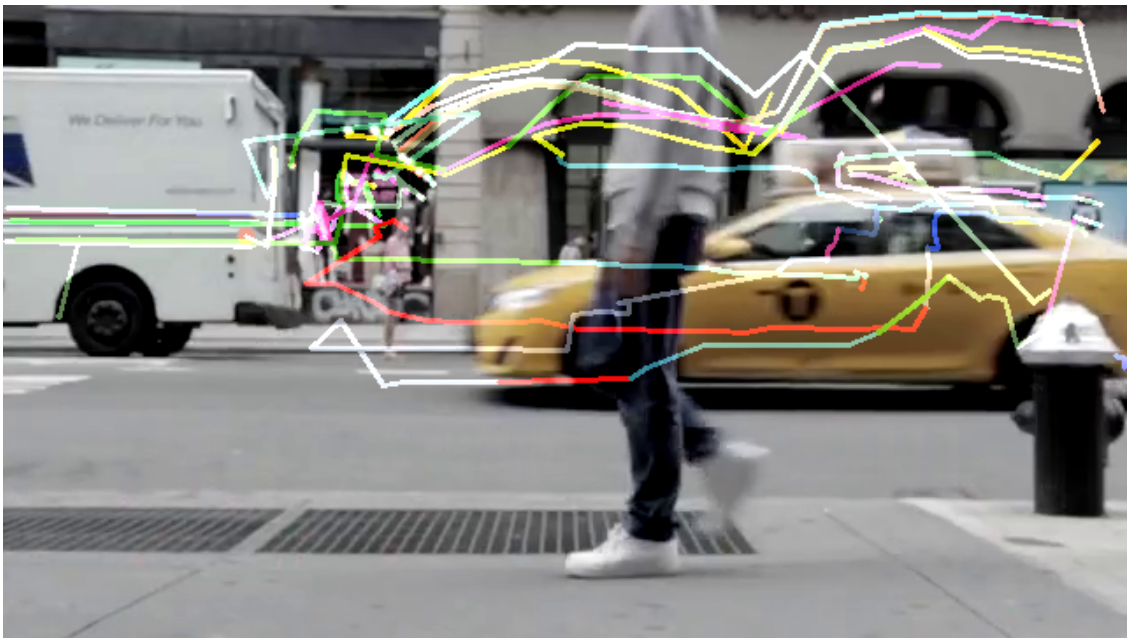


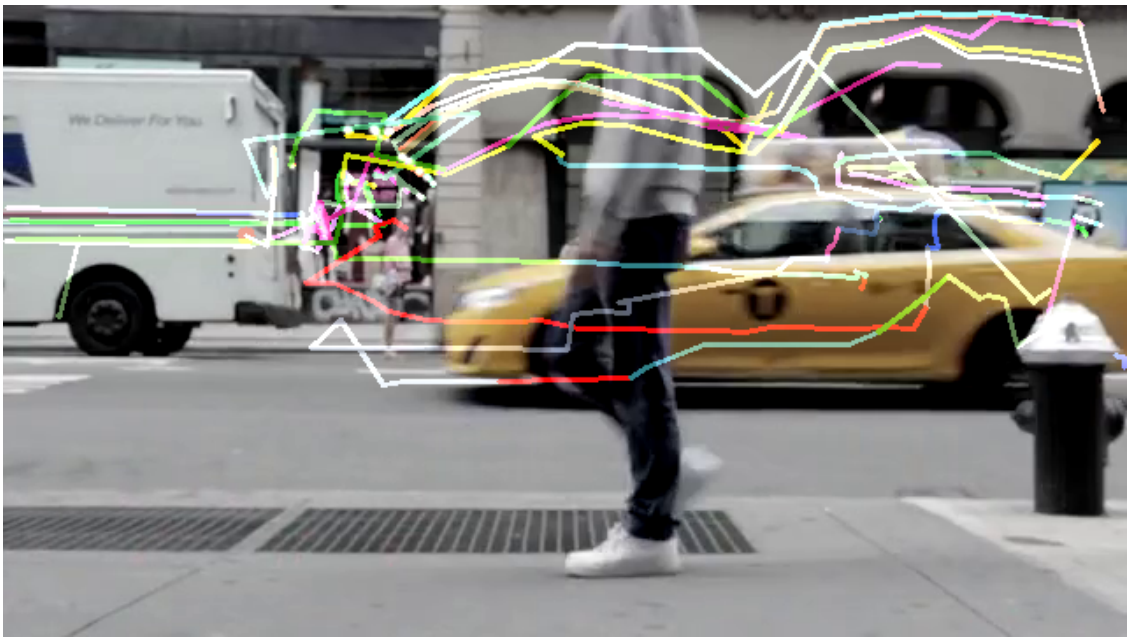


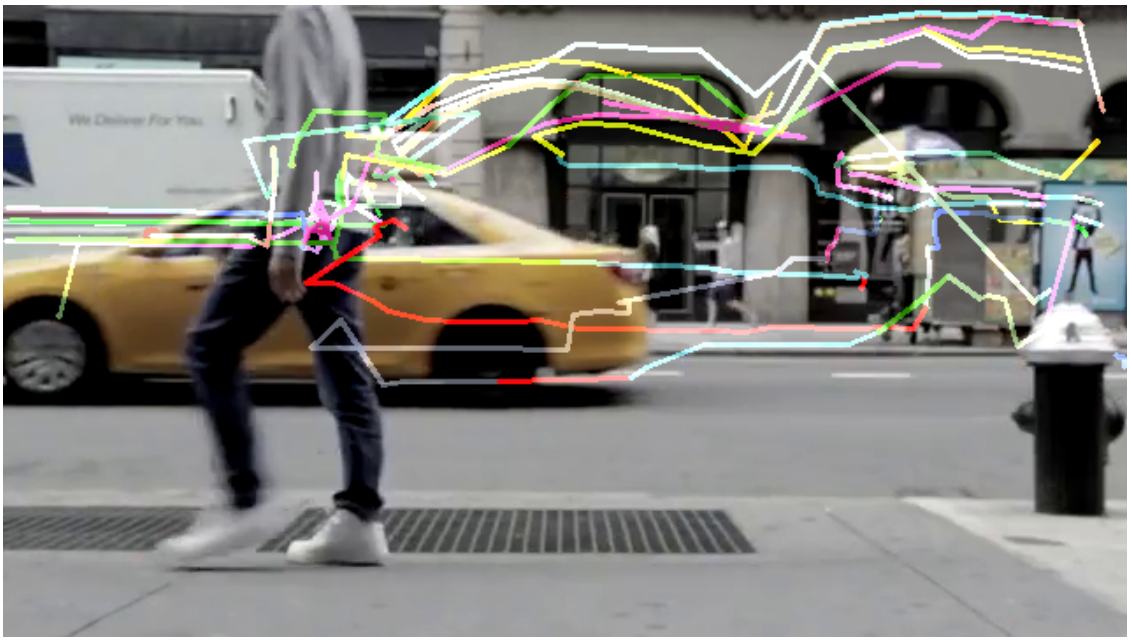












```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-b2d6d3c300ae> in <cell line: 63>()
     61
     62 # Call the function with the path to your video
---> 63 lucas_kanade_method("people.mp4")
```

```
<ipython-input-1-b2d6d3c300ae> in lucas_kanade_method(video_path)
     30         p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray, p0
 ↪None, **lk_params)
     31         # Select good points
---> 32         good_new = p1[st == 1]
     33         good_old = p0[st == 1]
     34

TypeError: 'NoneType' object is not subscriptable
```

[ ]: