Hi all

For Task B I did the Lab 7.4 Hungarian Cost.

Task B

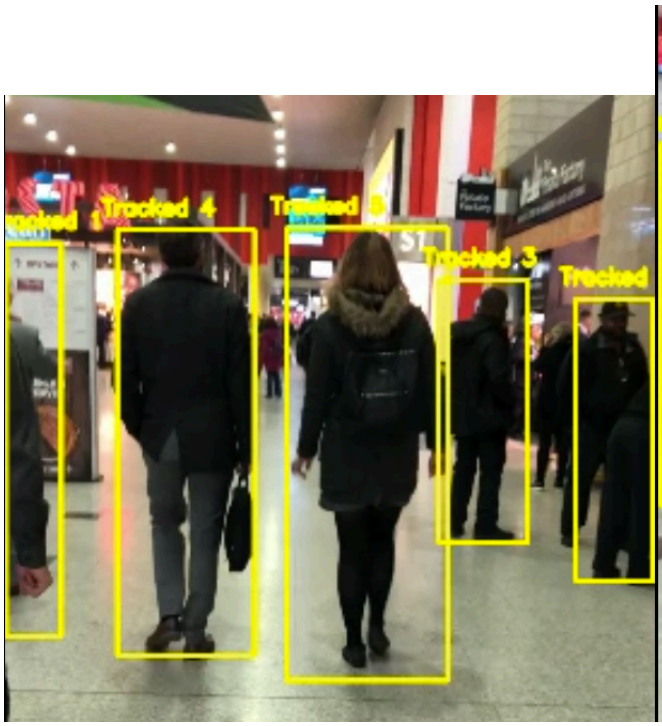
below is a breakdown of the excercise analysis

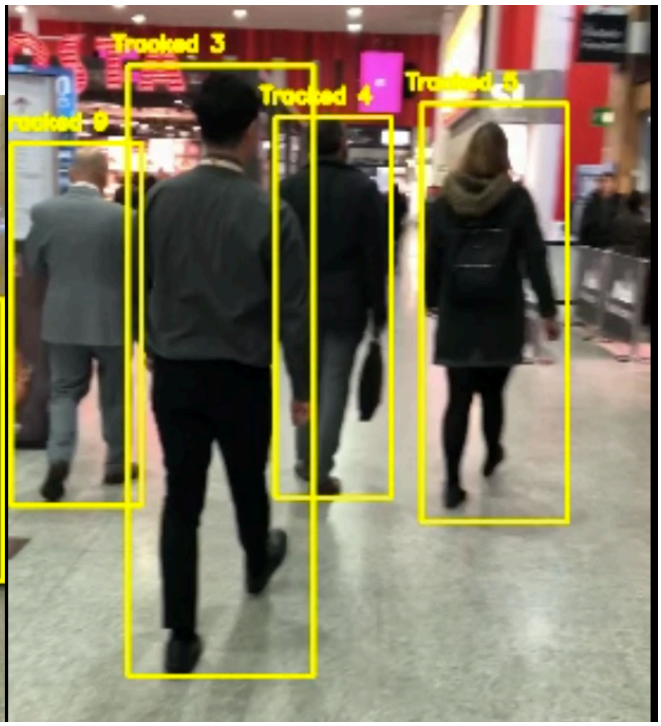
Lab_7_4_HungarianCost

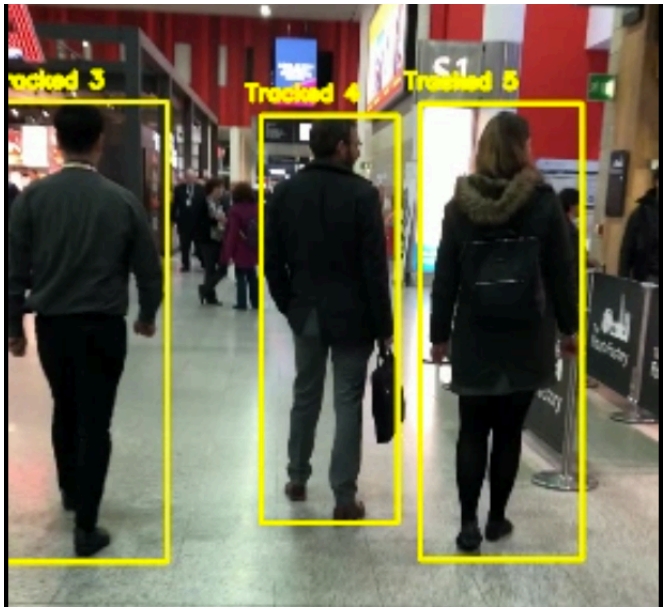
I modified the notebook functions per the exercises and ran 7 experiments.

- Link to Functions py here (contains the functions needed to run) - LINK HERE
- Link to the Full Notebook here (had to clear output as it was v.large) - LINK HERE

Results of Original Demo of HungarianCost







As you can see it detects and tracks 9 separate objects fairly accurately. It does not pick up people in the back ground due to the minBoxArea setting.

Exercise 1

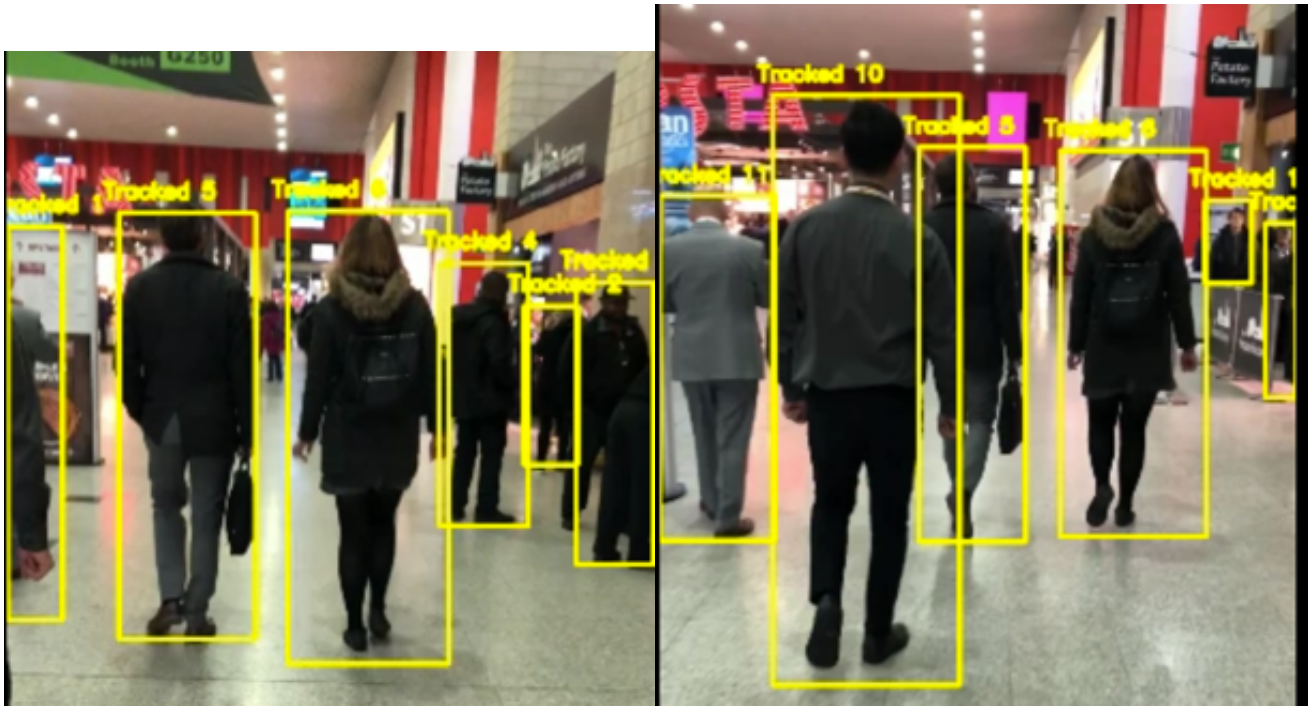
Two very small experiments.

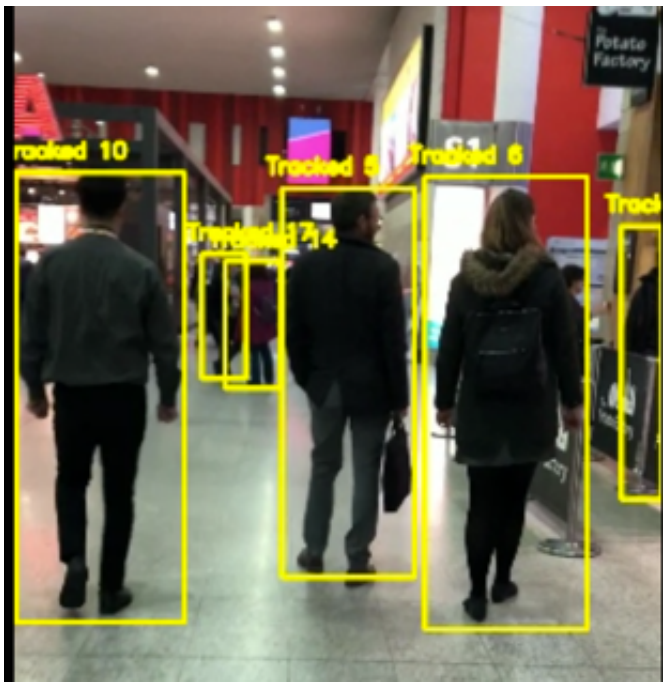
- Experiment 1.1 - Reduction of minBoxArea
- Experiment 1.2 - Increase the MaxAge of tracker to 128 frames

Experiment 1.1 Reduce the `minBoxArea` to effectively track smaller boxes.

You may also need to assign a `colour` term to the tracker as I'd expect there to be more objects to track and it should become harder to see what the program is doing. What does happen?

Results

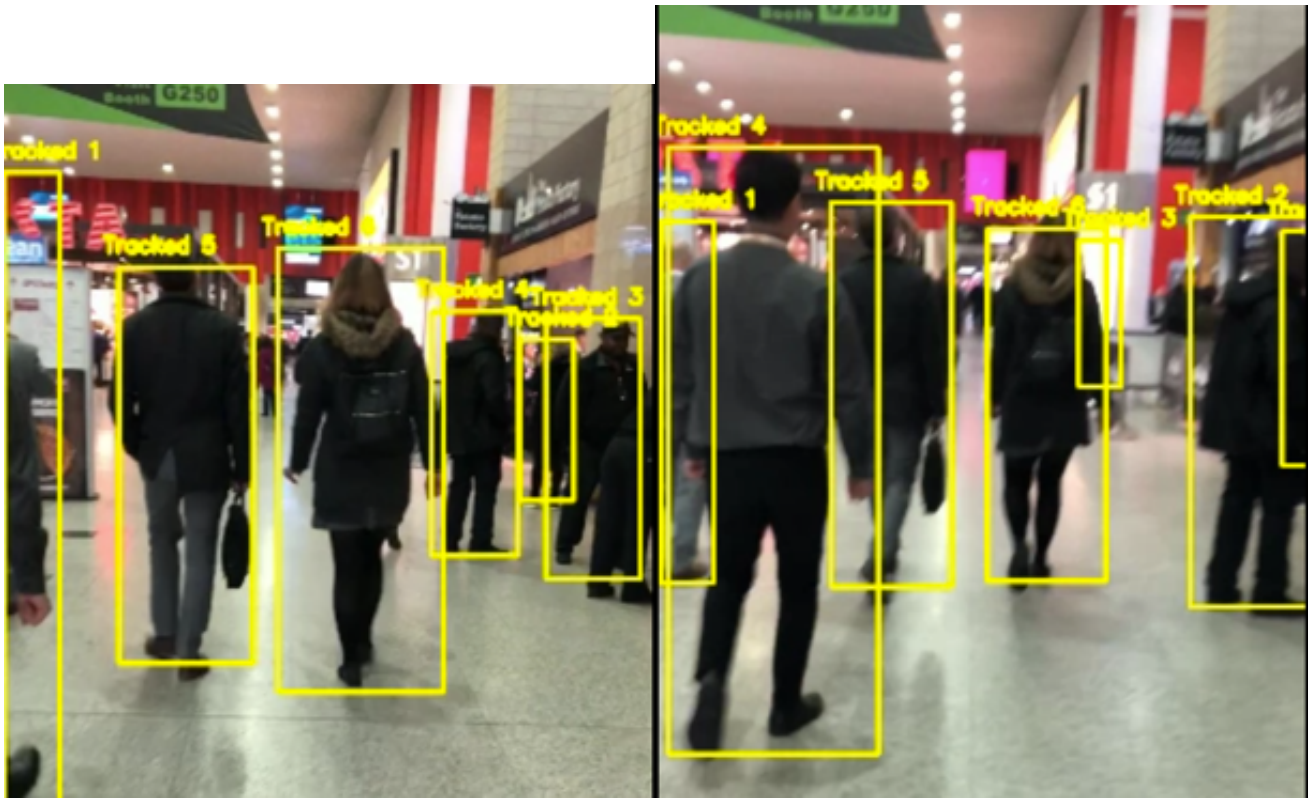


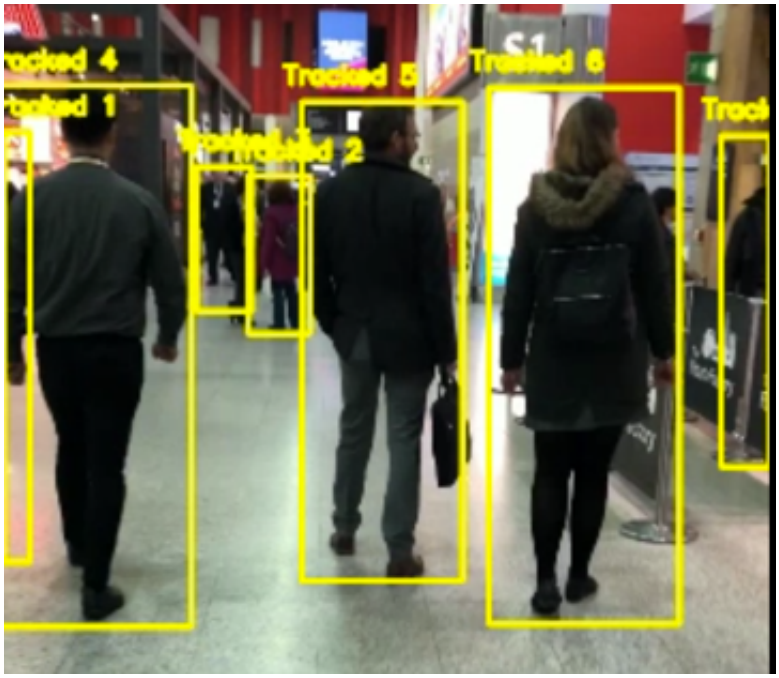


Experiment 1.2 Increase the MaxAge of the tracker for smaller boxes to 128 frames

maxAge - this is where we're setting the trade-off between treating a detection as part of an old series or starting a new series. All we do is simply say - if a tracker hasn't had a match for n frames, then get rid of it. If a detection comes in after that, then its a new Tracker.

Results





Analysis

Decreasing the area of the minBoxArea to 100 * 100 (10000) we pick up far more objects in the video. We detect 17 separate objects, including people in the background fairly accurately however, due to the occluding of object 14 and 17, this tracking is then propagated onto other objects in the background. Increasing the age of the tracking using the MaxAge to 128 frames did not improve tracking for occluded objects in fact as you can see from the second syreengrab for experiment 1.2 the tracker for object 3 gets orphaned as the object for the tracker goes off screen. Also at the start it tracks object 1's arm, that person leaves the frame and then comes back into the frame a few seconds later, but gets picked up as Tracked Object 4. It switches objects to tracked 1.

Exercise 2

Rework ***boxCost*** to combine:

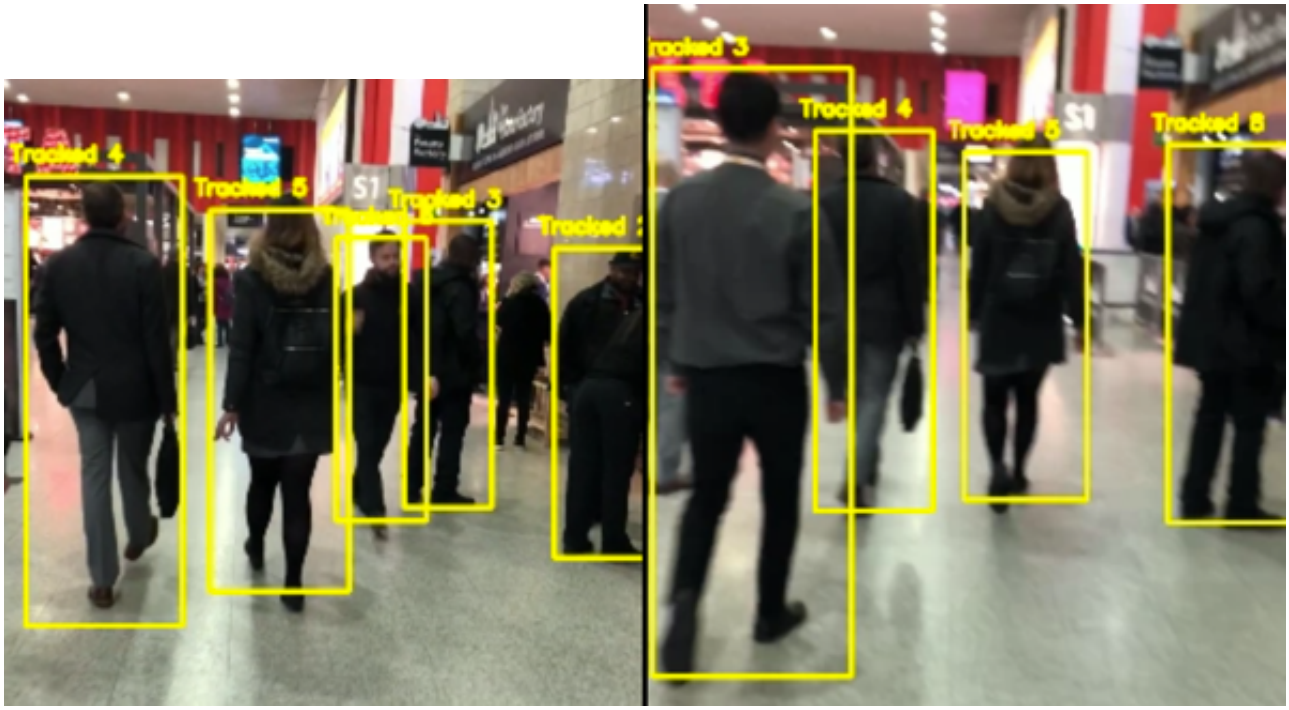
- Euclidean Distance
- One other term that you think might improve the cost function, e.g.:
- Don't forget the `ndThreshold` term will need to be adjusted as well. Does it improve the tracking.

For this I did 3 experiments.

- Experiment 2.1 ndThreshold Increase (double)
- Experiment 2.2 ndThreshold Decrease (degree 3)
- Experiment 2.3 Additional Term Experiment

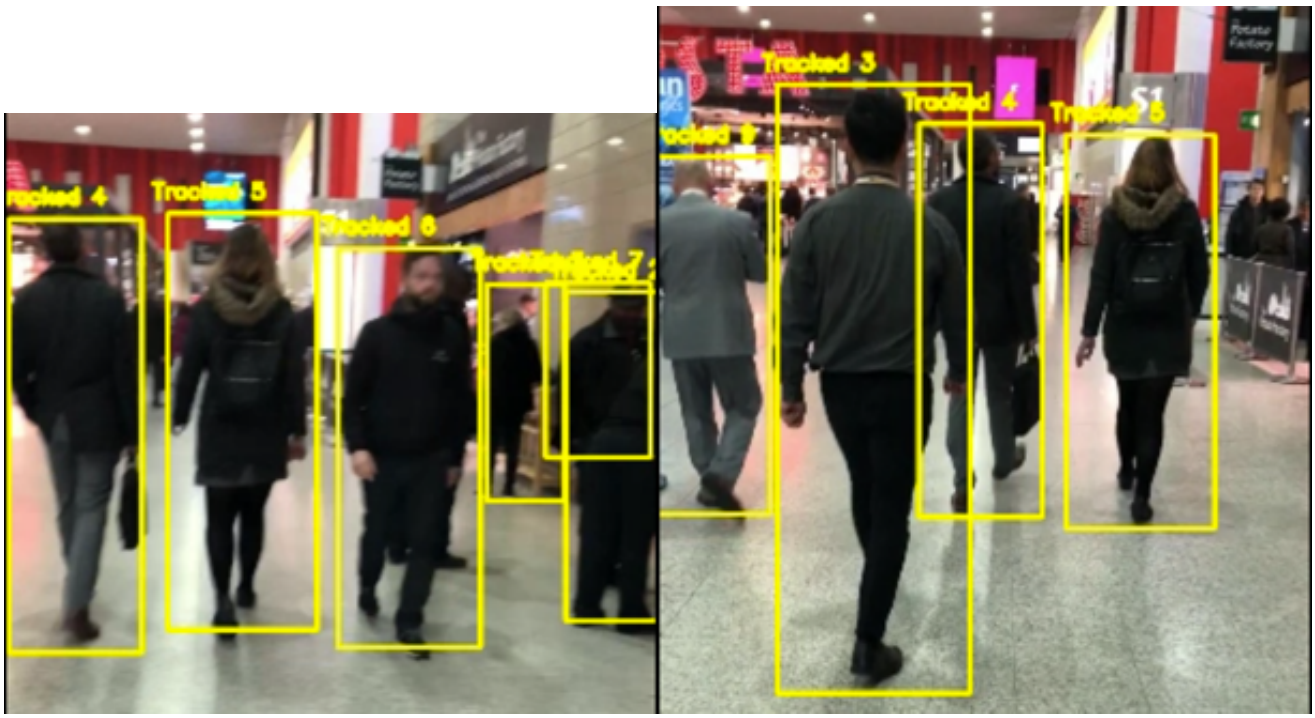
Experiment 2.1 ndThreshold Increase (double)

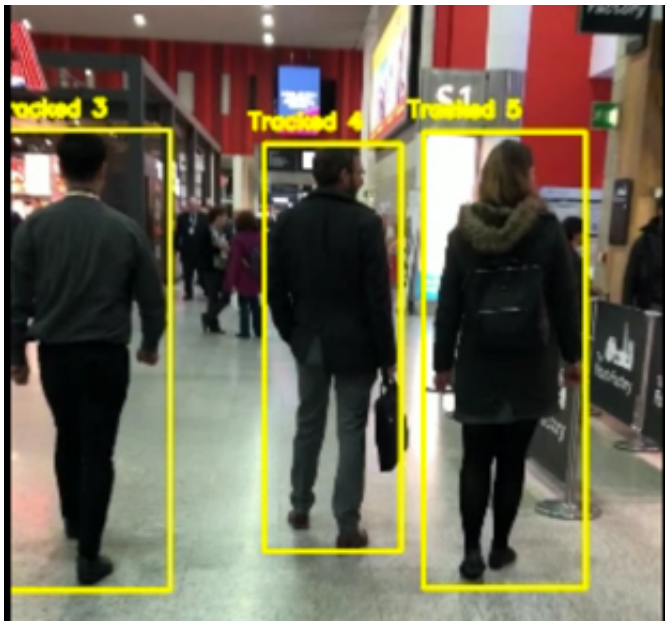
Results



Experiment 2.2 ndThreshold Decrease (degree 3)

Results





Experiment 2.3 Additional Term Experiment

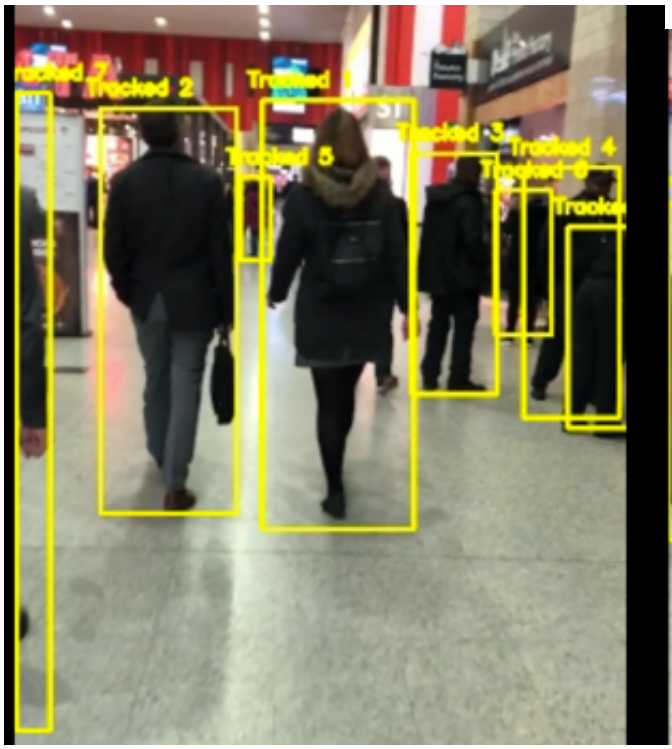
I am doing a very crude experiment to cap the large distances at a max for x and y to see if that can improve the tracking. So for instance if xDist is above 20 it gets capped at 20 and if yDist is above 20 it gets capped at 20 thus shortening the overall distances and forcing the boxCost to recognize boxes as being the same. I tinkered around with this and believe I got decent results below.

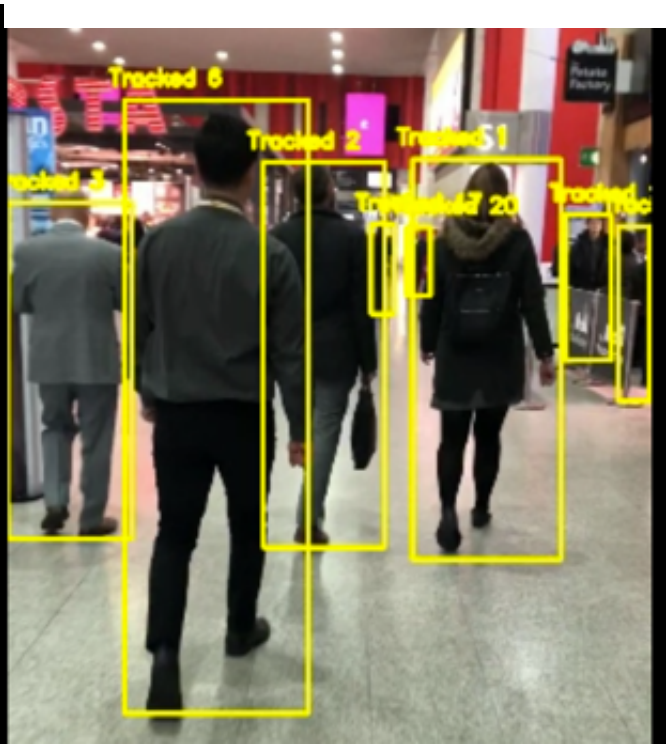
We simply pass x_thresh, y_thresh and a threshold_flag into boxCost().

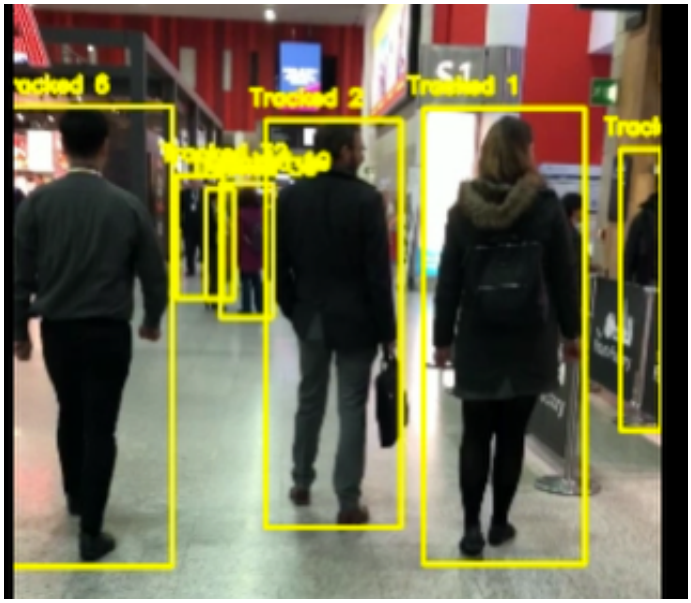
```
# Experiment 2.3: Additional Experiment to cap the distances
# Very Crude
if threshold_flag == True:
    if xDist >= x_thresh:
```

```
            xDist = float(x_thresh)

        if yDist >= y_thresh:

            yDist = float(y_thresh)
```

Results







Analysis

Increasing the ndThreshold to double the value does not improve tracking as a few tracked objects that are occluded or leave the frame have their tracking ids switched to other in-frame objects. Decreasing the ndThreshold by a degree of 3 didn't have much effect. Increasing the ndthreshold by a degree of 1 (to 0.03) caused the algorithm to fail to detect and track any object.

Adding the logic to cap the distances with the thresholding (20,20) and dropping the minBoxArea to 50 * 50

had interesting effects.

- It detected far more objects in the video and tracked 35 separate ones even people in the background and off to the sides.
- The tracker didn't jump around as I expected even though it had more objects to track with lower bounding boxes and smaller distances thus I believe it simply handled the objects better.
- The execution time to generate the video was significantly more than the others

Exercise 3

Rework ***boxCost*** to use Mahalanobis distance instead of Euclidean distance.See https://en.wikipedia.org/wiki/Mahalanobis_distance for details.

Experiment 3.1 Mahalanobis Distance Experiment

Added the following functions for both Euclidean and Mahalanobis Distances.

```
def euclidean_distance(xDist,yDist):
    # square root of x squared plus y squared
    cost = xDist**2 + yDist**2
    cost = math.sqrt(cost)

    return cost

def mahalanobis_distance(box1,box2,):
    covariance_matrix = np.cov(np.vstack([box1,box2]).T)

    inverse_covariance = np.linalg.inv(covariance_matrix)

    # delta = box1 - box2
    # distance = np.dot(np.dot(delta.T, inverse_covariance), delta.T)

    xDist = distance.mahalanobis(box1[0],box2[0], inverse_covariance)
    yDist = distance.mahalanobis(box1[1],box2[1], inverse_covariance)

    # square root of x squared plus y squared
    cost = xDist**2 + yDist**2
    cost = math.sqrt(cost)

    return cost
```

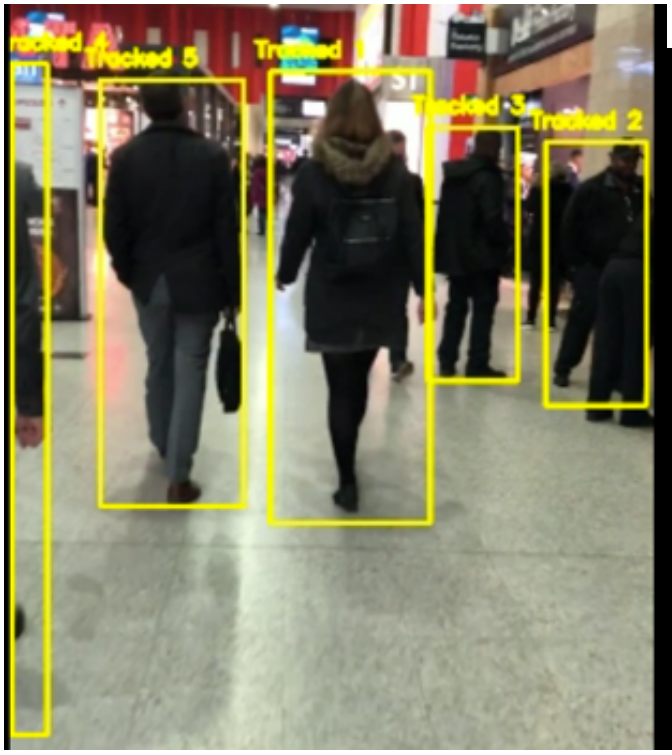
Shout out to Michel for his implementation. I was making mistakes with the distance then figured out after his post

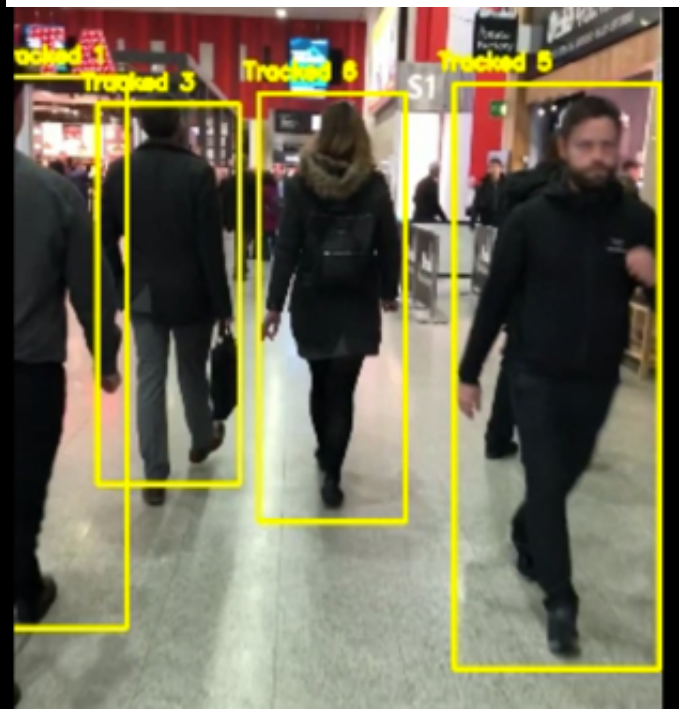
Then it's a simple distance_flag passed into the boxCost() function that drives how we calculate it.

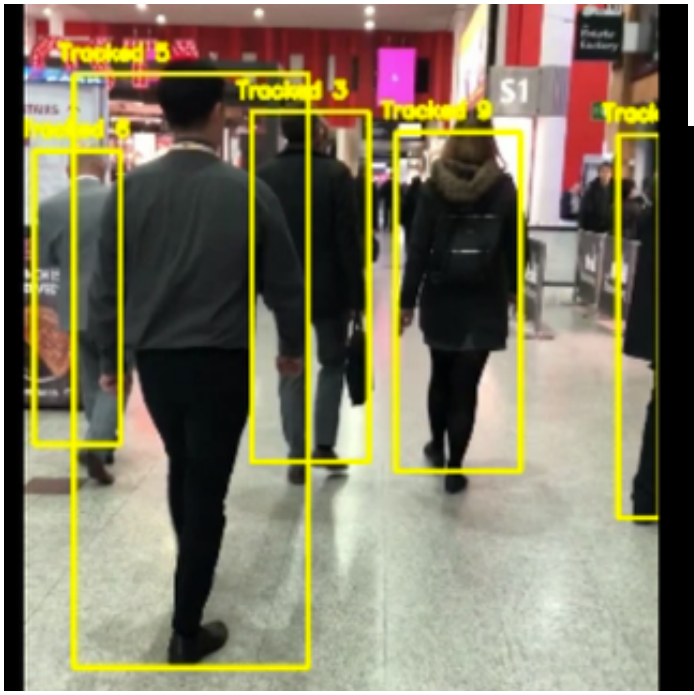
```
# Experiment 3.1: Mahalanobis Distance
if distance_flag = 'mahalanobis':
    cost = mahalanobis_distance(box1,box2)
else:
    cost = euclidean_distance(xDist,yDist)
```

This is a very crude implementation.

Results







Analysis

The Mahalanobis Distance was not as accurate as the Euclidean. In the results you can see the same object gets assigned tracking id 4, 1, 5 and 9. In fact, in 5 and 9 case the algorithm simply lost the tracking without any occlusion or interference from any other object though my implementation may be wrong.

Regards

Brian