Observations

Reference Line Selection

1. The algorithm that was used to count the “entry and exit” is based on simple principle; that is the id of a person assigned should not change before and after the reference line.
2. The observation is that the specified “reference line” for counting “entry and exit” tracks is not feasible as there is a heavy turnstile nearby causing a raise in identity switches.
3. Based on the above facts to get the better results I had to shift the “reference line” based on many iterations. You can find the reference line in the solution video.
4. I used the “yolo-v8-m” model for the detection purpose and “Byte Track” for the tracking purpose. Both of these are available in python libraries named “supervision” and “Ultralytics”.

Object Detection Model

5. One important thing that is overlooked in the tracking problem is the correct usage of object detection model. The stability of the object detections is super important when it comes to tracking. Hence, I used YOLO-v8-m model. I tested with “nano”, “small”, “medium” and “large” versions of YOLO-v8.
6. The problem with the nano model is it is missing some of the detections thereby decreasing the accuracy.
7. The detections from the “small” version model are stable but the accuracy is decreasing a lot compared to medium model. Look at the Table-2

Table-1: For threshold =0.7 and frame_buffer = 30 and processor = i5

Model	FPS	Accuracy
YOLO-v8-n	6	In: 67
YOLO-v8-s	3	In: 83
YOLO-v8-m	1	In: 108

Table-2: For threshold = 0.7 and frame_buffer = 30 and GPU = RTX-3070

Model	FPS	Accuracy
YOLO-v8-n	~43	In: 67
YOLO-v8-s	~38	In: 83
YOLO-v8-m	~34	In: 108
YOLO-v8-L	~26	In: 106

Tracker (Byte Track)

8. Byte Track is a SOA model for the tracking problem and the important feature of this model is that it doesn’t neglect the low threshold. Therefore, we need to specify a threshold parameter that differentiates between high threshold detections vs low detections thresholds.
9. From the original publication it was mentioned that the value of threshold should be 0.8 for giving better results. I had specified 0.8 in the current scenario. I tried with 0.6 and 0.7 values and there are some accuracy issues with those values. Look into the table-3 for more.

10. Frame buffer is another parameter which plays a key role in memorizing the tracker-id for the specified frame number when the object gets occluded or disappeared from the frame. The Byte Track fortunately is lite on the computational power and there is not much accuracy issues in this scenario. So, I took 60 frames. Look into the Table-4 and 5 for more.

    Table-3: Model: yolo-v8-m; track_buffer: 60; GPU: RTX-3070

Threshold	Accuracy
0.7	In: 106
0.8	In: 116
0.9	In: 116

    Table-4: Model= yolo-v8-m; threshold = 0.8; GPU = RTX-3070

Frame_buffer	Accuracy
30	In: 117
60	In: 116

    Table-5: Model= yolo-v8-m; threshold = 0.7; GPU = RTX-3070

Frame_buffer	Accuracy
15	In: 103
30	In: 108
60	In: 106

    Table-6 threshold=0.8; track_buffer-60; GPU = RTX-3070

Model	FPS	Accuracy
YOLO-v8-s	38	In: 100
YOLO-v8-m	34	In: 116

Disadvantages:

1. The line that defined in the current scenario is cut throat; changing the surrounding slightly will cause a lot of accuracy issues. For example, the change position of security guard.
2. The script that was written is dependent a lot on the API calls rather than the core models. Hence there is not much control over the outputs that we want. For instance, the libraries installed are dependent on the hardware; I cannot control the “cpu” and “cuda” with in the same machine. Detection threshold cannot be controlled in this code.

Final Thoughts:

1. The total number of **entries are 116** based on the reference line that I’ve chosen.
2. The total number of **exits are 3** based on the reference line that I’ve chosen.
3. You can find the solution video here.