

Object Tracking

Chapter Goals

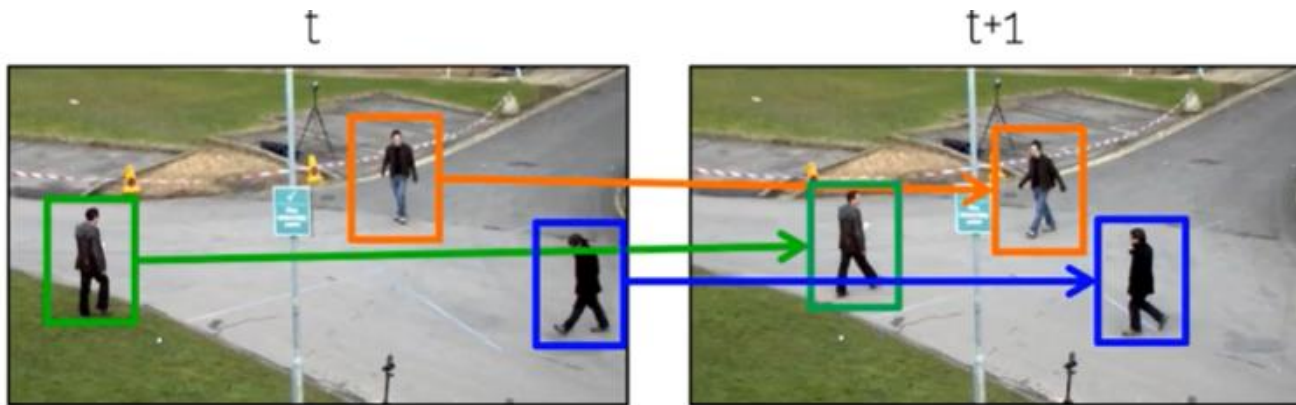
After completing this chapter, you should be able to understand :

- what is object tracking and its types
- Applications and limitations of object tracking
- Multiple Object Tracking: DeepSort
- Multiple Object Tracking: ByteTrack

What is Object Tracking

Object tracking is the process of **locating a moving object over time** in a sequence of frames

- Given a video, find out which parts of the image depict the same object in different frames
- Often, we use detectors as starting points

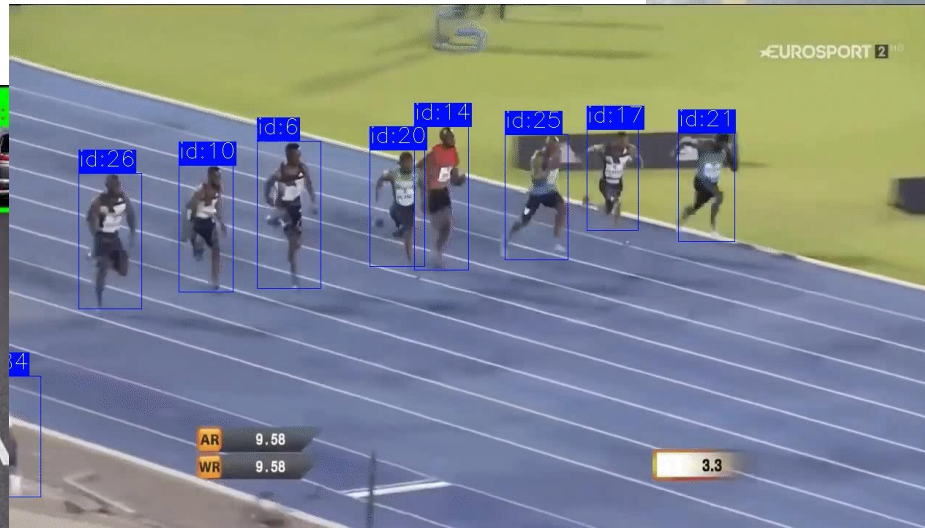
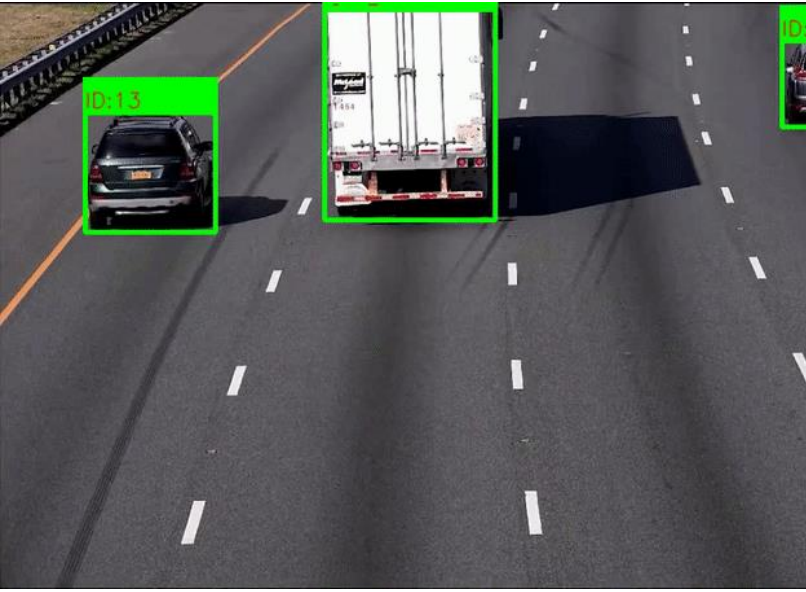


Why do we need tracking?

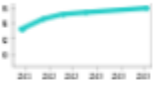
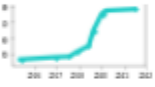



- 1. Temporal Understanding:** Detection alone is static — it only tells you where an object is at a single point in time. Tracking adds motion — it tells you where it moves, enabling temporal reasoning like speed, direction, and patterns.
- 2. Maintaining Object Identity:** In crowded scenes with multiple objects (e.g., people, cars), tracking helps assign consistent IDs to objects, so you know it's the same person from frame to frame.
- 3. Efficiency Over Re-detection:** Running full object detection in every frame is computationally expensive. Tracking is usually faster, as it updates object positions based on previous states rather than re-computing everything from scratch.

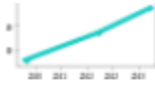
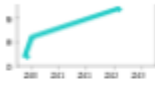

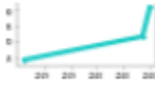

Applications of Object Tracking

- Military and security applications visual surveillance
- Traffic monitoring
- Pedestrian tracking
- Retail shoppers tracking
- Human-computer interaction
- Video editing
- Sports analysis



Tracking datasets

| Trend | Dataset | Best Model |
|---|------------|------------|
|  | MOT17 | UCMCTrack |
|  | MOT16 | PPTracking |
|  | DanceTrack | MOTRv2 |
|  | MOT20 | SMILEtrack |
|  | SportsMOT | Deep-EIoU |





| | | |
|---|-------------|----------|
|  | TAO | GLEE-Pro |
|  | Wildtrack | MVFlow |
|  | MOTS20 | ReMOTS |
|  | 2D MOT 2015 | GSDT |
|  | HiEve | CSTrack |

Tracking datasets: MOT

Multiple Object Tracking Benchmark

| Release | # Seq. | BBs | Persons | Length | Density | Tracks | HD |
|--------------|--------|--------|---------|--------|---------|--------|-------|
| <i>MOT15</i> | 22 | 101349 | 101349 | 16:36 | 8.98 | 1221 | 31.8% |
| <i>MOT16</i> | 14 | 476532 | 292733 | 07:43 | 26.05 | 1276 | 85.7% |



| Sample | Name | FPS | Resolution | Length | Tracks | Boxes | Density | Description |
|--|----------|-----|------------|-----------------------|--------|---------|---------|---|
|  | MOT20-05 | 25 | 1654x1080 | 3315 (02:13) | 1211 | 751330 | 226.6 | Crowded square by night time. |
|  | MOT20-03 | 25 | 1173x880 | 2405 (01:36) | 735 | 356728 | 148.3 | People leaving entrance of stadium by night time, elevated viewpoint. |
|  | MOT20-02 | 25 | 1920x1080 | 2782 (01:51) | 296 | 202215 | 72.7 | Crowded indoor train station. |
|  | MOT20-01 | 25 | 1920x1080 | 429 (00:17) | 90 | 26647 | 62.1 | Crowded indoor train station. |
| Total | | | | 8931 frm. (357 s.) | 2332 | 1336920 | 149.7 | |

Leal-Taixé, Laura, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. "Tracking the state of the art in multiple object tracking." arXiv (2017).

Milan, Anton, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. "MOT16: A benchmark for multiple object tracking." arXiv (2016).

- A large collection of datasets, some already in use and some new challenging sequences
- Detections for all the sequences.
- A common evaluation tool providing several measures, from recall to precision to running time.
- An easy way to compare the performance of state-of-the-art tracking methods.
- Several challenges with subsets of data for specific tasks such as 3D tracking, surveillance, sports analysis

Performance Metrics

Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML)

Each trajectory can be classified as mostly tracked (MT), partially tracked (PT), and mostly lost (ML).

A target is **mostly tracked** if it is successfully tracked for **at least 80%** of its life span, **mostly lost** if it is successfully tracked for **at most 20%**. All other targets are partially tracked.

MOTA (Multiple Object Tracking Accuracy)

The MOTA is the most used summary metric for MOT. It is defined as follows:

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t}$$

where **FN_t** is the number of false negatives (missed detections), **FP_t** the number of false positives (**ghost trajectories**), **IDS_t** the number of **identity switches at time t**. A target is considered missed if the **IoU** with the ground truth is inferior to a given threshold. (Note that the MOTA can be negative.)

The community usually reports MOTA and Mostly Tracked to evaluate performance.

Performance Metrics

What is IDF1?

IDF1 (ID F1 Score) is a key metric in evaluating Multiple Object Tracking (MOT) systems. It measures how well a tracker maintains consistent identities over time. Whereas metrics like MOTA evaluate how many objects are correctly detected, IDF1 tells you how consistently the same identity is assigned to each object across frames.

$$\text{IDF1} = \frac{2 \cdot \text{IDTP}}{2 \cdot \text{IDTP} + \text{IDFP} + \text{IDFN}}$$

Where: IDTP = Identity True Positives (correctly matched identities across time)
IDFP = Identity False Positives (predicted identities that do not match ground truth)
IDFN = Identity False Negatives (missed matches of ground truth identities)

Goal: A **high IDF1** score (closer to 1) means the tracker **preserved identity well** across the sequence.

Why Is IDF1 Important?

In many applications—like video surveillance, sports analytics, autonomous driving, or social behavior analysis—knowing that you're tracking the same person or object over time is critical. A tracker with high MOTA but low IDF1 may detect all objects but frequently assign wrong IDs. A tracker with high IDF1 reliably keeps the same ID for each object over time, even if a few objects are missed.

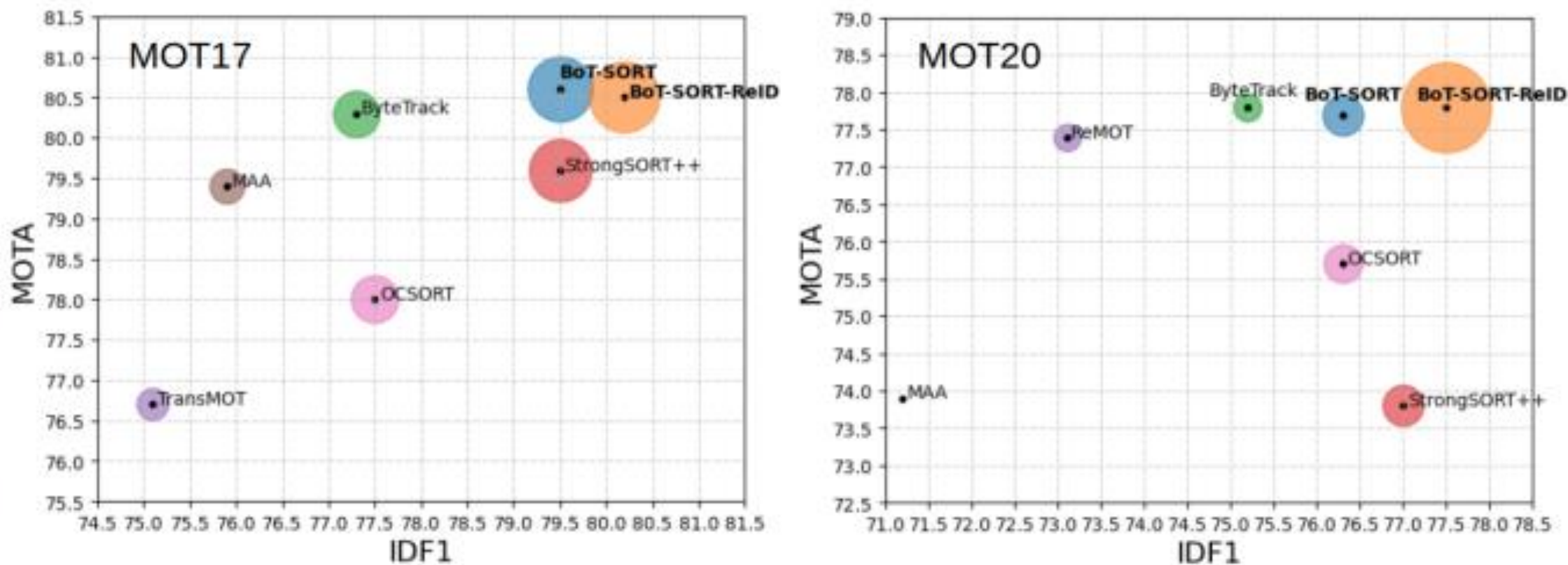


Figure 1: IDF1-MOTA-HOTA comparisons of state-of-the-art trackers with our proposed BoT-SORT and BoT-SORT-ReID on the MOT17 and MOT20 test sets. The x-axis is IDF1, the y-axis is MOTA, and the radius of the circle is HOTA. Ours BoT-SORT-ReID and our simpler version BoT-SORT achieve the best IDF1, MOTA, and HOTA performance from all other trackers on the leaderboards.

Object Tracking vs Object Detection

How is tracking different from detection?

- There is a tight **relationship** between tracking and detection.
- Detection consists in **locating** one or several objects in a given image
- whereas the goal of tracking is to locate these objects **throughout a whole video, keeping track of which object is which** along the video frames.
- In order to track an object, you first need to provide the image of the said object to the tracking algorithm, and this is either done by a detection algorithm (Detection-based trackers) or manually (Detection-free trackers).

Challenges for Object Tracking

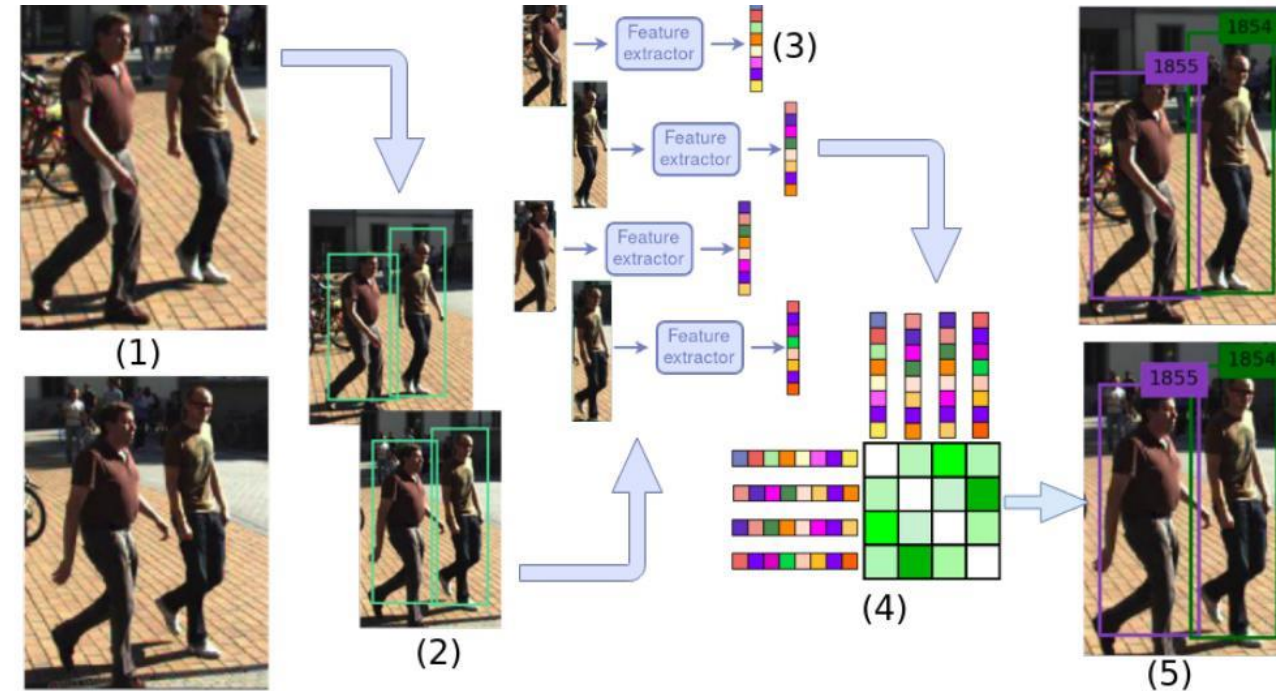
- Non-Linear motion, difficult and rapid motion
- Limited resolution , *Video captured from a low-end phone*
- Scene ambiguity: Similar objects in the scene, *Same colour of clothes, accessories ...*
- Highly crowded scenarios like streets, concerts, stadiums, markets
- occlusion
- Scale, illumination and change of appearance



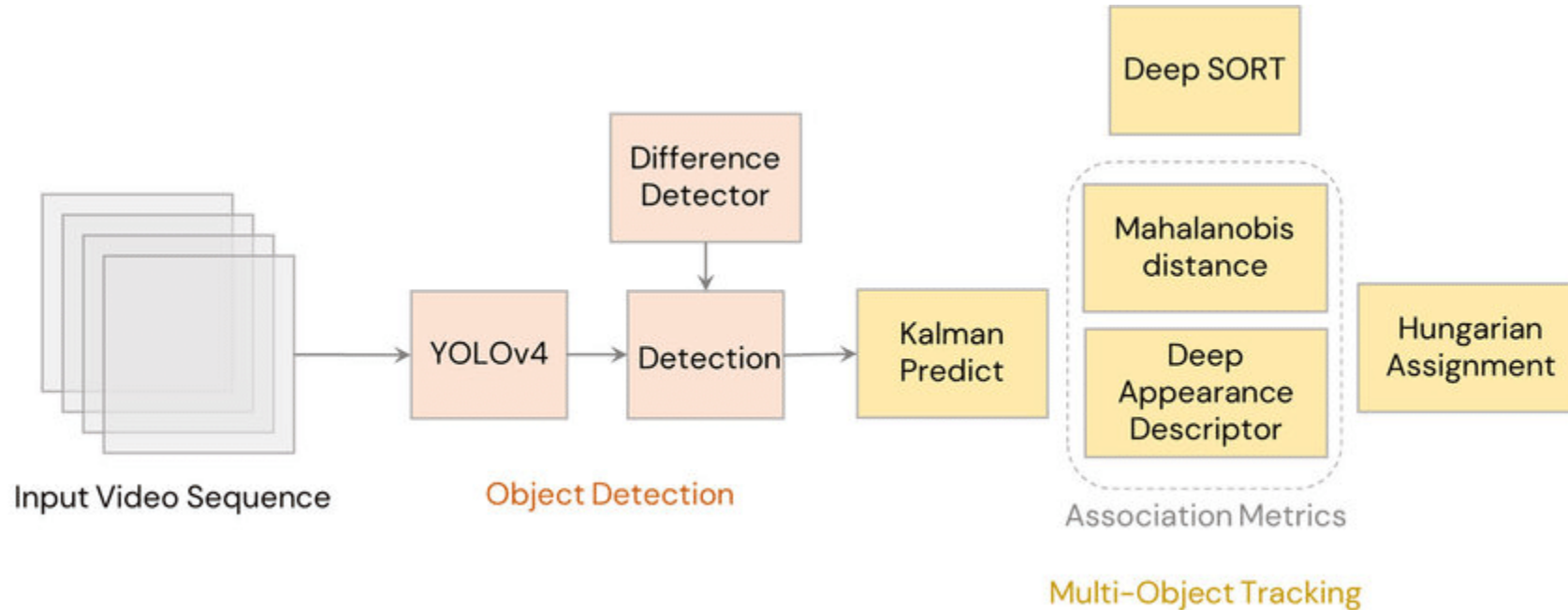
Multiple Object Tracking

DeepSORT (Simple Online and Realtime Tracking)

- **Object Detection** Detect objects (e.g., people) in each frame using an external detector like YOLO, Faster R-CNN, or SSD.
- **Feature Extraction** For each detected object, extract an appearance feature vector using a deep CNN (e.g., a pretrained ReID model).
- **Kalman Filtering** Predict the position of each existing tracked object in the next frame using a Kalman filter (motion model).
- **Data Association (Matching)** Match new detections to existing tracks using a combined metric:
 - Motion (e.g., IoU or Mahalanobis distance from Kalman filter)
 - Appearance (cosine distance between feature vectors)
 Use the Hungarian algorithm for optimal assignment.



DeepSORT (Simple Online and Realtime Tracking)

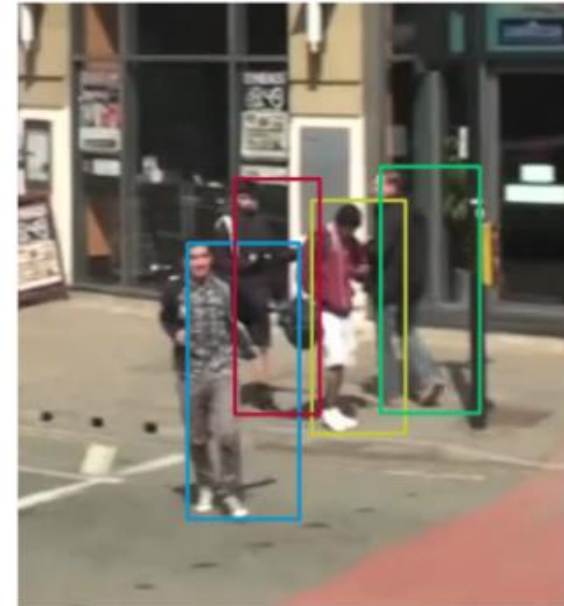


DeepSORT uniquely identifies and tracks objects by employing an advanced association metric that integrates both motion and appearance descriptors. This integration minimizes identity switches, promoting more efficient and reliable tracking. By considering not just velocity and motion, but also object appearance, DeepSORT stands out as a comprehensive tracking algorithm in the realm of computer vision

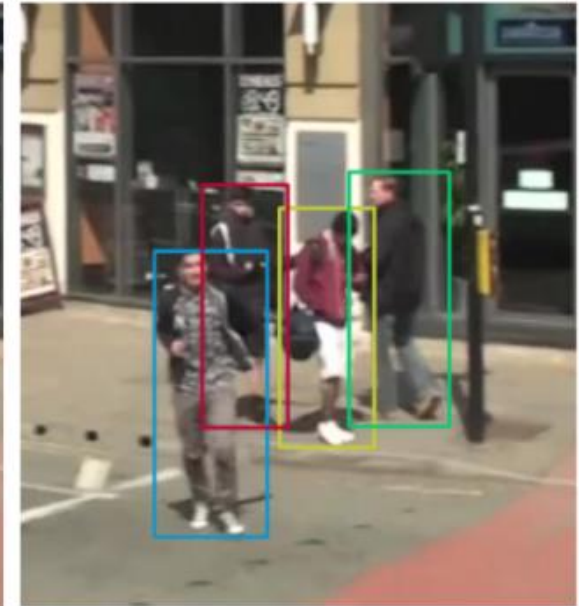
BotSORT

- DeepSORT: Uses appearance similarity to recover identities after occlusion.
- BotSORT: Uses a bottom-up strategy to associate fragmented tracklets, improving robustness to occlusions and reducing identity switches by merging short reliable tracklets.

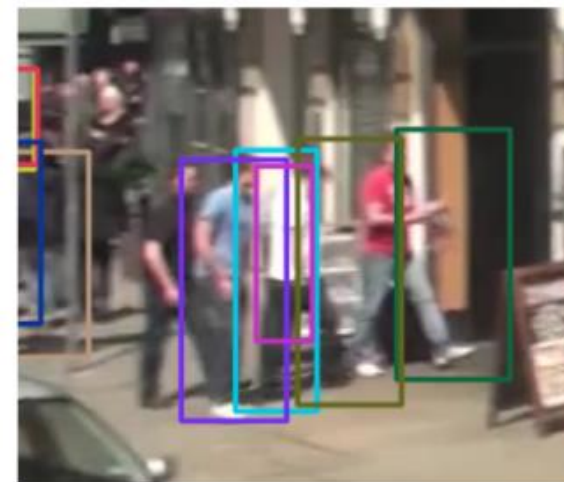
| Aspect | DeepSORT | BotSORT |
|---------------------|------------------------------|--------------------------------|
| Appearance Features | Yes (deep CNN embeddings) | Minimal or none |
| Tracking Approach | Kalman + appearance matching | Bottom-up tracklet association |
| Occlusion Handling | Appearance + motion | Tracklet merging |
| Speed & Efficiency | Slower | Faster |



(a.1) predicted BB **before** CMC.



(a.2) predicted BB **after** CMC.



(b.1) predicted BB **before** CMC.



(b.2) predicted BB **after** CMC.

BotSORTReID

- An extension of BotSORT that integrates ReID (Re-Identification) appearance features for improved data association.
- Combines the bottom-up tracklet association strategy with deep appearance embeddings (from ReID models) to better distinguish identities, especially in crowded or occluded scenes.
- Provides higher tracking accuracy by leveraging both motion and appearance.

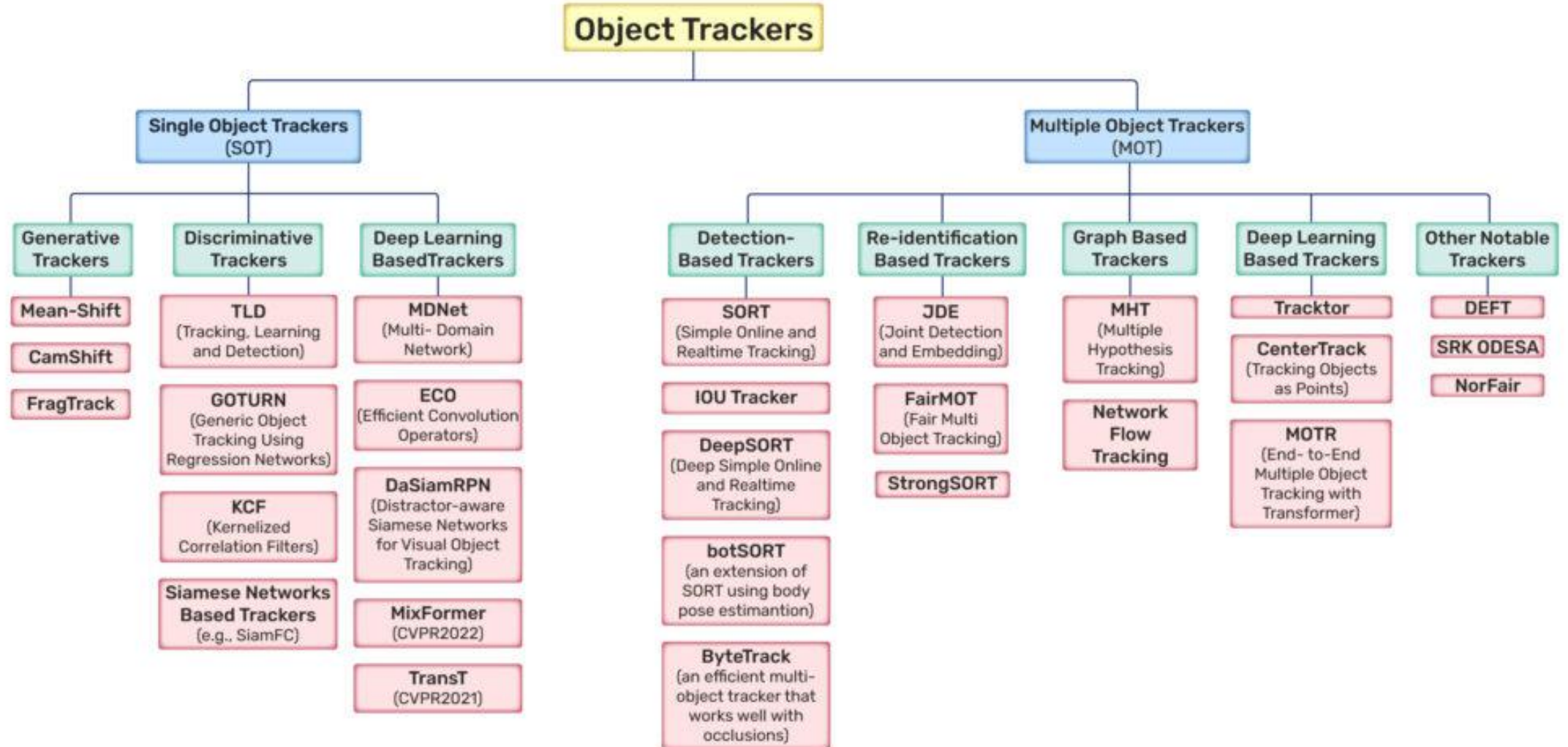
| Feature | BotSORT | BotSORTReID |
|---------------------|---------------------------|--|
| Appearance Features | Minimal or none | Uses deep ReID appearance features |
| Tracking Approach | Motion + spatial cues | Motion + spatial + ReID features |
| Use Case | Faster, lightweight | More accurate, handles occlusions better |
| Performance | Good for simple scenarios | Better in complex, crowded scenes |

Other Deep Learning Trackers

| Tracker | MOTA ↑ | IDF1 ↑ | FPS ↓ |
|-----------|-------------|-------------|-----------|
| SORT | 74.6 | 76.9 | 30 |
| DeepSORT | 75.4 | 77.2 | 13 |
| FairMOT | 77.2 | 79.8 | 25.9 |
| TransMOT | 76.7 | 75.1 | 9.6 |
| BYTETrack | 80.3 | 77.3 | 30 |

From “FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking”
<https://arxiv.org/pdf/2004.01888.pdf>

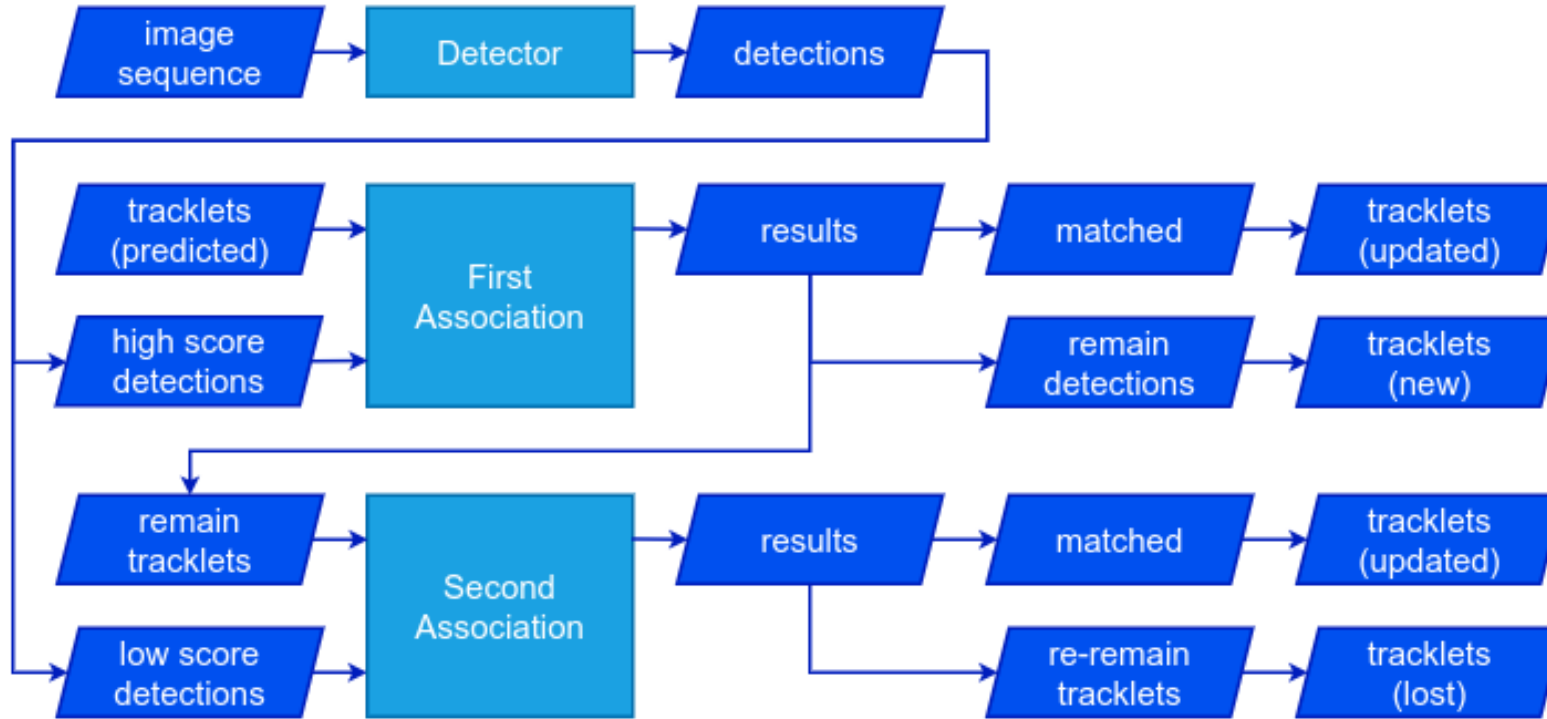
DeepSORT Demo



ByteTRACK

ByteTRACK

ByteTrack: Multi-Object Tracking by Associating Every Detection Box



© 2023 Luffca Inc.

ByteTrack improves performance by devising data association algorithms. The idea is to perform data association using all detection boxes, as the title of the paper says, “Associating Every Detection Box”. However, according to the implementation, data association is performed using detection boxes with a score higher than 0.1, so we feel that it is better to say that almost every detection box is suitable for the implementation.

Object Detection

Detect all objects in the frame using a high-quality detector like YOLOX. Include both high- and low-confidence detections.

Track High-Confidence Detections First

Match existing tracks with high-confidence detections only using IoU and the Hungarian algorithm, updating matched tracks.

Handle Low-Confidence Detections

Attempt to match unmatched tracks from step 2 with low-confidence detections, which helps recover missed or occluded objects.

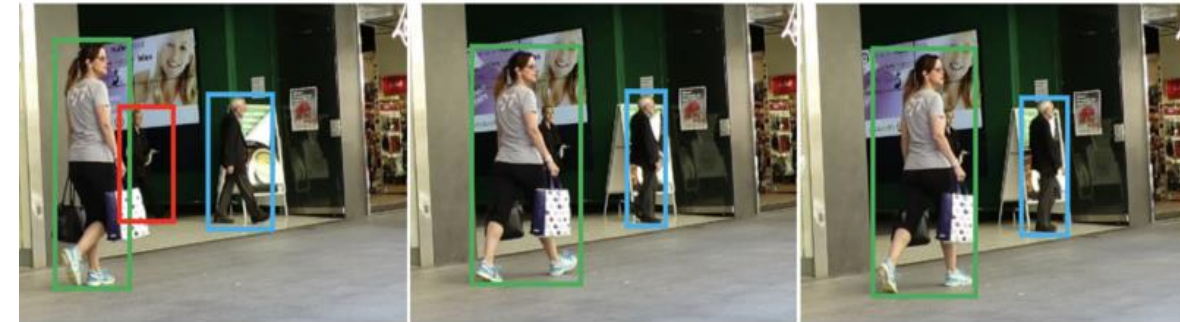
Track Lifecycle Management

Initialize new tracks from unmatched high-confidence detections, mark unmatched tracks as lost, and delete tracks that have been inactive too long.

Key Idea: Unlike other trackers, ByteTrack uses low-confidence detections wisely to improve robustness without relying on deep appearance features.



(a) detection boxes

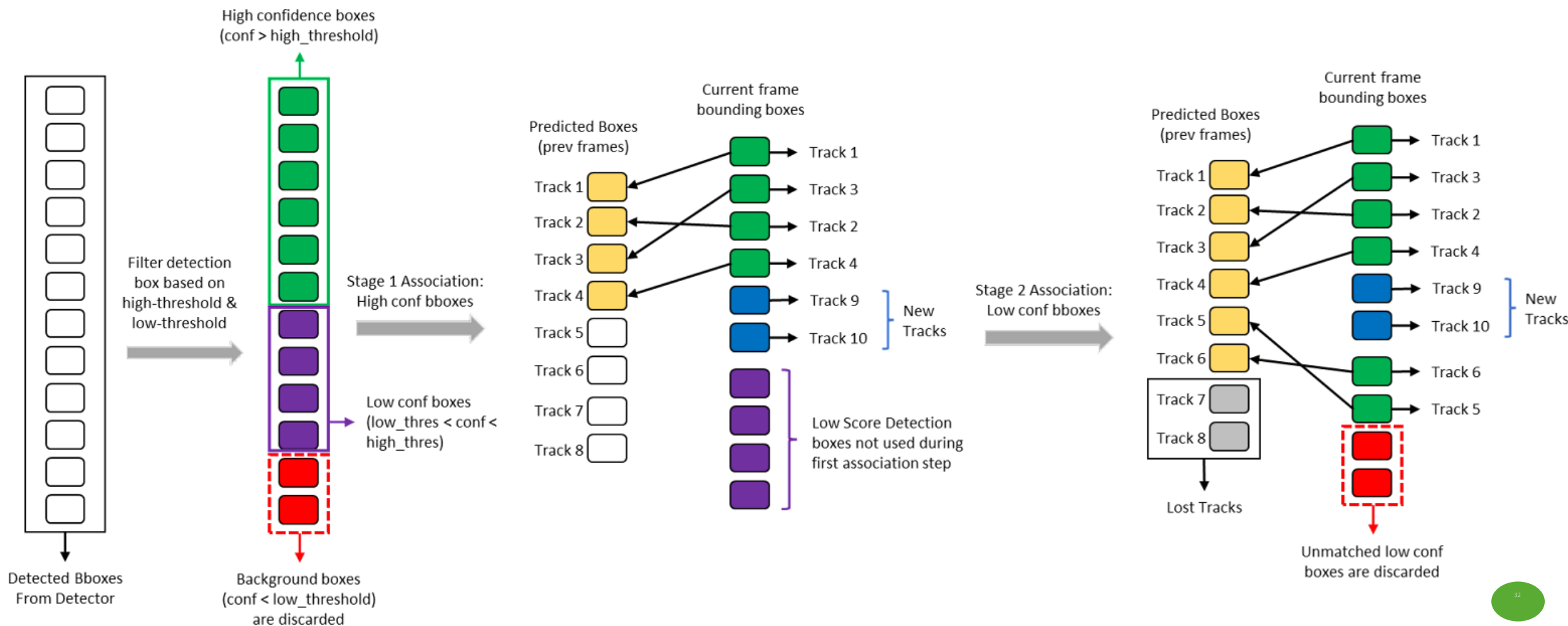


(b) tracklets by associating high score detection boxes



(c) tracklets by associating every detection box

ByteTRACK



Chapter Summary

- We saw what is object tracking and its types
- We saw some applications and limitations of object tracking
- We saw an example of Multiple Object Tracking: DeepSort
- We finished by an example of Multiple Object Tracking: ByteTrack

Hungarian Method:- Example I

| | Job1 | Job2 | Job3 | Job4 | |
|--------|------|------|------|------|-------|
| Crane1 | 4 | 2 | 5 | 7 | min=2 |
| Crane2 | 8 | 3 | 10 | 8 | min=3 |
| Crane3 | 12 | 5 | 4 | 5 | min=4 |
| Crane4 | 6 | 3 | 7 | 14 | min=3 |



| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 2 | 0 | 3 | 5 |
| Crane2 | 5 | 0 | 7 | 5 |
| Crane3 | 8 | 1 | 0 | 1 |
| Crane4 | 3 | 0 | 4 | 11 |

min: 2 0 0 1



- 1) Find the minimum of each row, and subtract from each row the min value.
- 2) Find the minimum of each column, and subtract from each column the min value.
- 3) Find the min number of vertical/horizontal lines

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 0 | 3 | 4 |
| Crane2 | 3 | 0 | 7 | 4 |
| Crane3 | 6 | 1 | 0 | 0 |
| Crane4 | 1 | 0 | 4 | 10 |

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 0 | 3 | 4 |
| Crane2 | 3 | 0 | 7 | 4 |
| Crane3 | 6 | 1 | 0 | 0 |
| Crane4 | 1 | 0 | 4 | 10 |

If the number of lines is equal to m , the optimal solution is available among the Covered Zeros. Otherwise, proceed to Step four.

$$3 \neq m = 4$$

c) find the min of uncovered values. Then, subtract the min from all the uncovered values and add it to the corner points. Then go back to Step 3.

go back to step 3: $3 \neq m = 4$

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 0 | 0 | 1 |
| Crane2 | 3 | 0 | 4 | 1 |
| Crane3 | 9 | 4 | 0 | 0 |
| Crane4 | 1 | 0 | 1 | 7 |

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 0 | 0 | 1 |
| Crane2 | 3 | 0 | 4 | 1 |
| Crane3 | 9 | 4 | 0 | 0 |
| Crane4 | 1 | 0 | 1 | 7 |

min = 1

minimum number of lines required = 4 = m ✓
 So, we are at the optimal table

Start the assignment from the row or column that has minimum number of zeros.

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 1 | 0 | 1 |
| Crane2 | 2 | 0 | 3 | 0 |
| Crane3 | 9 | 5 | 0 | 0 |
| Crane4 | 0 | 0 | 0 | 6 |

| | Job1 | Job2 | Job3 | Job4 | |
|--------|------|------|------|------|---|
| Crane1 | 0 | 1 | 0 | 1 | 2 |
| Crane2 | 2 | 0 | 3 | 0 | 2 |
| Crane3 | 9 | 5 | 0 | 0 | 2 |
| Crane4 | 0 | 0 | 0 | 6 | 3 |
| | 2 | 2 | 3 | 2 | |

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 0 | 0 | 1 |
| Crane2 | 3 | 0 | 4 | 1 |
| Crane3 | 9 | 4 | 0 | 0 |
| Crane4 | 1 | 0 | 1 | 7 |

min = 1

minimum number of lines required = 4 = m ✓
 So, we are at the optimal table

Start the assignment from the row or column that has minimum number of zeros.

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 1 | 0 | 1 |
| Crane2 | 2 | 0 | 3 | 0 |
| Crane3 | 9 | 5 | 0 | 0 |
| Crane4 | 0 | 0 | 0 | 6 |

| | Job1 | Job2 | Job3 | Job4 | |
|--------|------|------|----------------|----------------|----------------|
| Crane1 | 0 | 1 | 0 | 1 | 2 |
| Crane2 | 2 | 0 | 3 | 0 | 2 |
| Crane3 | 9 | 5 | 0 | 0 | 2 |
| Crane4 | 0 | 0 | 0 | 6 | 3 1 |
| | 2 | 2 | 3 2 | 1 1 | |

This is the final assignment:

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 0 | 1 | 0 | 1 |
| Crane2 | 2 | 0 | 3 | 0 |
| Crane3 | 9 | 5 | 0 | 0 |
| Crane4 | 0 | 0 | 0 | 6 |

$$X_{11} = X_{22} = X_{34} = X_{43} = 1$$

Find the Cost associated to these assignments in the original cost table.

| | Job1 | Job2 | Job3 | Job4 |
|--------|------|------|------|------|
| Crane1 | 4 | 2 | 5 | 7 |
| Crane2 | 8 | 3 | 10 | 8 |
| Crane3 | 12 | 5 | 4 | 5 |
| Crane4 | 6 | 3 | 7 | 14 |

$$Z^* = 4 + 3 + 5 + 7 = 19$$

$$\boxed{Z^* = 19}$$