Project work for **Machine Learning for Computer Vision**
Master's Degree in Artificial Intelligence
University of Bologna, Bologna, Italy

Jana Nikolovska

MOT on DanceTrack

Introduction

- **Idea:** Track multiple objects over time (with identity preservation)
- **Goal:** Show different approaches, compare them and progressively improve association quality and identity consistency

- **Dataset:** DanceTrack
  - Highly crowded dance scenes
  - Frequent occlusions
  - Similar appearance across targets

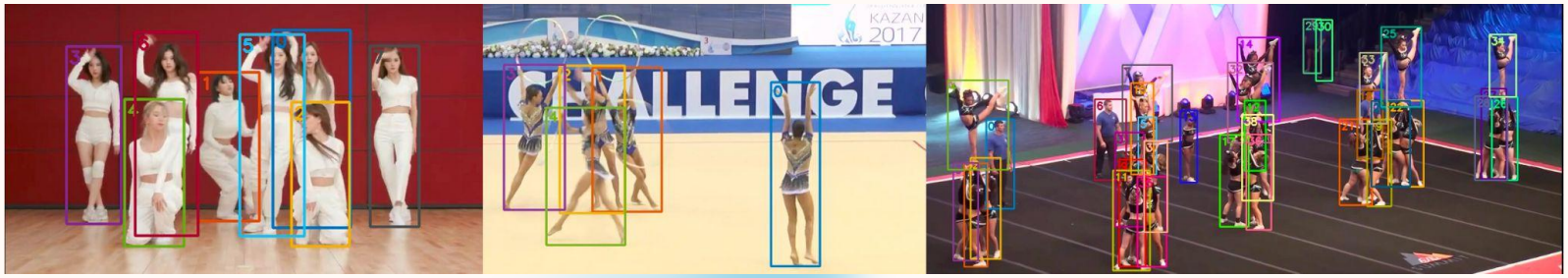


Table of contents

Detector

- Chose **YOLOv8n** for its **strong speed–accuracy trade-off** (e.g., F1 up to 0.749 @ IoU 0.25) and real-time suitability
- Used the pretrained model **without fine-tuning** to keep detection fixed and focus on tracking performance.
- **Precomputed detections once and reused them** for B0, B1, and B2 to **avoid redundancy** and ensure fair comparison.

Baseline Online Tracker (B0)

- Motion prediction between frames
- IoU-based matching for data association
- Simple and fast
- Stable when motion is smooth
- Fragile to occlusion
- Struggles with overlapping objects

B0 algorithm pseudo code

```
function STEP_B0(tracks, det_boxes,
frame_id):
    for t in tracks: t.age += 1

    iou = IoU_matrix(tracks.boxes,
det_boxes)
    matches = GREEDY_MATCH(iou,
threshold = iou_thresh)
      # returns pairs (track_i, det_j)

    for (ti, dj) in matches:
        UPDATE_TRACK(tracks[ti],
det_boxes[dj], frame_id)
        assigned_id[dj] =
tracks[ti].track_id
    for each detection dj not matched:
        new_track = INIT_TRACK(next_id,
det_boxes[dj], frame_id)
        tracks.append(new_track)
        assigned_id[dj] = next_id
        next_id += 1

    tracks = [t for t in tracks if t.age
<= max_age]
    return tracks, assigned_id
```

Two-stage association (B1)

- Inspired by ByteTrack association strategy
- Split detections into high-confidence and low-confidence sets
- First match high-confidence detections
- Use low-confidence detections to recover unmatched tracks
- Only high-confidence detections spawn new tracks
- Fewer false track births - from low-confidence detections
- More stable matching priority
- Still purely geometric

B1 algorithm pseudo

```
function STEP_B1(tracks, det_boxes,
det_scores, frame_id):
    for t in tracks: t.age += 1

    high = {dets with score >=
high_thresh}
    low  = {dets with low_thresh <=
score < high_thresh}

    # Stage 1: tracks -> high detections
    iou1 = IoU_matrix(tracks.boxes,
high.boxes)
    matches1, unmatched_tracks,
unmatched_high = GREEDY_MATCH(iou1,
iou_thresh_high)
    APPLY_MATCHES(tracks, high,
matches1)

    # Stage 2: remaining tracks -> low
detections
    if unmatched_tracks not empty AND
low not empty:
        iou2 =
IoU_matrix(tracks[unmatched_tracks].boxe
s, low.boxes)
        matches2 = GREEDY_MATCH(iou2,
iou_thresh_low)
        APPLY_MATCHES(tracks, low,
matches2)

    # Spawn NEW tracks only from
unmatched HIGH detections
    for each det in unmatched_high:
        spawn new track with new id

    prune tracks with age > max_age
    return tracks, assigned_id
```

Identity Preservation with Appearance (B2)

- Introduce appearance similarity for association
- Applied only in ambiguous cases
- Used for re-identification after occlusion
- Motion and IoU remain primary cues
- Can reconnect after spatial jump
- Especially useful when camera moves
- But, If people look similar -> identity swaps.

B2 algorithm pseudo (1)

```
function STEP_B2(tracks, det_boxes,
det_scores, image, frame_id):
    # Stage 0: age
    for t in tracks: t.age += 1

    # Stage 1 & 2: same as B1
    assigned_id = -1 for all dets
    do B1 matching to fill assigned_id
and update tracks
    unmatched_tracks_after_iou = tracks
still not matched
    free_detections = detections with
assigned_id == -1
```

B2 algorithm pseudo (2)

```
    # Stage 3: ReID only for "old"
unmatched tracks
    reid_tracks = {ti in
unmatched_tracks_after_iou where
tracks[ti].age >= reid_age}

    if reid_tracks not empty and
free_detections not empty:
        det_emb = EMBEDDINGS(image crops
of free_detections)
        tr_emb  = EMBEDDINGS(from cached
track emb or crop current track box)

        cost[i,j] = 1 -
cosine_similarity(tr_emb[i], det_emb[j])
        # greedy on flattened cost list
        pairs = sort all (i,j) by cost
asc
        for (i,j) in pairs:
            if cost[i,j] >
reid_cos_thresh: break
            if track i already used or
det j already used: continue
            MATCH track_i with det_j
(update box, reset age, hits++)
            cache embedding into
track.emb
            assigned_id[det_j] =
track.track_id

    # Spawn new tracks from unmatched
HIGH detections (same idea as B1)
    spawn from unmatched_high

    prune tracks with age > max_age
    return tracks, assigned_id
```

Evaluation protocol

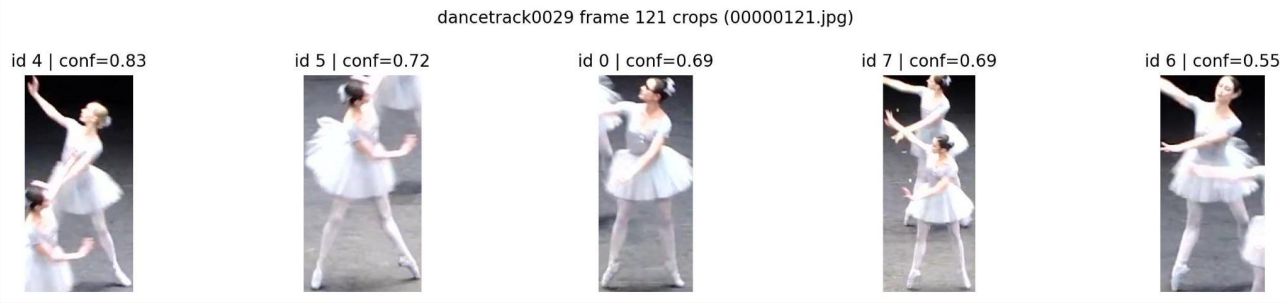
- Evaluation with M
- **Recall** – **How much** of the ground-truth trajectories **did we successfully track**?
    - Range: 0–1 (higher is better)
- **IDF1** – Did we keep the **correct identity** for each object over time?
    - Low IDF1 -> keep switching IDs
    - Range: 0–1 (higher is better)
- **Identity Switches (IDSW)** – How many times did we **assign the wrong identity** to an object?
    - Counts incorrect ID reassignments
    - Lower is better
- **Fragmentation (FRAG)** – How many times did we **interrupt tracking** of the same object?
    - Lower is better
- Qualitative visualizations for long-term tracking behavior

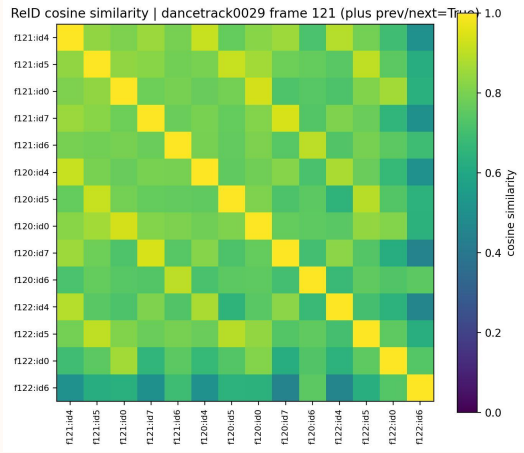
Results

	Recall	IDF1	IDSW	FRAG	Key Insight
B0	0.591	0.258	8977	15576	Weak identity stability
B1	0.540	0.319	3440	12212	Huge improvement
B2	0.558	0.317	4143	13276	Slightly worse than B1

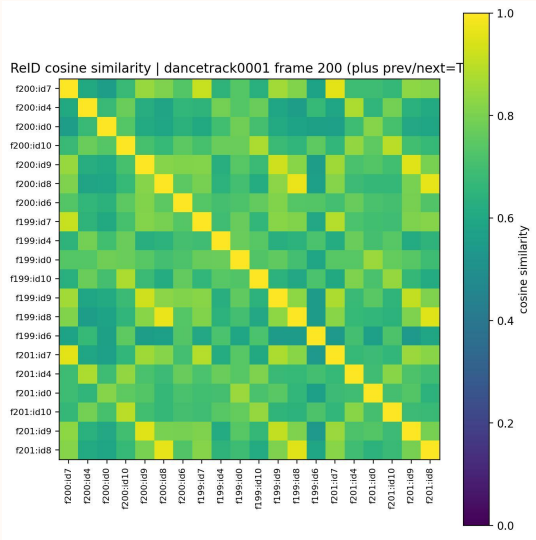
We notice:

- **IDSW** reduced by more than 50%, **FRAG** significantly reduced, **IDF1** jumps from 0.25 → 0.32 when changing **from B0 to B1**
- Similar (slightly worse) performance when switching **from B2 to B1** -> **Appearance embeddings may not be very discriminative in this dataset**
- The **average runtime** per frame is **17.78 ms for YOLO** detection, **6.41 ms for the B0** tracker, **6.44 ms for the B1** tracker, and **8.74 ms for the B2** tracker, showing that adding ReID in B2 introduces a noticeable computational overhead compared to B0 and B1


dancetrack0029 frame 121 crops (00000121.jpg)
id 4 | conf=0.83
id 5 | conf=0.72
id 0 | conf=0.69
id 7 | conf=0.69
id 6 | conf=0.55


ReID cosine similarity | dancetrack0029 frame 121 (plus prev/next=True)
f121:id4
f121:id5
f121:id0
f121:id7
f121:id6
f120:id4
f120:id5
f120:id0
f120:id7
f120:id6
f122:id4
f122:id5
f122:id0
f122:id6
cosine similarity
1.0
0.8
0.6
0.4
0.2
0.0


dancetrack0001 frame 200 crops (00000200.jpg)
id 7 | conf=0.84
id 4 | conf=0.82
id 0 | conf=0.79
id 10 | conf=0.77
id 9 | conf=0.69
id 8 | conf=0.58
id 6 | conf=0.43


ReID cosine similarity | dancetrack0001 frame 200 (plus prev/next=T
f200:id7
f200:id4
f200:id0
f200:id10
f200:id9
f200:id8
f200:id6
f199:id7
f199:id4
f199:id0
f199:id10
f199:id9
f199:id8
f199:id6
f201:id7
f201:id4
f201:id0
f201:id10
f201:id9
f201:id8
cosine similarity
1.0
0.8
0.6
0.4
0.2
0.0

Thank you for your time!

Questions?