Supplementary Material: Simple Cues Lead to a Strong Multi-Object Tracker

Jenny Seidenschwarz[1*] Guillem Brasó [12] Victor Castro Serrano[1] Ismail Elezi[1] Laura Leal-Taixé[1†]

[1]Technical University of Munich [2]Munich Center for Machine Learning

Abstract

In this supplementary material, we first comment on novelty in science in Section 1 before we show results on integrating GHOST into to tracking appeoaches in Section 2. The, we give the per-class performance of GHOST on BDD100k validation set in Section 3. In Section 4 we introduce the computation of the rate of correct associations (RCA) followed by a deeper analysis of it. We then conduct a deeper analysis of our domain adaptation in Section 5. Afterwards, we show how we choose parameters based on our analysis in Section 6 and deeper investigate on the usage of different proxies in Section 6.1, the combination of appearance and motion using different weights for the sum in Section 6.2, the number of frames to be used to approximate the velocity in Section 6.3 as well as on how to utilize different thresholds in Section 6.4. Also, we conduct experiments on different values of inactive patience in Section 7. Then, we introduce how we generated the distance histograms in Section 8. Moreover, in Section 9 we first outline the difference between our approach and trackers with similar components and compare the generality of our model to the generality of ByteTrack [12] in Section 10. Finally, we comment on the latency of our approach in Section 11, and visualize several long-term occluded and low visibility bounding boxes on MOT17 public detection that GHOST successfully associates in Section 12.

1. "A painting can be beautiful even if it is simple and the technical complexity is low. So can a paper." [3].

Inspired by the blog post of Michael Black on novelty in science [3], we would like to discuss the common understanding of novelty in this paragraph. Despite often confused, incremental changes do not necessarily mean that a paper can not introduce novelty and novel ideas. "If nobody thought to change that one term, then it is ipso facto novel. The inventive insight is to realize that a small change could have a big effect and to formulate the new loss" [3]. Furthermore, it is of major importance to sometimes step back and formulate "a simple idea" since this "means stripping away the unnecessary to reveal the core of something. This is one of the most useful things that a scientist can do." [3]. If a simple idea improves the state of the art, "then it is most likely not trivial" [3]. Technical novelty is the most obvious type of novelty that reviewers look for in papers, but it is not the only one [3]. In our understanding, if the reader takes away an idea from the paper that changes the way they do research, this can be considered a positive impact of the paper. Hence, the paper is novel (it has sparked a new idea in the reader's mind). We hope readers also see it that way and we can progress with simpler, more interpretable, stronger models, and not only complex transformer-based pipelines trained on huge GPU farms.

2. Integrating GHOST within Other Trackers.

Our baseline in the main paper is the simple Hungarian tracker in introduced in Sec 3.1. Furthermore, we apply GHOST additionally to other trackers as visualized in Fig 3 of the main paper. However, to show that it can also be integrated into existing trackers, in Tab 2 we provide the performance of utilizing our reID instead of the baseline reID in Tracktor on MOT17. Since Tracktor on its own is a motion model we cannot apply our linear motion. Moreover, we provide results on utilizing our reID model and our linear motion instead of the Kalman Filter in ByteTrack on DanceTrack. Apart from the gain of using reID, the Kalman Filter struggles with the extreme motion while we can adapt the number of frames for velocity computation to the dataset.

3. Per-Class Evaluation on BDD100k Validation Set

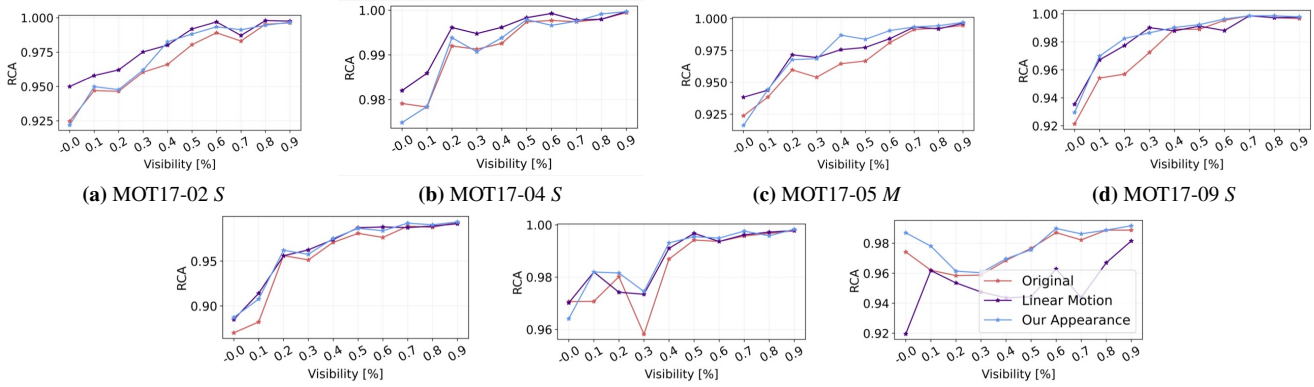
In this section, we show the class-wise performance on the BDD100k validation set (Table 1). As on the test set (see

*Correspondance to j.seidenschwarz@tum.de.

†Currently at NVIDIA.

	ByteTrack [12]			QDTrack			GHOST			
	HOTA ↑	**IDF1 ↑**	**MOTA ↑**	**HOTA ↑**	**IDF1 ↑**	**MOTA ↑**	**HOTA ↑**	**IDF1 ↑**	**MOTA ↑**	# GT det
pedestrian	48.2	58.8	56.0	46.9	59.8	49.1	48.9	59.9	54.8	56865
rider	42.9	56.3	45.1	38.0	51.7	35.2	44.7	60.6	47.0	2527
car	64.5	72.8	73.5	64.5	74.9	69.5	64.5	73.5	72.9	339521
bus	60.5	70.6	56.2	52.7	62.3	40.7	59.9	70.0	56.0	9035
truck	53.3	61.1	47.9	48.9	58.1	39.2	54.0	63.4	48.2	27280
train	0	0	0	0	0	0	0	0	-0.6	307
motorcycle	47.8	59.7	39.6	43.5	56.1	28.4	48.3	62.5	40.0	898
bycicle	46.0	57.6	43.4	38.9	49.2	28.6	45.0	55.1	41.1	4123
class average	45.4	54.6	45.2	41.7	51.5	36.3	45.7	55.6	44.9	440556
detection average	61.6	70.2	68.7	60.9	71.4	63.7	61.7	70.9	68.1	440556

Table 1. Class-Wise BDD100k Validation Set Performance.



(a) MOT17-02 *S* **(b)** MOT17-04 *S* **(c)** MOT17-05 *M* **(d)** MOT17-09 *S*

(e) MOT17-10 *M* **(f)** MOT17-11 *M* **(g)** MOT17-13 *M*

Figure 1. RCA for static *S* and moving *M* sequences with respect to visibility.

		Original			Original + GHOST		
	dataset	HOTA	IDF1	MOTA	HOTA	IDF1	MOTA
ByteTrack	DanceTrack	47.1	51.9	88.3	54.0	89.5	54.5
Tracktor	MOT17	57.7	65.9	61.8	58.7	67.5	61.8

Table 2. Applying GHOST in other Trackers.

main paper), we perform better than ByteTrack [12] and QDTrack [7] in the overall IDF1 and HOTA metrics as well as in IDF1 and HOTA of less frequent classes like rider, bus, bycicle. On the other hand, QDTrack [7] outperforms us in the overall IDF1 metric per detection box, mainly due to their higher performance for highly frequent classes like car or pedestrian. This shows that QDTrack works well only for highly frequent classes, indicating a high dependency to the train set. Note, our model is only trained on the pedestrian class, which makes our performance on other classes a good demonstration of the generality of our approach.

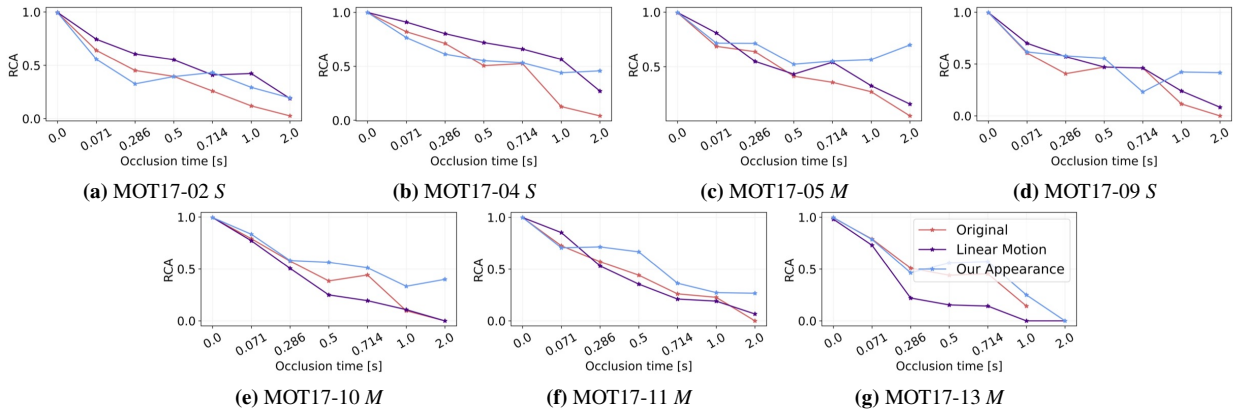
4. Detailed Analysis of Rate of Correct Associations per Sequence

We show a per sequence analysis of the rate of correct associations (RCA) of motion and appearance with respect to visibility in Fig 1, and short-term vs. long-term association in Fig 2 where *M* indicates moving sequence and *S* indicates static sequence. For this we first introduce how to compute the RCA value.

Computation of RCA. Given the output file of a tracker, to compute the rate of correct associations (RCA), we first match the given detections to ground truth identities following the same matching as the one used for the computation of the MOTA metric [4]. For each detection o_i, we then find the last previous detection that belongs to the same ground truth ID $o_{i,prev}$. If o_i was assigned to the same tracker ID as $o_{i,prev}$, we count it as a true positive association (TP-Ass), and if it was assigned to a different tracker ID, we count it as a false positive association (FP-Ass). This leads to the RCA value:

$$RCA = \frac{\text{TP-Ass}}{\text{FP-Ass} + \text{TP-Ass}}, \tag{1}$$

To get the performance for different visibility levels and occlusion time, we organize o_i into bins. For example, when we investigate the performance for visibility $0-33\%$, we take only detections o_i into account that are $0-33\%$ visible. The same holds for occlusion time: if we investigate occlusion time $0.5-0.7s$, we only take detections o_i into account whose prior detection of the same ground



(a) MOT17-02 *S* **(b)** MOT17-04 *S* **(c)** MOT17-05 *M* **(d)** MOT17-09 *S*

(e) MOT17-10 *M* **(f)** MOT17-11 *M* **(g)** MOT17-13 *M*

Figure 2. RCA for static *S* and moving *M* sequences with respect to short-term vs. long-term associations.

truth ID was $0.5 - 0.7s$ ago. This procedure allows us to investigate the performance of different trackers with respect to different influencing factors solely based on the detection output files.

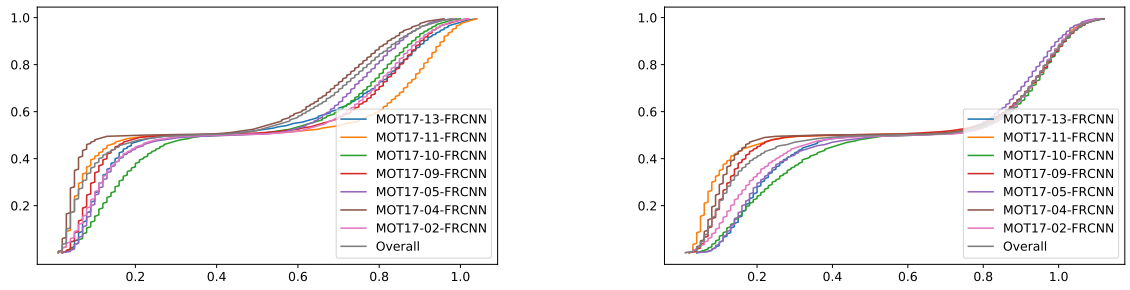
Visibility. Motion cues perform better especially in the static sequences MOT17-02 and MOT17-04. In the static sequence MOT17-09, which is recorded from a low viewpoint, and the moving sequences MOT17-05, MOT17-10, and MOT17-11, motion and appearance perform approximately on par. In MOT17-13, which shows heavy camera movements, the performance of the motion model drops significantly. Those observations show that for suitable camera angles in static sequences motion outperforms appearance independent of the visibility, while for sequences with severe camera movement or unsuitable camera angles, appearance outperforms motion even for low visibility scenarios. For moving camera sequences, the motion of the object and the camera add up, resulting in more noisy and non-linear motion observed in pixel space, even though the underlying motion might be linear. Similarly, a low viewpoint leads to a distorted observation of the underlying motion from the camera perspective. When the camera angle comes closer to a bird's eye view perspective (MOT17-04) the observed motion is less distorted.

Occlusion time. Fig 2 shows that all moving sequences show a higher RCA for appearance than motion cues. For static sequences, motion performs slightly better in MOT17-02 and MOT17-04. In the static sequence MOT17-09, the sequence recorded from a low viewpoint, both perform approximately on par. For suitable camera angles motion is a good cue even for long-term associations in static sequences, while appearance outperforms motion even for short-term associations in moving ones. This stems from the fact that motion gets more non-linear observed from camera perspective in moving camera sequences. While appearance still suffers from occlusion in static sequences even when recorded from a well-suited camera angle, those conditions allow for surprisingly well performance of motion even with respect to long-term associations.

5. A Deeper Analysis on on-the-fly Domain Adaptation

In the main paper, we visualize the distance histograms between active and inactive tracks to new detections of the same or different classes (see Fig 2 main paper). In this section, we show that the *intersection point* that divides the distance histograms between active (inactive) tracks of the same and different classes well varies less over the different sequences when using our on-the-fly domain adaptation (see Fig 2(b) in main paper) compared to when not using it (see Fig 2(a) in main paper). Furthermore, we show that the distributions are generally more *similar and stable* over different sequences with out on-ht-fly domain adaptation.

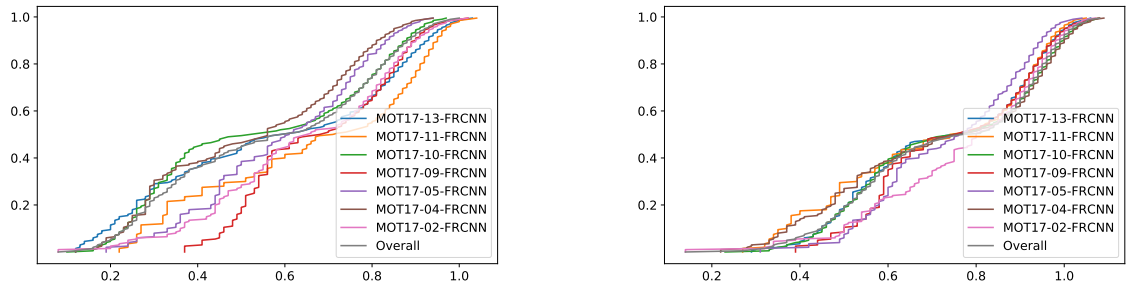
Intersection Points. Given the distribution $f_{a,d}$ of distances between active tracks (a) and new detections of a different ID (d) and the distribution $f_{a,s}$ of distances between active tracks and new detections of the same ID (s), we find a well suited intersection point ${x^a_{s,d}}^*$ separating both distributions by minimizing the sum of the costs of false positive and false negative matches. Towards this end, for a given point $x^a_{s,d}$ we define the false positive costs as percentile value of $f_{a,d}$ at $x^a_{s,d}$ given by $p^{x_{s,d}}_{a,d}$, *i.e.*, the percentile of $f_{a,d}$ that lies left to $x^a_{s,d}$. We define the false negative cost for $x^a_{s,d}$ as $100 - p^{x_{s,d}}_{a,s}$ utilizing the percentile value of $f_{a,s}$ at $x^a_{s,d}$ since we want to punish the false negatives that lie to the right of this point. Similar points $x^a_{s,d}$ across sequences allow to choose one single well-suited threshold τ_{act} over all sequences. The same holds for the inactive track distributions $f_{i,s}$ and $f_{i,d}$ and the corresponding $x^i_{s,d}$ of the different sequences. As we visualize in Fig 5, $x^i_{s,d}$ varies significantly less across tracking sequences when using our on-the-fly DA compared to when not using it.



(a) Without on-the-fly domain adaptation.

(b) With on-the-fly domain adaptation.

Figure 3. Cumulative sum of absolute bin difference between $f_{a,d}$ and $f_{a,s}$ on MOT17 validation set.



(a) Without on-the-fly domain adaptation.

(b) With on-the-fly domain adaptation.

Figure 4. Cumulative sum of absolute bin difference between $f_{a,d}$ and $f_{a,s}$ on MOT17 validation set.

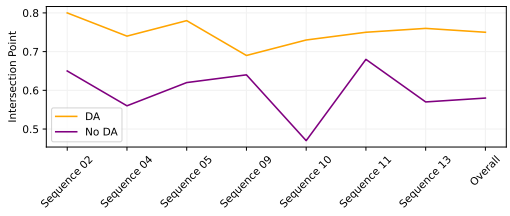


Figure 5. Visualization of the intersection points between distance histograms from detections to inactive tracks of the same and different identities when using domain adaptation (DA) compared to when not using it.

Similarity and Stability. While the variance of $x^a_{s,d}$ is higher for both settings, *i.e.*, with and without the on-the-fly DA, the distributions are generally more separated when using DA compared to when not using it. To show this, we conduct a second experiment. For each sequence, we define the same bins in the range from $0-1$ and turn the distributions $f_{a,d}$ and $f_{a,s}$ into histograms $h_{a,d}$ and $h_{a,s}$. In each bin i we compute the absolute difference between the two histograms $d_{a,i}$ and normalize it by the sum of all absolute distances. Finally, we plot the cumulative histogram (see Fig 3). The more aligned the cumulative histograms over all sequences and the broader the saddle point, the more similar the sequences across each other and the better separated the distributions of the same and different IDs, respectively. Note, that the cumulative sums of the different sequences are much more aligned when utilizing on-the-fly domain adaptation (see Fig 3b) than when not using it (see Fig 3a) which makes it easier to find a common threshold τ_{act}. Moreover, the saddle point is much broader when using on-the-fly domain adaptation which makes our approach more stable with respect to different thresholds τ_{act}.

Despite the difference visualization for the inactive track distributions being less unified over the sequences in general, the differences over the different sequences when using on-the-fly domain adaptation are more aligned compared to when not using it (see Fig 4). Combined with the less varying ${x^i_{s,d}}^*$, this leads not only to an overall better suited but also more stable threshold τ_{inact}.

6. Using the Knowledge of our Analysis.

In our work we present an in-depth analysis on appearance distance computation based on embedding features (see Fig 2 in the main paper) as well as motion vs. appearance model performance (see Fig 3-6 in the main paper). Based on those insights we introduce our simple tracker GHOST. For example, we utilize the analysis of the differences between the reID distance of active and inactive tracks to detections to adapt the thresholds and and choose a proxy distance computation method. Also, we utilize the

inisghts that reID performs worse for high occlusion levels and linear motion performs worse in moving camera scenes and with extreme motion to adapt the motion weight as well as the number of frames used in the motion model. We do not only present GHOST but also a large number of analysis that reveal insights for the community. In the following we provide a deeper analysis on the hyperparameters and design choices of GHOST for the single datasets.

6.1. The Usage of Different Proxies

We now explore different proxies for the distance computation between new detections and inactive tracks. We start from the feature vectors generated using our reID network and normalize them before further processing. As introduced in the main paper, we utilize the mean of the distances of a new detection to all detections of an inactive track. This proxy distance between new detection i and inactive track k is given by:

$$\begin{aligned}\tilde{d}(i,k) &= \frac{1}{N_k}\sum_{n=1}^{N_k} d(f_i, f_k^n) \\ &= \frac{1}{N_k}\sum_{n=1}^{N_k} 1 - \frac{f_i \cdot f_k^n}{||f_i|| \cdot ||f_k^n||} \\ &= 1 - \frac{1}{N_k}\sum_{n=1}^{N_k} f_i \cdot f_k^n\end{aligned} \tag{2}$$

where N_k is the number of detections in the inactive track and f_k^n is the feature vector corresponding to its n-th detection. We omit $||f_i|| \cdot ||f_k^n||$ as we normalize all feature vectors.

Another option is to first compute a proxy feature vector and then compute the distance between a new detection and the proxy feature vector. We investigate four proxy feature vector computations and compare them on the validation set of all four datasets.

Mean Feature Vector. The mean feature vector of all detections in the inactive track k which is also used in Tracktor [2] is given by

$$\tilde{f}_k = \frac{1}{N_k}\sum_{n=0}^{N_k} f_k^n \tag{3}$$

Computing the cosine distance of this mean feature vector leads to

$$\begin{aligned}\tilde{d}(i,k) &= 1 - \frac{f_i \cdot \frac{1}{N_k}\sum_{n=1}^{N_k} f_k^n}{||f_i|| \cdot ||\frac{1}{N_k}\sum_{n=1}^{N_k} f_k^n||} \\ &= 1 - \frac{\sum_{n=1}^{N_k} f_i \cdot f_k^n}{||\sum_{n=1}^{N_k} f_k^n||}\end{aligned} \tag{4}$$

This differs from our proxy distance by the normalization constant $\frac{1}{||\sum_{n=1}^{N_k} f_k^n||}$.

Mode Feature Vector. Compared to the mean feature vector, the feature vector of inactive track k is given by the value that appeared most in each feature dimension.

Median Feature Vector. Viewing f_k^n as a random variable, in each dimension the median feature vector contains the value for which 50% of the probability mass of feature values in this dimension lies on the right and left of it, *i.e.*, it divides the probability mass into two equal masses.

Exponential Moving Average Feature Vector. Utilizing the exponential moving average (EMA) as feature vector as done in JDE [8] or FairMOT [13] means that at given a new detection, the feature vector is updated by:

$$\tilde{f}_k^t = \tilde{f}_k^{t-1} * \alpha + f_k^t * (1 - \alpha) \tag{5}$$

where $\tilde{f}_k^{t-1}$ is the EMA feature vector at the previous time step, f_k^t is the feature vector of the new detection, and $\alpha = 0.9$ is a weighting factor. The EMA feature vectors build on the underlying assumption that feature vectors should change only slightly and, therefore, smooths the feature vector development.

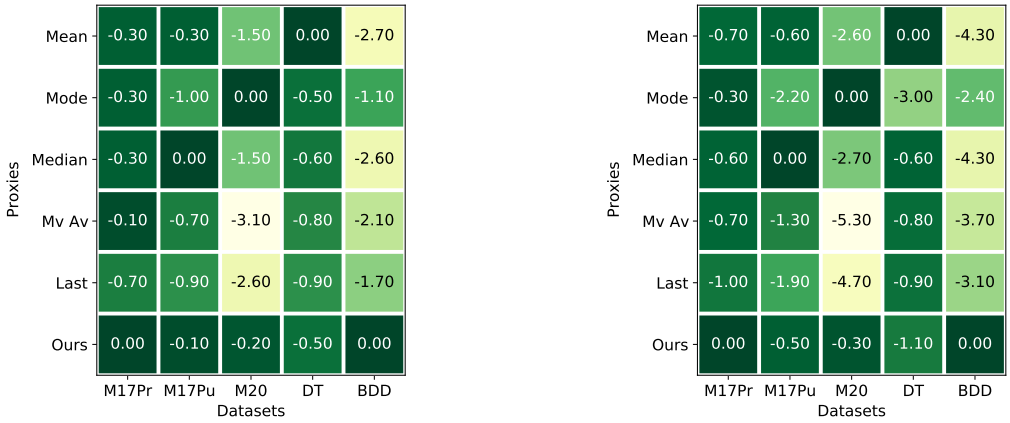
We show the performance drop on different datasets when using different proxies in Fig 6. Ours, *i.e.*, the mean distance shows the most stable performance over the different datasets and we, therefore, decided to utilize this proxy.

6.2. The Impact of Motion Weights

In this subsection, we visualize the performance drop when utilizing different motion weights on different datasets (see Fig 7). On MOT17 public detections, the best performance is achieved when using motion weight 0.4 while for private detections the best weight is 0.6. This is caused by the fact, that the appearance model gets less certain with increasing occlusion level and the private detections set contains more difficult, *i.e.*, occluded detections. On MOT20 private detections a motion weight of 0.8 for private detections performs best as the occlusion level is generally much higher than on MOT17 dataset. On DanceTrack dataset, the best motion weight is 0.4. Since the motion and articulation on this dataset are generally more diverse and extreme, the performance of the motion model is less certain compared to the appearance model. BDD dataset solely contains sequences recorded using a moving camera. As we showed in the main paper, the performance of the motion distance decreases when moving cameras are used. This is due to the fact, that the observed motion gets less linear since the motion of the camera and the object add up. Consequently, a motion weight of 0.4 works best on BDD100k MOT dataset. All those observations are in line with our analysis in the main paper as well as the more detailed analysis in this supplementary in Section 4.

	BDD	DanceTrack	MOT17	MOT20
motion	moving cam	extreme motion	partially moving cam	static cam
occlusion	medium	medium	medium	high
motion weight	0.4	0.4	0.6	0.8
# frames motion model	10	5	90	30

Table 3. Motion Model Parameters.



(a) HOTA **(b)** IDF1

Figure 6. Drop in Performance for Different Proxies on Different Datasets. M17Pr = MOT17 private detections, M17Pu = MOT17 public detections, M20 = MOT20 public detections, DT = DanceTrack, BDD = BDD100k. Mean = Mean Feature, Mode = Mode Features, Median = Median Features, Mv Av = Moving Average of Features, Last = Last Features, Ours = Our Proxy Distance.

6.3. The Impact of Different Numbers of Frames for Velocity Computation

The less linear the motion or the observed motion, the fewer frames approximate the future motion better. We visualize the impact of different numbers of frames in Fig 8. While on MOT17 private detections, the linear motion model performs well using the positions of the last 90 tracks (or less if a track contains less), on MOT20 using only the last 30 frames performs best since the scenes are highly crowded and, therefore, the motion is less linear. On DanceTrack, the motion is more extreme and, therefore, using only the last 5 frames approximates the future motion best. Similarly, on BDD100k as the observed motion is more non-linear due to the combination of the camera motion and the object motion utilizing only the last 10 frames to approximate the motion performs best. The lower frame rate of BDD sequences compared to the frame rate of MOT17, MOT20 and DanceTrack even increases this effect, since more time passes within the same number of frames on BDD. Overall, as already stated in the main paper, short-term future motion can be approximated fairly well utilizing a linear motion model. Depending on the characteristics of the motion, a different number of frames approximates the future motion best and, therefore, leads to the best tracking results.

6.4. How to use Different Thresholds τ_i

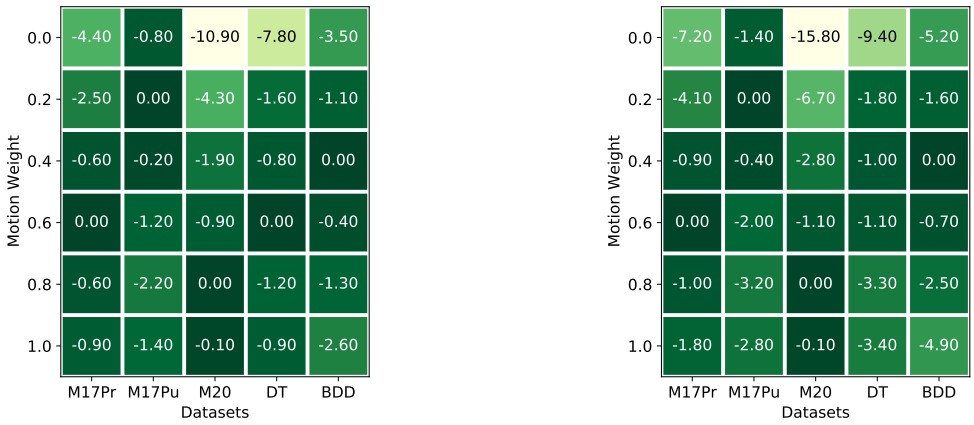
As stated in Subsection 3.2. in the main paper, we utilize different thresholds for active and inactive tracks. While commonly only one threshold is used, we empirically find that it is beneficial to allow different ones. Therefore, we apply the thresholds *after* the bipartite matching to filter the detection-trajectory pairs (i, j). We visualize our matching in Algorithm 1. n represents the number of active track, τ_{act} and τ_{inact} the threshold for active and inactive tracks and d the cost matrix.

7. Different Inactive Patience Values

Similar to other approaches [2, 5, 8, 9, 13] that only keep inactive tracks for a fixed number of frames, called inactive patience, we keep them for 50 frames for all datasets. To show that this choice is reasonable, we visualize HOTA, IDF1, and MOTA on MOT17 validation set for different inactive patience values in Fig 9. We use the same setting as in sections 4.4 and 4.5 in the main paper, *i.e.*, we use the bounding boxes of MOT17 validation set of several private trackers. The performance drops heavily for inactive patience 0 and then only slightly changes up to using all frames of a sequence after 30 frames.

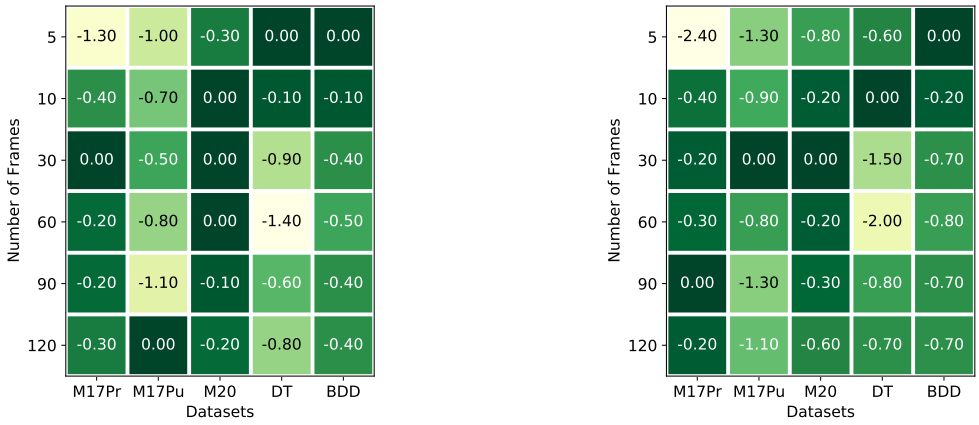
8. Computation of Distance Histograms

In the main paper, we visualize distributions of distances from active and inactive tracks to detections of the same and different classes in Fig 2. In Fig 2(a) we utilize the embeddings of the last detection of any existing track to compute the distance to the embeddings of new detections, in Fig 2(b) we utilize the distance computation



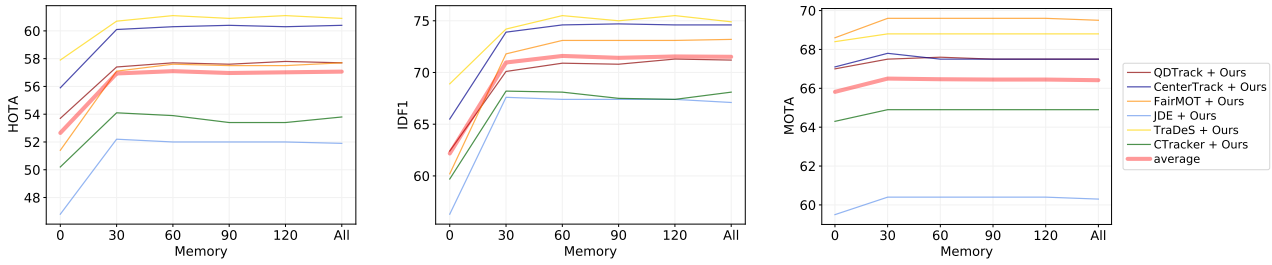
(a) HOTA **(b)** IDF1

Figure 7. Drop in Performance for Different Motion Weights on Different Datasets. M17Pr = MOT17 private detections, M17Pu = MOT17 public detections, M20 = MOT20 public detections, DT = DanceTrack, BDD = BDD100k.



(a) HOTA **(b)** IDF1

Figure 8. Drop in Performance for Different Number of Frames for Velocity Computation on Different Datasets. M17Pr = MOT17 private detections, M17Pu = MOT17 public detections, M20 = MOT20 public detections, DT = DanceTrack, BDD = BDD100k.



(a) HOTA **(b)** IDF1 **(c)** MOTA

Figure 9. Performance on MOT17 public validateion set with respect to different inactive patience values.

as introduced in Sec 3.2, and in 2(c) we visualize the motion distance. Since different distance metrics could be applied for feature vector distance and motion distance, we define both in Sec 4.1. To generate those distributions, we first match given detections to ground truth identities following the same matching as used for the computation of the MOTA metric [4]. If a detection of the same ID occurred in the last frame, we compute the distance to it and add it to the distances between active tracks and detections of the same ID. Similarly, if there was a detection present but not in the last frame, we compute the distance and add it to the distances from inactive tracks to detections of the same ID. Afterwards, we compute the distances to all other IDs that occurred in the last frame as well as

Algorithm 1: Assignment with different thresholds

Data: $n, \tau_{act}, \tau_{inact}$, cost matrix $d \in \mathcal{R}^{|T| \times |D|}$;
Result: approved rows, approved cols;
approved rows $= \emptyset$, approved cols $= \emptyset$;
matched rows, matched cols $=$ Bipartite(c);
for r, c *in zip*(*matched rows*, *matched cols*) **do**
  if $r < n$ *and* $d_{r,c} < \tau_{act}$ **then**
    approved rows $=$ approved rows $+ r$;
    approved cols $=$ approved cols $+ c$;
  else if $r >= n$ *and* $d_{r,c} < \tau_{inact}$ **then**
    approved rows $=$ approved rows $+ r$;
    approved cols $=$ approved cols $+ c$;
  else
    Discard Match.;
  end
end

to all other IDs that occurred prior to the last frame and add them to the distances from detections to active and inactive tracks of a different ID, respectively. We can add inactive patience and proxy computation methods to this basic framework. Despite we only show the distributions for MOT17 validation set, this method can be used for any dataset for which ground truth detections are available.

9. On Similar Approaches

In this section, we discuss the differences between state-of-the-art trackers that share some of their components with GHOST. ByteTrack [12] uses a Kalman Filter as motion model while we use a more simple linear motion model. More importantly, the authors treat active and inactive **tracks** the same but distinguish between high and low confidence **detections**, *i.e.*, they differentiate on a detection level while we differentiate on track level. However, as we showed in the main paper, active and inactive tracks show significant differences and treating them the same way does not leverage the full potential of the underlying cues. Also, their assignment strategy leads to a multi-level association process while GHOST only requires a single association step. Similarly, in [1] the authors treat **high and low confidence tracks** differently, *i.e.*, high confidence tracks are assigned locally to new detections and low confidence tracks are globally assigned with other tracks and detections. The inactivity of a track is only one factor of confidence. Note that this again involves multiple bipartite matchings while we assign active and inactive tracks **at the same time** which only requires one. The authors of DeepSORT [11] utilize a Kalman filter as well as an appearance model. However, the parameter that weights appearance and motion is set to $\lambda = 0$, *i.e.*, only appearance is considered. However, as we show in our analysis motion can compensate for failure cases of appearance, especially in low visibility regimes making our approach more robust. Moreover, the authors propose a cascaded matching strategy that requires more than one bipartite matching per frame while we, again, only require one. With respect to domain adaptation, HCC [6] trains on tracking sequences and uses sophisticated test-time mining to fine-tune. We rely on a simpler scheme and do not need any of the above.

Despite all of the above-mentioned approaches showing similarities to our approach, they still differ with respect to significant design choices positioning our approach as a complementary work with respect to them. Furthermore, our approach leverages the motion and appearance cues in a simple yet highly effective and general way without multi-level association procedures.

10. On the Generality of ByteTrack [12]

Recently, ByteTrack [12], which also follows the tracking-by-detection paradigm, also reported results on MOT17 and MOT20 as well as on the highly different datasets DanceTrack and BDD100k MOT. In this section, we compare GHOST to ByteTrack with respect to generality. Despite not being mentioned in the paper, the authors add tricks to the tracking procedure which are different for each dataset. **First**, they multiply their IoU cost matrix, which they obtain by using a Kalman Filter, by the detection confidence when applying their tracker to MOT17, but not when applying it to other datasets. **Second**, the authors apply interpolation on MOT17 and MOT20 datasets which turns their approach into an offline approach. **Third**, on DanceTrack and BDD100k, they allow all bounding boxes to be used, while they filter out bounding boxes if $\frac{w}{h} > 1.6$ on MOT17 and MOT20, where w and h are bounding box width and height, respectively. **Fourth**, ByteTrack uses a reID model, namely UniTrack [10], on BDD100k dataset whilst they do not use any reID model on the other datasets. **Fifth**, they adapt the tracking thresholds *per sequence* on MOT17 and MOT20 during training *and testing*. On BDD and DanceTrack the tracking thresholds are applied per dataset. **Sixth**, as commonly done the authors adapt other model parameters, *e.g.*, the matching thresholds, confidence threshold for detections as well as the confidence threshold for new tracks.

We believe these are small but significant changes that put into question the generality of ByteTrack. In contrast, we keep our tracking pipeline the same over different datasets but solely change our model parameters for each dataset as a **whole**. To be specific, we adapt the thresholds τ_i, the detection confidence thresholds to filter plain detections and start new tracks, the motion weights, as well as the number of frames used in the linear motion model. This makes our approach more general and easier to

apply to new datasets.

11. Latency

For a fair comparison with other methods, we evaluated GHOST, Tracktor [2], and FairMOT [13] on the public detections on the MOT17 validation set utilizing the same GPU, namely a Quadra P6000. Note that we utilize CenterTrack pre-processed detections here. With 10FPS GHOST is on the same magnitude of speed as current SOTA trackers. While Tracktor [2] runs at 2FPS, FairMOT [13] runs at 17FPS as it was optimized for real-time. When evaluating the private detection setting, our method's latency increases slightly due to the increase of bounding boxes to process to 6FPS. The average latency per frame and per model part is given by 10ms for the computation of the reID features, 30ms for the reID distance computation, 0.1 ms for updating the velocity per track, 0.0025ms for the motion step per track, 0.4ms for the motion distance computation, 0.35ms for the biparite matching, and 0.1ms for updating all tracks.

12. Visualizations

In this section, we visualize associations on CenterTrack re-fined public bounding boxes that our model is able to correctly associate while CenterTrack is not. Correct and wrong associations are determined in the same way as done for the computation of the RCA Section 4. This means we determine wrong associations by first matching all detection bounding boxes to the ground truth IDs. A wrong association is given if the prior detection of the same ground truth ID as a current detection was assigned to a different tracker ID than the current detection. In Figures 10-17, we visualize the prior detection on the left side and the current detection on the right side. All examples were associated wrongly by CenterTrack and correctly by GHOST. We give the time distance between the prior and the current frame in the caption as well as the visibility level of the re-appearing pedestrian. By our combination of appearance and motion, we are able to correctly associate pedestrians after long occlusions and low visibility in highly varying sequences.

References

[1] Seung Hwan Bae and Kuk-Jin Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *CVPR*, 2014. 8

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *ICCV*, 2019. 5, 6, 9

[3] Michael Black. Novelty in science. https://tinyurl.com/25rch3cb. 1

[4] Rangachar Kasturi, Dmitry B. Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John S. Garofolo, Rachel Bowers, Matthew Boonstra, Valentina N. Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *tPAMI*, 2009. 2, 7

[5] Qiankun Liu, Qi Chu, Bin Liu, and Nenghai Yu. GSM: graph similarity model for multi-object tracking. In *IJCAI*, 2020. 6

[6] Liqian Ma, Siyu Tang, Michael J. Black, and Luc Van Gool. Customized multi-person tracker. In *ACCV*, 2018. 8

[7] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *CVPR*, 2021. 2

[8] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 5, 6

[9] Daniel Stadler and Jürgen Beyerer. Improving multiple pedestrian tracking by track management and occlusion handling. In *CVPR*, 2021. 6

[10] Zhongdao Wang, Hengshuang Zhao, Ya-Li Li, Shengjin Wang, Philip H. S. Torr, and Luca Bertinetto. Do different tracking tasks require different appearance models? In *NeurIPS*, 2021. 8

[11] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 8

[12] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *ECCV*, 2022. 1, 2, 8

[13] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *IJCV*, 129(11):3069–3087, 2021. 5, 6, 9

Figure 10. Occlusion time: $2s$, visibility of re-appearing pedestrian: 0.6.

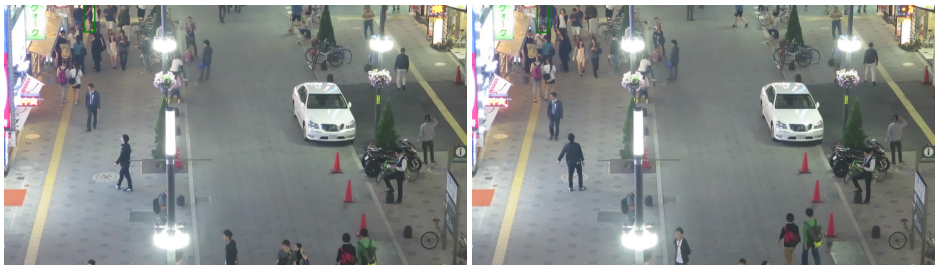


Figure 11. Occlusion time: 1.1s, visibility of re-appearing pedestrian: 0.3.

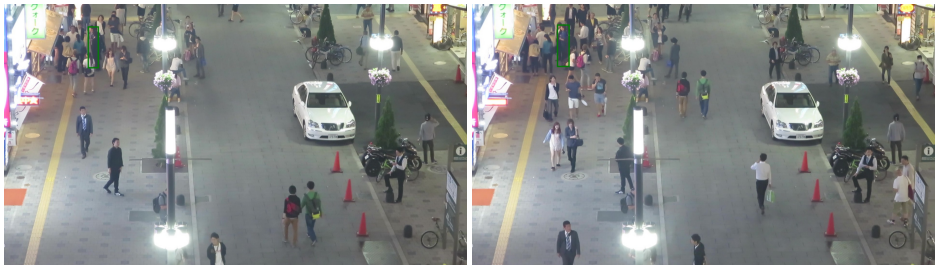


Figure 12. Occlusion time: 9.4s, visibility of re-appearing pedestrian: 0.4.

Figure 13. Occlusion time: 1.1s, visibility of re-appearing pedestrian: 0.5.

Figure 14. Occlusion time: 1.1s, visibility of re-appearing pedestrian: 0.2.

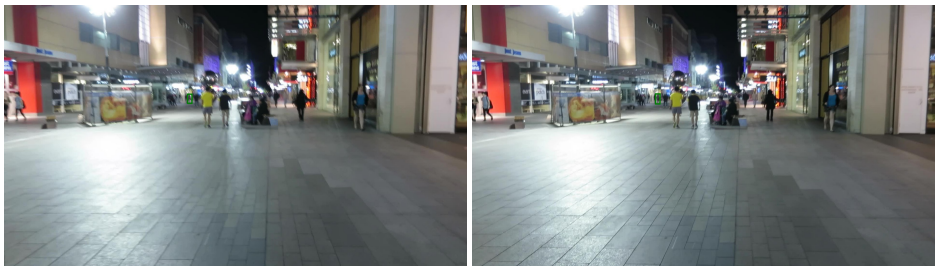


Figure 15. Occlusion time: 0.1s, visibility of re-appearing pedestrian: 0.6.

Figure 16. Occlusion time: 0.9s, visibility of re-appearing pedestrian: 0.5.

Figure 17. Occlusion time: 0.2s, visibility of re-appearing pedestrian: 0.8.