Appendices

In this supplementary material, we first provide details about training in Sec. A and about inference in Sec. B. We then report a more detailed analysis of our method in Sec. C and provide more detailed results of the experiments shown in the main paper in Sec. D.

Furthermore, we want to draw the attention of the reader to the compiled video available on the project page at https://github.com/visionml/pytracking. The video provides more visual insights about our tracker and compares visually with the baseline tracker SuperDiMP. It shows tracking of distractors and indicates the same object ids in consecutive frames with agreeing color.

A. Training

First, we describe the training data generation and sample selection to train the network more effectively. Then, we provide additional details about the training procedure such as training in batches, augmentations and synthetic sample generation. Finally, we briefly summarize the employed network architecture.

A.1. Data-Mining

We use the LaSOT [15] training set to train our target candidate association network. In particular, we split the 1120 training sequences randomly into a train-train (1000 sequences) and a train-val (120 sequences) set. We run the base tracker on all sequences and store the target classifier score map and the search area on disk for each frame. During training, we use the score map and the search area to extract the target candidates and its features to provide the data to train the target candidate association network.

We observed that many sequences or sub-sequences contain mostly one target candidate with a high target classifier score. Thus, in this cases target candidate association is trivial and learning on these cases will be less effective. Conversely, tracking datasets contain sub-sequences that are very challenging (large motion or appearance changes or many distractors) such that trackers often fail. While these sub-sequences lead to a more effective training they are relatively rare such that we decide to actively search the training dataset.

First, we assign each frame to one of six different categories. We classify each frame based on four observations about the number of candidates, their target classifier score, if one of the target candidates is selected as target and if this selection is correct, see Tab. 1. A candidate corresponds to the annotated target object if the spatial distance between the candidate location and center coordinate of the target object is smaller than a threshold.

Assigning each frame to the proposed categories, we observe, that the dominant category is D (70%) that corresponds to frames with a single target candidate matching the annotated target object. However, we favour more challenging settings for training. In order to learn distractor associations using self supervision, we require frames with multiple detected target candidates. Category H (18.4%) corresponds to such frames where in addition the candidate with the highest target classifier score matches the annotated target object. Hence, the base tracker selects the correct candidate as target. Furthermore, category G corresponds to frames where the base tracker was no longer able to track the target because the target classifier score of the corresponding candidate fell below a threshold. We favour these frames during training in order to learn continue tracking the target even if the score is low.

Both categories J and K correspond to tracking failures of the base tracker. Whereas in K the correct target is de-

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of candidates</th>
<th>Is a candidate selected as target?</th>
<th>Does the candidate with max score correspond to the target?</th>
<th>Does any candidate correspond to the target?</th>
<th>Num Frames</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1.8M</td>
<td>67.9%</td>
</tr>
<tr>
<td>H</td>
<td>&gt; 1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>498k</td>
<td>18.4%</td>
</tr>
<tr>
<td>G</td>
<td>&gt; 1</td>
<td>x</td>
<td>✓</td>
<td>-</td>
<td>8k</td>
<td>0.3%</td>
</tr>
<tr>
<td>J</td>
<td>&gt; 1</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>76k</td>
<td>2.8%</td>
</tr>
<tr>
<td>K</td>
<td>&gt; 1</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>42k</td>
<td>1.5%</td>
</tr>
<tr>
<td>other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>243k</td>
<td>9.1%</td>
</tr>
</tbody>
</table>

Table 1. Categories and specifications for each frame in the training dataset used for data-mining.
ected but not selected, it is not detected in frames of category J. Thus, we aim to learn from tracking failures in order to train the target candidate association network such that it learns to compensate for tracking failures of the base tracker and corrects them. In particular, frames of category K are important for training because the two candidates with highest target classifier score no longer match such that the network is forced to include other cues for matching. We use frames of category J because frames where the object is visible but not detected contain typically many distractors such that these frames are suitable to learn distractor associations using self-supervised learning.

To summarize, we select only frames with category H, K, J for self-supervised training and sample them with a ratio of $2:1:1$ instead of $10:2:1$ (ratio in the dataset). We ignore frames from category D during self-supervised training because we require frames with multiple target candidates. Furthermore, we select sub-sequences of two consecutive frames for partially supervised training. We choose challenging sub-sequences that either contain many distractors in each frame (HH, 350k) or sub-sequences where the base tracker fails and switches to track a distractor (HK, 1001) or where the base tracker is no longer able to identify the target with high confidence (HG, 1380). Again we change the sampling ratio from approximately $350:1:1$ to $10:1:1$ during training. We change the sampling ratio in order to use failure cases more often during training than they occur in the training set.

A.2. Training Data Preparation

During training we use two different levels of augmentation. First, we augment all features of target candidate to enable self-supervised training with automatically produced ground truth correspondences. In addition, we use augmentation to improve generalization and overfitting of the network.

When creating artificial features we randomly scale each target classifier score, randomly jitter the candidate location within the search area and apply common image transformations to the original image before extracting the appearance based features for the artificial candidates. In particular, we randomly jitter the brightness, blur the image and jitter the search area before cropping the image to the search area.

To reduce overfitting and improve the generalization, we randomly scale the target candidate scores for synthetic and real sub-sequences. Furthermore, we remove candidates from the sets $V'$ and $V$ randomly in order to simulate newly appearing or disappearing objects. Furthermore, to enable training in batches we require the same number of target candidates in each frame. Thus, we keep the five candidates with the highest target classifier score or add artificial peaks at random locations with a small score such that five candidates per frame are present. When computing the losses, we ignore these artificial candidates.

A.3. Architecture Details

We use the SuperDiMP tracker [8] as our base tracker. SuperDiMP employs the DiMP [1] target classifier and the probabilistic bounding-box regression of PrDiMP [12], together with improved training settings. It uses a ResNet-50 [22] pretrained network as backbone feature extractor. We freeze all parameters of SuperDiMP while training the target candidate association network. To produce the visual features for each target candidate, we use the third layer ResNet-50 features. In particular, we obtain a $29 \times 29 \times 1024$ feature map and feed it into a $2 \times 2$ convolutional layer which produces the $30 \times 30 \times 256$ feature map $f$. Note, that the spatial resolution of the target classifier score and feature map agree such that extracting the appearance based features $f_i$ for each target candidate $v_i$ at location $c_i$ is simplified.

Furthermore, we use a four layer Multi-Layer Perceptron (MLP) to project the target classifier score and location for each candidate in the same dimensional space as $f_i$. We use the following MLP structure: $3 \to 32 \to 64 \to 128 \to 256$ with batch normalization. Before feeding the candidate locations into the MLP we normalize it according to the image size.

We follow Sarlin et al. [34] when designing the candidate embedding network. In particular, we use self and cross attention layers in an alternating fashion and employ two layers of each type. In addition, we append a $1 \times 1$ convolutional layer to the last cross attention layer. Again, we follow Sarlin et al. [34] for optimal matching and reuse their implementation of the Sinkhorn algorithm and run it for 10 iterations.

B. Inference

In this section we provide the detailed algorithm that describes the object association module (Sec. 3.7 in the paper). Furthermore, we explain the idea of search area rescaling at occlusion and how it is implemented. We conclude with additional inference details.

B.1. Object Association Module

Here, we provide a detailed algorithm describing the object association module presented in the main paper, see Alg. 1. It contains target candidate to object association and the redetection logic to retrieve the target object after it was lost.

First, we will briefly explain the used notation. Each object can be modeled similar to a class in programming. Thus, each object $o$ contains attributes that can be accessed using the "$\cdot$" notation. In particular $(o_j)_s$ returns the score
object class contains two attributes: the list of scores \( s \) and the object-id \( id \). Both setting and getting the attribute values is possible.

The algorithm requires the following inputs: the set of target candidates \( V \), the set of detected objects \( O' \) and the object selected as target \( \hat{o} \) in the previous frame. First, we check if a target candidate matches with any previously detected object and verify that the assignment probability is higher than a threshold \( \omega = 0.75 \). If such a match exists, we associate the candidate to the object and append its target classifier score to the scores and add the object to the set of currently visible object \( O \). If a target candidate matches none of the previously detected objects, we create a new object and add it to \( O \). Hence, previously detected objects that miss a matching candidate are not included in \( O \). Once, all target candidates are associated to an already existing or newly created object. We check if the object previously selected as target is still visible in the current scene and forms the new target \( \hat{o} \). After the object was lost it is possible that the object selected as target is in fact a distractor. Thus, we select an other object as target if this other object achieves a higher target classifier score in the current frame than any score the currently selected object achieved in the past. Furthermore, if the object previously selected as target object is no longer visible, we try to redetect it by checking if the object with highest target classifier score in the current frame achieves a score higher than a threshold \( \eta = 0.25 \). If the score is high enough, we select this object as the target.

\[ \text{Algorithm 1: Object Association Algorithm.} \]

**Require:** Set of target candidates \( V \)

**Require:** Set of objects of previous frame \( O' \)

**Require:** Target object \( \hat{o} \)

1. \( O = \{ \}, N = |V| \)
2. for \( i = 1, \ldots, N \) do
3. if matchOf(\( v_i \)) \( \neq \) dustbin && \( p(v_i) \geq \omega \) then
4. \( v'_i \leftarrow \text{matchOf}(v_i) \)
5. \( (o'_i).s \leftarrow \text{concat}((o'_i).s, [s_i]) \)
6. \( o_i \leftarrow o'_i \)
7. else
8. \( o_i \leftarrow \text{newObject(getNewId(), [s_i])} \)
9. \( O \leftarrow O \cup \{ o_i \} \)
10. if \( \hat{o}' \neq \) none and \( \hat{o}'.id \in \{ o.id \mid o \in O \} \) then
11. \( \hat{o} = \text{getObjectById}(O, \hat{o}'.id) \)
12. for \( i = 1, \ldots, N \) do
13. if \( \max(\hat{o}.s) < (o_i).s[-1] \) then
14. \( \hat{o} = o_i \)
15. else
16. \( i = \text{argmax}_i \left\{ (o_i).s[-1] \mid o_i \in O \right\} \)
17. if \( (o_i).s[-1] \geq \eta \) then
18. \( \hat{o} = o_i \)
19. else
20. \( \hat{o} = \text{none} \)
21. return \( \hat{o}, O \)

B.2. Search Area Rescaling at Occlusion

The target object often gets occluded or moves out-of-view in many tracking sequences. Shortly before the target is lost the tracker typically detects only a small part of the target object and estimates a smaller bounding box than in the frames before. The used base tracker SuperDiMP employs a search area that depends on the currently estimated bounding box size. Thus, a partially visible target object causes a small bounding box and search area. The problem of a small search area is that it complicates redetecting the target object, e.g., the target reappears at a slightly different location than it disappeared and if the object then reappears outside of the search area redetection is prohibited. Smaller search areas occur more frequently when using the target candidate association network because it allows to track the object longer until we declare it as lost.

Hence, we use a procedure to increase the search area if it decreased before the target object was lost. First, we store all search areas during tracking in an list \( a \) as long as the object is detected. If the object was lost \( k \) frames ago, we compute the new search area by averaging the last \( k \) entries of \( a \) larger than the search area at occlusion. We average at most over 30 previous search areas to compute the new one. If the target object was not redetected within these 30 frames with keep the search area fixed until redetection.

B.3. Inference Details

In contrast to training, we use all extracted target candidates to compute the candidate associations between consecutive frames. In order to save computations, we extract the candidates and features only for the current frame and cache the results such that they can be reused when computing the associations in the next frames.

B.3.1 KeepTrack Settings

We use the same settings as for SuperDiMP but increase the search area scale from 6 to 8 leading to a larger search area (from \( 352 \times 352 \) to \( 480 \times 480 \)) and to a larger target score map (from \( 22 \times 22 \) to \( 30 \times 30 \)). In addition, we employ the aforementioned search area rescaling at occlusion and skip running the target candidate association network if only one target candidates with high target classifier score is detected in the current and previous frame, in order to save computations.
### B.3.2 KeepTrackFast Settings

We use the same settings as for SuperDiMP. In particular, we keep the search area scale and target score map resolution the same to achieve a faster run-time. In addition, we reduce the number of bounding box refinement steps from 10 to 3 which reduces the bounding box regression time significantly. Moreover, we double the target candidate extraction threshold $\tau$ to 0.1. This step ensures that we neglect local maxima with low target classifier scores and thus leads to less frames with multiple detected candidates. Hence, KeepTrackFast runs the target candidate association network less often than KeepTrack.

### C. More Detailed Analysis

In addition to the ablation study presented in the main paper (Sec. 4.1) we provide more settings in order to assess the contribution of each component better. In particular, we split the term search area adaptation into larger search area and search area rescaling. Where larger search area refers to a search area scale of 8 instead of 6 and a search area resolution of 480 instead of 352 in the image domain. Tab. 2 shows all the results on NFS [18], UAV123 [32] and LaSOT [15]. We run each experiment five times and report the average. We conclude that both search area adaptation techniques improve the tracking quality but we achieve the best results on all three datasets when employing both at the same time. Furthermore, we evaluate the target candidate association network with different numbers of Sinkhorn iterations and with different number of GNN layers of the embedding network or dropping it at all, see Tab. 3. We conclude, that using the target candidate association network even without any GNN layers outperforms the baseline on all three datasets. In addition, using either two or nine GNN layers improves the performance even further on all datasets. We achieve the best results when using nine GNN layers and 50 Sinkhorn iterations. However, using a large candidate embedding network and a high number of Sinkhorn iterations reduces the run-time of the tracker to 12.7 FPS. Hence, using only two GNN layers and 10 Sinkhorn iterations results in a negligible decrease of 0.1 on UAV123 and LaSOT but accelerates the run-time by 44%.

### D. Experiments

We provide more details to complement the state-of-the-art comparison performed in the paper. And provide results for the VOT2018LT [24] challenge.

#### D.1. LaSOT and LaSOTExtSub

In addition to the success plot, we provide the normalized precision plot on the LaSOT [15] test set (280 videos) and LaSOTExtSub [15] test set (150 videos). The normalized precision score $\text{NPr}_D$ is computed as the percentage of frames where the normalized distance (relative to the target size) between the predicted and ground-truth target center location is less than a threshold $D$. $\text{NPr}_D$ is plotted over a range of thresholds $D \in [0, 0.5]$. The trackers are ranked using the AUC, which is shown in the legend. Figs. 1b and 2b show the normalized precision plots. We compare with state-of-the-art trackers and report their success (AUC) in Tab. 4 and where available we show the raw results in Fig. 1. In particular, we use the raw results provided by the authors except for DaSiamRPN [49], GlobaTrack [23], SiamRPN++ [27] and SiamMask [38] such results were not provided such that we use the raw results produced by Fan et al. [15]. Thus, the exact results for these methods might be different in the plot and the table, because we show in the table the reported result the corresponding paper. Similarly, we obtain all results on LaSOTExtSub directly from Fan et al. [14] except the result of SuperDiMP that we produced.

#### D.2. UAV123, OTB-100 and NFS

We provide the success plot over the 123 videos of the UAV123 dataset [32] in Fig. 3a, the 100 videos of the OTB-100 dataset [39] in Fig. 3b and the 100 videos of the NFS dataset [18] in Fig. 3c. We compare with state-of-the-art trackers SuperDiMP [8], PrDiMP50 [12], UPDT [3], SiamRPN++ [27], ECO [10], DiMP [1], CCOT [11], MDNet [33], ATOM [9], and DaSiamRPN [49]. Our method provides a significant gain over the baseline SuperDiMP on UAV123 and NFS and performs among the top methods on OTB-100. Tab. 5 shows additional results on UAV123, OTB-100 and NFS in terms of success (AUC).
Figure 1. Success and normalized precision plots on LaSOT [15]. Our approach outperforms all other methods by a large margin in AUC, reported in the legend.

<table>
<thead>
<tr>
<th>Method</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeepTrack</td>
<td>67.1</td>
</tr>
<tr>
<td>KeepTrackFast</td>
<td>66.8</td>
</tr>
<tr>
<td>AlphaRefine</td>
<td>65.9</td>
</tr>
<tr>
<td>TransT</td>
<td>64.9</td>
</tr>
<tr>
<td>Siam R-CNN</td>
<td>64.8</td>
</tr>
<tr>
<td>TiDiMP</td>
<td>63.9</td>
</tr>
<tr>
<td>SuperDiMP</td>
<td>63.1</td>
</tr>
<tr>
<td>STMTtrack</td>
<td>60.8</td>
</tr>
<tr>
<td>PODiMP50</td>
<td>59.8</td>
</tr>
<tr>
<td>DiMPtrack</td>
<td>58.4</td>
</tr>
<tr>
<td>TACT</td>
<td>57.5</td>
</tr>
<tr>
<td>LTMU</td>
<td>57.2</td>
</tr>
<tr>
<td>DiMP</td>
<td>56.9</td>
</tr>
<tr>
<td>Ocean</td>
<td>56.0</td>
</tr>
<tr>
<td>GlobalTrack</td>
<td>56.0</td>
</tr>
<tr>
<td>DaSiamRPN</td>
<td>55.5</td>
</tr>
</tbody>
</table>

Table 4. Comparison with state-of-the-art on the LaSOT [15] test set in terms of overall AUC score. The average value over 5 runs is reported for our approach. The symbol † marks results that were produced by Fan et al. [15] otherwise they are obtained directly from the official paper.

Figure 2. Success and normalized precision plots on LaSOTExtSub [14]. Our approach outperforms all other methods by a large margin in AUC, reported in the legend.
Overlap Precision [%]

E. Speed Analysis

Our method adds an overhead of 19.3ms compared to the baseline tracker. Whereas target candidate extraction is required for every frame, running the target candidate association network is only required if more than one candidate is detected. Candidate extraction is relatively fast and takes only 2.7 ms while performing candidate association requires 16.6 ms. Where computing the candidate embedding takes 10.0 ms, running the Sinkhorn algorithm for 10 iterations takes 3.5 ms and object association 3.1 ms. Thus, we achieve an average run-time of 18.3 FPS for KeepTrack and 29.6 FPS for KeepTrackFast when using SuperDiMP as base tracker. We report this timings on a single Nvidia GTX 2080Ti GPU.

E.1. Number of Candidates and Time Complexity

In practice, we found the number of target candidates to vary from 0 up to 15 in exceptional cases. For frames with less than 2 candidates, we naturally do not need to apply our association module. Further, we observed no measurable increase in run-time from 2 to 15 candidates, due to the effective parallelization. Therefore, we do not explicitly limit the number of detected candidates.
F. Failure Cases

While KeepTrack is particularly powerful when distractor objects appear in the scene, it also fails to track the target object in complex scenes, such as the examples shown in Fig. 4.

The top row shows such a challenging case, where the target object is the right hand of the person on the right (indicated by [■] or [■]). KeepTrack manages to individually track all hands (●,●,●) visible in the search area until frame number 134. In the next frame, both hands of the person are close and our tracker only detects one candidate for both hands (●). Thus, the tracker assigns the target id to the remaining candidate. The tracker detects two candidates (●,●) as soon as both hands move apart in frame 143. However, now it is unclear which hand is the sought target. If two objects approach each other it is unclear whether they cross each other or not. In this scenario positional information is of limited use. Hence, deeper understanding of the scene and the target object seems necessary to mitigate such failure cases.

The bottom row in Fig. 4 shows a similar failure case where again a distractor object (●) crosses the target’s (●) location. This time, the tracker fails to extract the candidate corresponding to the target from frame number 77 onwards. The tracker detects that the target candidate previously assumed to represent the target has vanished but the remaining distractor object (●) achieves such a high target score that the tracker reconsiders its previous target selection and continues tracking the distractor object (●) instead. Thus, the tracker continues tracking the distractor object even if the a target candidate for the sought target (●) appears (frame number 93).

G. Attributes

Tabs. 7 and 8 show the results of various trackers including KeepTrack and KeepTrackFast based on different sequence attributes. We observe that both trackers are superior to other trackers on UAV123 for most attributes. In particular, we outperform the runner-up by a large margin in terms of AUC on the sequences corresponding to the following attributes: Aspect Ratio Change (+2.5/2.6%), Full Occlusion (+1.8/1.9%), Partial Occlusion (+2.5/2.3%), Background Clutter (+1.5/1.3%), Illumination Variation (+1.7/1.5%), Similar Object (+1.8/1.1%). Especially, the superior performance on sequences with the attributes Full Occlusion, Partial Occlusion, Background Clutter and Similar Object clearly demonstrates that KeepTrack mitigates the harmful effect of distractors and allows to track the target object longer and more frequently than other trackers. Fig. 3a shows a similar picture: KeepTrack is the most robust tracker but others achieve a higher bounding box regression accuracy. In addition, Tab. 7 reveals that KeepTrack achieves the highest (red) or second-highest (blue) AUC on the sequences corresponding to each attribute except for the attribute Out-of-View.

The attribute-based analysis on LaSOT allows similar observations. In particular, KeepTrack and KeepTrackFast outperform all other trackers by a large margin in AUC on the sequences corresponding oth the following attributes: Partial Occlusion (+1.8/1.5%), Background Clutter (+2.3/1.2), Viewpoint Change (+1.6/2.3%), Full Occlusion (+2.7/1.8%), Fast Motion (4.1/3.5%), Out-of-View (+1.9/1.2%), Low Resolution (+3.4/3.4%). Moreover, the superior performance is even clearer when comparing to the base tracker SuperDiMP, e.g., and improvement of +6.0/5.2% for Full Occlusion or +7.0/6.4% for Fast Motion. KeepTrack achieves the highest AUC score for every attributed except two where KeepTrackFast achieves slightly higher scores. Again, the best performance on sequences with attributes such as Background Clutter or Full Occlusion clearly demonstrates the effectiveness of our proposed target and distractor association strategy.
Table 7. UA V123 attribute-based analysis in terms of AUC score. Each column corresponds to the results computed on all sequences in the dataset with the corresponding attribute.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Illumination Partial Occlusion</th>
<th>Deformation</th>
<th>Motion Blur</th>
<th>Camera Motion</th>
<th>Rotation</th>
<th>Background Clutter</th>
<th>Viewpoint Change</th>
<th>Scale Variation</th>
<th>Full Occlusion</th>
<th>Fast Occlusion</th>
<th>Out-of-View</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTMU</td>
<td>56.5</td>
<td>54.0</td>
<td>57.2</td>
<td>55.8</td>
<td>61.6</td>
<td>55.1</td>
<td>49.9</td>
<td>56.7</td>
<td>57.1</td>
<td>49.9</td>
<td>57.1</td>
<td>51.4</td>
</tr>
<tr>
<td>PrDiMP0</td>
<td>63.7</td>
<td>56.9</td>
<td>60.8</td>
<td>57.9</td>
<td>64.2</td>
<td>58.1</td>
<td>54.3</td>
<td>59.2</td>
<td>59.4</td>
<td>51.3</td>
<td>51.3</td>
<td>53.5</td>
</tr>
<tr>
<td>STMTrack</td>
<td>65.2</td>
<td>57.1</td>
<td>64.0</td>
<td>55.3</td>
<td>63.3</td>
<td>60.1</td>
<td>54.1</td>
<td>58.2</td>
<td>60.6</td>
<td>47.8</td>
<td>42.4</td>
<td>51.9</td>
</tr>
<tr>
<td>SuperDiMP</td>
<td>67.8</td>
<td>59.7</td>
<td>63.4</td>
<td>62.0</td>
<td>68.0</td>
<td>61.4</td>
<td>57.3</td>
<td>63.4</td>
<td>62.9</td>
<td>54.1</td>
<td>50.7</td>
<td>59.0</td>
</tr>
<tr>
<td>TrDiMP</td>
<td>67.5</td>
<td>61.1</td>
<td>64.4</td>
<td>62.4</td>
<td>68.1</td>
<td>62.4</td>
<td>58.9</td>
<td>62.8</td>
<td>63.4</td>
<td>56.4</td>
<td>53.0</td>
<td>60.7</td>
</tr>
<tr>
<td>Siam R-CNN</td>
<td>64.6</td>
<td>62.2</td>
<td>65.2</td>
<td>63.1</td>
<td>68.2</td>
<td>64.1</td>
<td>54.2</td>
<td>65.3</td>
<td>64.5</td>
<td>55.3</td>
<td>51.5</td>
<td>62.2</td>
</tr>
<tr>
<td>TransT</td>
<td>65.2</td>
<td>62.0</td>
<td>67.8</td>
<td>63.0</td>
<td>67.2</td>
<td>64.3</td>
<td>57.9</td>
<td>61.7</td>
<td>64.6</td>
<td>55.3</td>
<td>51.0</td>
<td>58.2</td>
</tr>
<tr>
<td>AlphaRefine</td>
<td>69.4</td>
<td>62.3</td>
<td>66.3</td>
<td>65.2</td>
<td>68.0</td>
<td>63.0</td>
<td>58.8</td>
<td>63.1</td>
<td>65.1</td>
<td>57.4</td>
<td>53.6</td>
<td>61.1</td>
</tr>
<tr>
<td>KeepTrackFast</td>
<td>70.1</td>
<td>63.8</td>
<td>66.2</td>
<td>65.0</td>
<td>70.7</td>
<td>65.1</td>
<td>60.1</td>
<td>66.7</td>
<td>66.6</td>
<td>59.2</td>
<td>57.1</td>
<td>63.4</td>
</tr>
<tr>
<td>KeepTrack</td>
<td>69.7</td>
<td>64.1</td>
<td>67.8</td>
<td>66.7</td>
<td>71.0</td>
<td>65.3</td>
<td>61.2</td>
<td>66.9</td>
<td>66.8</td>
<td>60.1</td>
<td>57.7</td>
<td>64.1</td>
</tr>
</tbody>
</table>

Table 8. LaSOT attribute-based analysis. Each column corresponds to the results computed on all sequences in the dataset with the corresponding attribute.

References


