Learning a Proposal Classifier for Multiple Target Tracking (Supplementary Material)

Peng Dai[1] Renliang Weng[2] Wongun Choi[2] Changshui Zhang[1] Zhangping He[2] Wei Ding[2]

[1]Tsinghua University, Beijng, China. [2]Aibee Inc

[1]{daip2020, zcs}@mail.tsinghua.edu.cn [2]{rlweng, wgchoi, zphe, weiding}@aibee.com

A. Detailed Algorithm

In this section, we first detail the gating strategy in affinity graph construction, and then provide the pseudocode of the algorithms presented in the main paper.

A.1. Gating Strategy

To reduce the complexity of the graph, we adopt a simple gating strategy to remove the edges exceeding the thresholds. Specifically, let $\mathcal{O}_i$ represent the valid neighbors of vertex $\mathbf{v}_i$, and $\mathcal{O}_i$ is obtained by:

$$\mathcal{O}_i = \{\forall v_j;\ \mathcal{I}^t(\mathbf{t}_i, \mathbf{t}_j, \tau_t) \& \mathcal{I}^p(\mathbf{p}_i, \mathbf{p}_j, \tau_p) \& \mathcal{I}^a(\mathbf{a}_i, \mathbf{a}_j, \tau_a)\} \tag{1}$$

where $\mathcal{I}^t$ is an indicator function to check if the minimum time gap between vertex $\mathbf{v}_i$ and $\mathbf{v}_j$ is less than τ_t, $\mathcal{I}^p$ is also an indicator function to check if the location distance is less than τ_p when having the minimum time gap, and $\mathcal{I}^a$ checks if the appearance distance is less than τ_a. The thresholds τ_t, τ_p and τ_a determine the radius of the gate.

A.2. Proposal Generation and Deoverlapping

Algorithm 1 and Algorithm 2 show the detailed procedures to generate proposals. In these algorithms, s_{max} (maximum cluster size) and Δ (cluster threshold step) are utilized to improve the purity of the generated clusters in the early iterations. It should be noted that we adopt a compatible function to keep all pairwise vertices within a cluster to be temporally compatible, i.e., no temporally overlapping vertices are allowed within the same cluster.

Algorithm 3 provides a summary of the de-overlapping procedures to generate the final tracking output.

Algorithm 1: Iterative Proposal Generation

Input: Node set $\mathcal{V}$, iterative number I, maximum cluster size s_{max}, cluster threshold step Δ.
Output: Proposal set $\mathcal{P}$

initialization: $\mathcal{P} = \varnothing$, $i = 0$, $\mathcal{V}' = \mathcal{V}$
while $i < I$ **do**
  $\mathcal{G} = BuildAffinityGraph(\mathcal{V}')$;
  $\mathcal{C} = ClusterNodes(\mathcal{G}, s_{max}, \Delta)$;
  $\mathcal{P} = \mathcal{P} \cup \mathcal{C}$;
  $\mathcal{V}' = UpdateNodes(\mathcal{C})$;
  $i = i + 1$;
end
Return $\mathcal{P}$

B. Parameter Sensitivity Analysis

Here, we investigate the effects of different settings on parameter s_{max}, Δ and K (the maximum number of edges linked to one vertex) to the tracking performance. The parameter s_{max} and Δ are used to control the growth speed of the proposals. The results in Figure 1 and Figure 2 show that we can choose $s_{max} \in [2, 4]$, $\Delta \in [0.02, 0.06]$ to achieve the satisfactory and stable performance. With the the increasing s_{max} or Δ, more noises will be introduced to the proposals in early iterations, hence reducing the performance. The parameter K controls the number of edges in the graph construction. The results in Figure 3 show that a satisfactory and stable performance can be achieved when $K > 1$.

C. Evaluation Results on MOT16

We also report the quantitative results obtained by our method on MOT16 in Table 1 and compare it to methods that are officially published on the MOTChallenge benchmark. Our method can also obtain state-of-the-art IDF1 score on MOT16.

D. Qualitative Analysis

Figure 4 and Figure 5 give a qualitative comparison between MPNTrack[3] and our method on MOT17. It validates that our method has better performance in handling long-term occlusions, hence achieving higher IDF1 score.

Algorithm 2: Cluster Nodes

Input: Symmetric affinity matrix $\mathcal{G}$, maximum cluster size s_{max}, cluster threshold step Δ.
Output: Clusters $\mathcal{C}$

function *main***:**
  $\mathcal{C} = \varnothing, \mathcal{R} = \varnothing, \tau = min(\mathcal{G})$;
  $\mathcal{C}', \mathcal{R} = FindClucters(\mathcal{G}, \tau, s_{max})$;
  $\mathcal{C} = \mathcal{C} \cup \mathcal{C}'$;
  while $\mathcal{R} \neq \varnothing$ **do**
    $\tau = \tau + \Delta$;
    $\mathcal{C}', \mathcal{R} = FindClucters(\mathcal{G}_{\mathcal{R}}, \tau, s_{max})$;
    $\mathcal{C} = \mathcal{C} \cup \mathcal{C}'$;
  end
  return $\mathcal{C}$;
function $FindClucters\,(\mathcal{G}, \tau, s_{max})$ **:**
  $\mathcal{G}' = PruneEdge(\mathcal{G}, \tau)$;
  $\mathcal{S} = FindConnectedComponents(\mathcal{G}')$;
  $\mathcal{C}' = \{c | c \in \mathcal{S},\ |c| < s_{max}$ and $Compatible(c)\}$;
  $\mathcal{R} = \mathcal{S} \backslash \mathcal{C}'$;
  return $\mathcal{C}', \mathcal{R}$;
function $Compatible\,(c)$ **:**
  if $\mathbf{d}(\mathbf{t}_i, \mathbf{t}_j) > 0, \forall i, j \in c, i \neq j$ **then**
    return True ;
  else
    return False ;
  end

Algorithm 3: De-overlapping

Input: Ranked Proposals $\{\hat{\mathcal{P}}_1, \hat{\mathcal{P}}_2, \cdots, \hat{\mathcal{P}}_{N_p}\}$
Output: Tracking Results $\mathbb{T}$
Dictionary $\mathbb{T} = \{\}$, Occupied Set $\mathbf{I} = \varnothing, i = 1$;
while $i <= N_p$ **do**
  $\mathcal{C}_i = \hat{\mathcal{P}}_i \backslash \mathbf{I}$;
  for v_i *in* $\mathcal{C}_i$ **do**
    $\mathbb{T}[v_i] = i$;
  end
  $\mathbf{I} = \mathbf{I} \cup \mathcal{C}_i$;
  $i = i + 1$;
end
Return $\mathbb{T}$;

E. Further Performance Comparison

We also noticed that MPNTrack [3] used a different ReIdentification (ReID) model from our method. In order to achieve a completely fair comparison, we also provide the comparison results between our method and MPNTrack using our ReID model on the training set of MOT17. Table 2 shows the detailed results. By comparing our method with MPNTrack[2], it is clear that our method achieves better performance on identity preservation, improving the IDF1

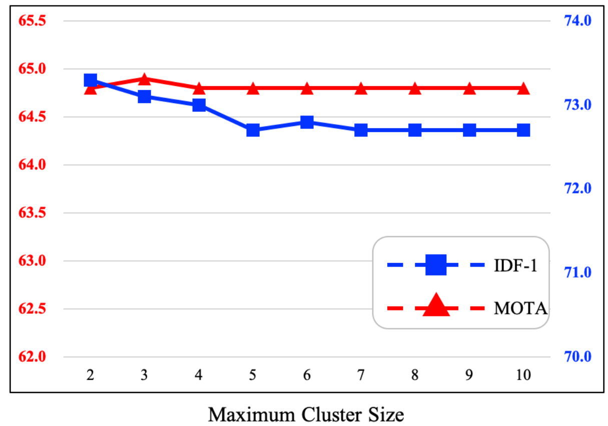


Figure 1. Influence of the maximum cluster size s_{max} on proposal generation performance.

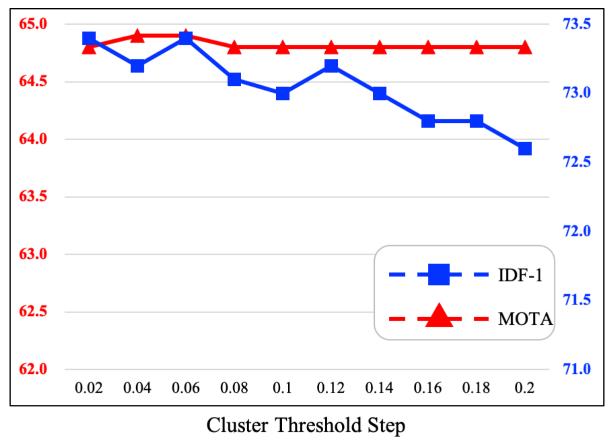


Figure 2. Influence of the cluster threshold step Δ on proposal generation performance.

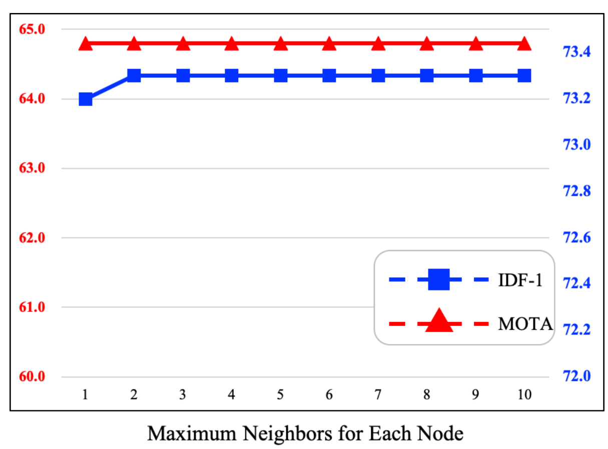
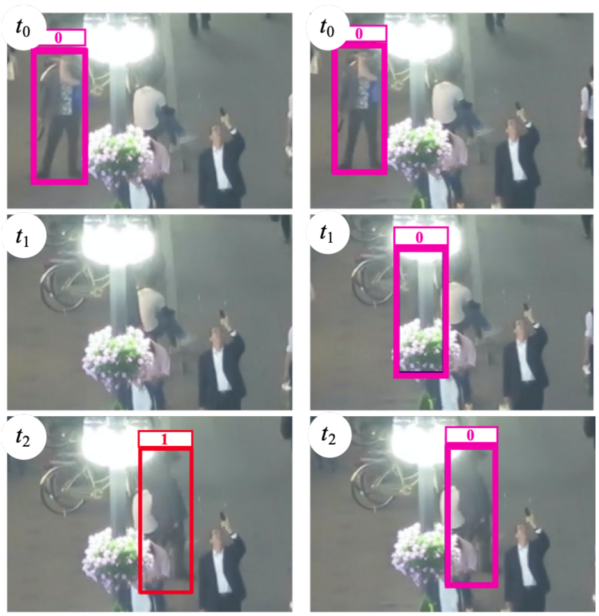


Figure 3. Influence of the maximum neighbors for each node K on proposal generation performance.

Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDs↓	Hz↑
Ours	58.8	**67.6**	**27.3**	35.0	6167	68432	435	4.3
Lif_T [5]	61.3	64.7	27.0	34.0	4844	65401	389	0.5
MPNTrack [3]	58.6	61.7	**27.3**	34.0	4949	70252	**354**	6.5
HDTR [1]	53.6	46.6	21.2	37.0	4714	79353	618	3.6
TPM [9]	51.3	47.9	18.7	40.8	2701	85504	569	0.8
CRF_TRACK [10]	50.3	54.4	18.3	35.7	7148	82746	702	1.5
NOTA [4]	49.8	55.3	17.9	37.7	7248	83614	614	**19.2**
UnsupTrack [6]	**62.4**	58.5	27.0	**31.9**	5909	**61981**	588	1.9
GNNMatch [8]	57.2	55.0	22.9	34.0	3905	73493	559	0.3
Tracktor [2]	56.2	54.9	20.7	35.8	**2394**	76844	617	1.6
TrctrD16 [11]	54.8	53.4	19.1	37.0	2955	78765	645	1.6
PV [7]	50.4	50.8	14.9	38.9	2600	86780	1061	7.3

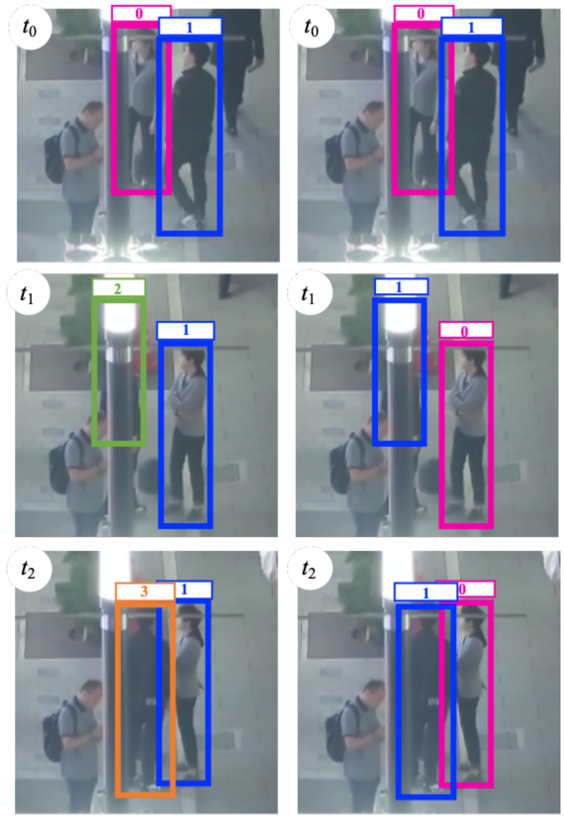
Table 1. Performance comparison with start-of-the art on MOT16 (top: offline methods; bottom: online methods).



(a) (b)

Figure 4. A qualitative example showing (a) a failure case of MPNTrack[3] in handling long-term occlusions, which reduces the IDF1 score; (b) our method can effectively handle this case. The numbers are the object IDs. Best viewed in color.

score by 1.5 percentage. By comparing MPNTrack[1] with MPNTrack[2], we can see that the overall performance gap is small. In summary, our method can achieve better association accuracy than MPNTrack [3].



(a) (b)

Figure 5. A qualitative example showing (a) a failure case of MPN-Track [3] in handling occlusions, which leads to an identity transfer when one person passes the other and a fragmentation when one is fully occluded; (b) our method can effectively handle this case. The numbers are the object IDs. Best viewed in color.

Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDs↓
Ours	63.9	**71.8**	647	377	7176	**113700**	728
MPNTrack[1]	**64.0**	70.0	**648**	**362**	**6169**	114509	602
MPNTrack[2]	63.9	70.3	634	365	6228	114723	**523**

[1] with their own ReID model
[2] with our ReID model

Table 2. Further performance comparison on the training set of MOT17.

References

[1] Maryam Babaee, Ali Athar, and Gerhard Rigoll. Multiple people tracking using hierarchical deep tracklet re-identification. *arXiv preprint arXiv:1811.04091*, 2018. 3

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *ICCV*, pages 941–951, 2019. 3

[3] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *CVPR*, pages 6247–6257, 2020. 1, 2, 3

[4] Long Chen, Haizhou Ai, Rui Chen, and Zijie Zhuang. Aggregate tracklet appearance features for multi-object tracking. *IEEE Signal Processing Letters*, 26(11):1613–1617, 2019. 3

[5] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *International Conference on Machine Learning*, pages 4364–4375, 2020. 3

[6] Shyamgopal Karthik, Ameya Prabhu, and Vineet Gandhi. Simple unsupervised multi-object tracking. *arXiv preprint arXiv:2006.02609*, 2020. 3

[7] Xuesong Li, Yating Liu, Kunfeng Wang, Yong Yan, and Fei-Yue Wang. Multi-target tracking with trajectory prediction and re-identification. In *2019 Chinese Automation Congress (CAC)*, pages 5028–5033, 2019. 3

[8] Ioannis Papakis, Abhijit Sarkar, and Anuj Karpatne. Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization. *arXiv preprint arXiv:2010.00067*, 2020. 3

[9] Jinlong Peng, Tao Wang, Weiyao Lin, Jian Wang, John See, Shilei Wen, and Erui Ding. Tpm: Multiple object tracking with tracklet-plane matching. *PR*, 107:107480, 2020. 3

[10] Jun Xiang, Guohan Xu, Chao Ma, and Jianhua Hou. End-to-end learning deep crf models for multi-object tracking deep crf models. *CSVT*, 31(1):275–288, 2020. 3

[11] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *CVPR*, pages 6787–6796, 2020. 3