Relational Prior for Multi-Object Tracking

Anonymous ICCV submission

Paper ID ****

Abstract

Tracking multiple objects individually differs from tracking groups of related objects. When an object is a part of the group, its trajectory is conditioned on the trajectories of the other group members. Most of the current state-of-the-art trackers follow the approach of tracking each object independently, with the mechanism to handle the overlapping trajectories where necessary. Such an approach does not take inter-object relations into account, which may cause unreliable tracking for the members of the groups, especially in crowded scenarios, where individual cues become unreliable. To overcome these limitations, we propose a plug-in Relation Encoding Module (REM). REM encodes relations between tracked objects by running a message passing over a spatio-temporal graph of tracked instances, computing the relation embeddings. The relation embeddings then serve as a prior for predicting future positions of the objects. Our experiments on MOT17 and MOT20 benchmarks demonstrate that extending a tracker with relational prior improves tracking quality.

1. Introduction

For online multi-object tracking, when objects are part of a group, the frequent mutual occlusions make individual tracking harder. Rather than rejecting that information, identifying group membership is interesting by itself, where in principle the group is easier to identify having more uniquely identifying characteristics than an individual object would. In this paper we set out to exploit group relations for multi-object tracking.

When tracking pedestrians online in a crowd, following one specifically is generally harder than following all members of a family of three, just because their *combination* offers good distinction: one tall with one small person, each with a trolley. Occlusion may hamper complete view of one of the targets but then the characteristics of related members may be borrowed to render approximate tracking for the occluded one like parents with a child in shopping malls and other forms of crowd control.

Figure 1. Top: tracking with Tracktor [1], where independent trajectories are assumed. Dense bodily interaction causes tracking failure. Bottom: extending the tracker with relational prior makes it more robust.

Multi-object online tracking has recently made great progress with tracking-by-regression [1, 23, 22, 10, 20, 21, 19]. These methods track each object independently until, at a crossroad of tracks, a mechanism is called upon to determine which object continues on what track. The current methods demonstrate good speed and good accuracy. They do not, however, consider inter-object relations, which may cause tracking to become unreliable especially when the interaction between bodies becomes dense where occlusion becomes a major obstacle, as in (Figure 1).

We draw inspiration from multi-object processing, where the whole video is available for the analysis. In [15, 16, 8, 7, 17], the trajectories are derived by running a graph optimization on the object detections. While the structure of the graph encodes the inter-object relations in these offline trackers, their capability of finding relations relies heavily on having all detections in the video at once, combining information *before and after* dense interactions. This offline information blocks the methods unsuited for online multi-object tracking as the detections of the future are not yet available.

In this work, we extend current tracking-by-regression methods with online group relations. Inspired by offline graph-based video analysis, we learn to encode inter-object relations from limited data *a priori*. In our relation encoding module, a message passing algorithm is running over a dynamic object-graph to produce relation embeddings, which encode the group structure for each object. The resulting relation embeddings serve as a prior for predicting positions of the objects. The relation encoding module is implemented as a plug-in extension for tracking-by-regression methods.

To sum up, we *(i)* develop a method to encode inter-object relations online in dense scenes by running spatial-temporal message passing, *(ii)* we demonstrate the virtue of relational prior to improve the tracking of objects by adding relations on top of current tracking-by-regression methods.

2. Related work

Multi-object tracking by graph association Many the multi-object trackers first apply an object detector on the whole sequence, then link the detections across frames on the basis of a best match criterion [15, 16, 8, 7, 17]. They follow the tracking by a graph association paradigm. The matching is usually posed as an offline graph association problem connecting the detections into trajectories. In [15, 16], the authors solve association as a multicut problem, where trajectories are derived from a dense graph of detections by extracting weighted subgraphs. Along the same lines, in [8], Keuper *et al*. propose a multicut formulation to decompose a dense detection graph into a set of trajectories. To better handle occlusions, Tang *et al*. [17] further extend multicuts with lifted edges.

Graph associations are powerful as they reason about groups of detections, while taking inter-object relations into account. However, their offline nature limits the real-life application of such methods. Also, due to their combinatorial non-differentiable formulation, it is not trivial to combine these graph association algorithms with modern end-to-end trackers. In this work, we take inspiration from these offline graph association works and develop a new method for relation encoding, which learns to encode the dynamics of multiple objects for online tracking. Our relation encoding module is fully end-to-end compatible with modern trackers.

Multi-object tracking by regression association Recently, a family of methods called tracking-by-regression has become the state-of-the-art approach in multi-object tracking. The key idea is to assess the association of detections to previously detected objects by utilizing the regression head of the object detector. In the pioneering work of Bergmann *et al*. [1], tracking is based on the second stage of the Faster R-CNN [13] object detector with the previous positions of detected objects as proposals. Later, more sophisticated object detectors were used [23, 10, 22, 20]. In [23] Zhou *et al*. modify CenterNet [24] for multi-object tracking. In [10], authors modify the lightweight RetinaNet [5] for faster inference. In [22, 20], the authors extend the detector with ReID embeddings, which allows for better identity preservation in case of occlusion.

In all these works, objects are tracked independently of one another. When scenes become crowded or filled with similar targets, independent tracking becomes hard or impossible. Whereas the above methods function well generally, they tend to break when individual cues are no longer available (Figure 1). To function in these hard but frequent circumstances, a tracking method has to employ prior knowledge about the dynamics of the objects. In this paper, we propose to extend the regression-based multi-object trackers with relations encoding, so they can jointly reason about the groups of the objects.

3. Encoding relations

To encode inter-object relations, the relation encoding module takes a set of tracked instances as input and produces *relation embeddings* by running a message passing algorithm over the spatial-temporal graph. Figure 2 renders the architecture of the module.

3.1. Building relational graph

We define $G_T = \{(V_t, E_t)_{t=1}^{T}; (Z_t)_{t=1}^{T-1}\}$ as a spatial-temporal graph, where V_t, E_t represent the vertices and edges of the graph at the time step t, respectively. Z_t is a set of temporal edges from t to $t+1$. Vertices correspond to the objects as tracked, while the temporal edges encode their trajectories. Only the nodes, which correspond to the same instance, are linked in time. To decide on the spatial edges at time step t, we first compute the distance matrix D^t with entries:

$$D_{ij}^t = \sqrt{\frac{(x_i^t - x_j^t)^2}{\bar{w}_{ij}^t} - \frac{(y_i^t - y_j^t)^2}{\bar{h}_{ij}^t}} \quad (1)$$

where $(x_i^t, y_i^t, w_i^t.h_i^t)$ corresponds to the center coordinates, width and height of the i-th object and $\bar{w}_{ij}^t = \min(w_i^t, w_j^t)$ respectively. We use the scaled Euclidean distance to prevent linking remote instances, which may be close if evaluated only by the center coordinates, but far away in depth. To obtain an adjacency matrix A we simply threshold the distances, i.e. $A_{ij}^t = \mathbb{1}[D_{ij}^t \leq d_{\text{th}}]$, where d_{th} is a hyper-parameter.

3.2. Graph-attention message passing

Inter-object relations are modulated by running message passing over the relational graph. The procedure consists of 4 steps: compute input node features, compute messages between spatial nodes, aggregate messages and compute spatial-temporal updates of node representations. This procedure is recurrently performed for each time step until the end of the graph is reached.

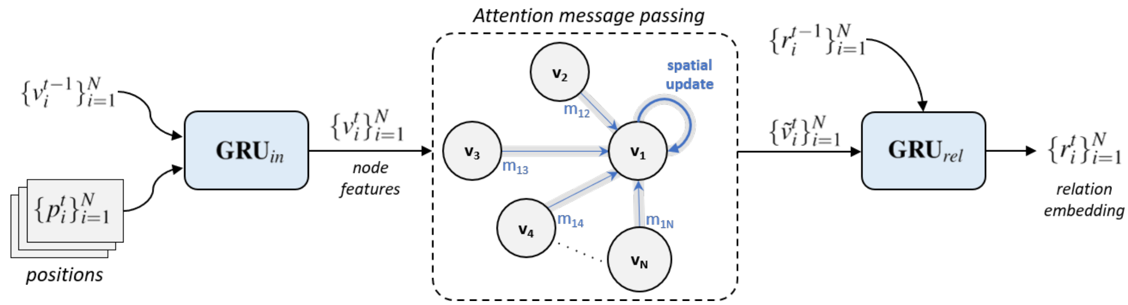

Figure 2. Computing relation embeddings $\{r_i^t\}_{i=1}^N$ for N object from time step $t-1$ to t. Input coordinates are passed through an input GRU-cell to produce node features. Message passing is performed on top of the relational graph to update internal node representations. Finally, node representations are passed through another GRU-cell, which emulates message passing along temporal edges.

Node features To construct the input feature we use bounding box coordinates of the detection and the positional offset with respect to the previous time step. Let $p_i^t \in \mathbb{R}^4$ be the bounding box of the *i-th* object at time t, the input feature $v_i^t \in \mathbb{R}^F$ for the node is then computed as:

$$\tilde{p}_i^t = \sigma(\mathbf{W}_{\text{in}}[p_i^t \| p_i^t - p_i^{t-1}] + \mathbf{b}_{\text{in}}) \quad (2)$$
$$v_i^t = \mathbf{GRU}_{\text{in}}(\tilde{p}_i^t, v_i^{t-1}) \quad (3)$$

where $\mathbf{W}_{\text{in}}, \mathbf{b}_{\text{in}}$ are learnable parameters, σ is a non-linearity and $\|$ denotes concatenation operator. The initial hidden states of the $\mathbf{GRU}_{\text{in}}$ cell are set to zeros.

Message sending A message between two nodes of the graph is designed to encode their pairwise interaction. We define the message as a function of both the sending and the receiving nodes i and j, respectively. To make the message aware of the geometry of the graph, we also include the distance D_{ij}^t between the objects as an additional input for the message function. The message $m_{ij}^t : \mathbb{R}^F \times \mathbb{R}^F \times \mathbb{R} \rightarrow \mathbb{R}^F$ is calculated as:

$$m_{ij}^t = \sigma\big(\mathbf{W}_{m_2}(\sigma(\mathbf{W}_{m_1}[v_i^t \| v_j^t \| D_{ij}^t] + \mathbf{b}_{m_1})) + \mathbf{b}_{m_2}\big) \quad (4)$$

Aggregating messages When the messages have been computed, they are gathered in an aggregated message. An aggregation function should be permutation equivariant with respect to the neighbors' features. In this work, we follow the graph attention approach [18], which computes attention between features to weigh them according to their importance. The attention mechanism $\alpha_{ij}^t : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}_+$ computes the attention coefficients as:

$$\alpha_{ij}^t = \frac{\exp\big(\text{LeakyReLU}\big([\mathbf{W}_{a_1} v_i^t]^T [\mathbf{W}_{a_2} v_j^t]\big)\big)}{\sum_{j \in \mathcal{N}_i} \exp\big(\text{LeakyReLU}\big([\mathbf{W}_{a_1} v_i^t]^T [\mathbf{W}_{a_2} v_j^t]\big)\big)} \quad (5)$$

where $\mathcal{N}_i$ denotes the set of the nodes *spatially* adjacent to *i-th* node in the graph. Temporal edges are not considered at this stage. The attention coefficients are then used to compute a linear combination of the corresponding neighbors' representation into an aggregated feature.

Spatial-Temporal update In the final step, we update node representations spatially and temporally. For the spatial update, we concatenate the self-feature of the node with the aggregated message from its neighbors and pass it through the perceptron. The temporal update is performed by passing the features through the GRU-cell. Formally:

$$\textit{(spatial)} \quad \tilde{v}_i^t = \sigma(\mathbf{W}_u[v_i^t \| \sum_{j \in \mathcal{N}_i} \alpha_{ij}^t v_j^t] + \mathbf{b}_u) \quad (6)$$

$$\textit{(temporal)} \quad r_i^t = \mathbf{GRU}_{\text{rel}}(\tilde{v}_i^t, r_i^{t-1}) \quad (7)$$

We call the resulting feature $r_i^t \in \mathbb{R}^F$ *relation embedding* of the *i-th* node at time t. Relation embeddings at $t = 0$ are all set to zero vectors.

As can be seen in Equations 6, 7, in our implementation the temporal updates follow the spatial update. Early experiments demonstrated that such an approach is slightly superior to when the temporal update is performed first.

3.3. Tracking with relational prior

Next, we extend tracking-by-regression models to reason about the object's position based both on appearance and relation cues. To do so, we condition the predicted positions of the objects on their relation priors. To that end, we concatenate the appearance features extracted from proposal regions with the relation embeddings of the corresponding objects. The positional offset is then predicted by passing the combined feature via the regression head of the object detector. The model can be seen as the REM with the tracker attached to graph nodes. Such a framework applies to a wide range of trackers [1, 23, 22, 10, 20]. It does not require modification of the tracker other than adjusting the regression head.

4. Experiments

We evaluate relation-aware tracker on the MOTChallenge benchmarks MOT17 [12] and MOT20 [4].

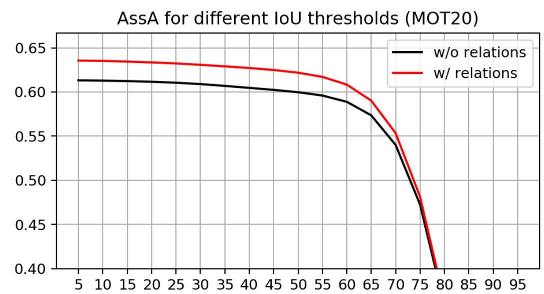

Figure 3. Association Accuracy (AssA) of the relation-aware and the baseline tracker over different localization thresholds.

4.1. Implementation details

We use Tracktor[1] as our baseline model as it provides good speed-accuracy balance. We extend Tracktor to relation-aware RelTracktor by plugging in the REM. To do so, we modify the regression head of the tracker to take the concatenated relation-appearance feature instead of just the appearance feature as the input. The rest of the tracker remains unchanged.

We use Xavier initialization [6] for the relation encoding module. We also initialize the modified regression head from the backbone tracker. We then jointly train the modified regression head and the relational module. To do so, we randomly sample $T = 10$ consecutive frames from MOT17/MOT20 datasets, compute relation embeddings and feed them into the regression head together with appearance features to refine bounding boxes at time step $T + 1$. We train for 50 epochs using the Adam optimizer [9] with a learning rate of 0.0001 while setting $d_{\text{th}} = 15$ to build relational graphs. We choose $F = 128$ for a dimension of the relation embedding vectors. Generalized intersection over union [14] is used as a loss function. We highlight that only the relational module and the regression head are trained, while the rest of the model is kept as is.

4.2. Datasets and evaluation metrics

The MOT17 benchmark consists of 7 train and 7 test sequences, which contain pedestrians with annotated full-body bounding boxes. The MOT20 benchmark contains 4 train and 4 test sequences of moving pedestrians in unconstrained environments with bounding boxes, covering the visible part of the objects.

Following [1], we evaluate the multi-object tracking quality in a public detection setting. We employ standard the MOT-metrics [2] and the HOTA metric [11] as an indicator of the overall performance.

[1] https://github.com/phil-bergmann/tracking_wo_bnw

	Method	HOTA ↑	IDF1 ↑	MOTA ↑	MOTP ↑	MT ↑	ML ↓
MOT17	**RelTracktor** *(Ours)*	**45.8**	**56.5**	**57.2**	**79.0**	21.9	**34.3**
	Tracktor [1]	44.8	55.1	56.3	78.8	21.1	35.3
	deepMOT [21]	42.4	53.8	53.7	77.2	19.4	36.6
MOT20	**RelTracktor** *(Ours)*	**43.4**	**53.0**	**54.1**	79.2	**36.7**	**22.6**
	Tracktor [1]	42.1	52.7	52.6	**79.9**	29.4	26.7
	SORT20 [3]	36.1	45.1	42.7	78.5	16.7	26.2

Table 1. Performance comparison on MOT17 and MOT20. The relation-aware RelTracktor model outperforms the baseline model with no relations on both benchmarks.

4.3. Relation-aware tracking-by-regression

We compare the relation-aware RelTracktor versus the baseline method from [1]. We run the tracker on the test subset of MOT benchmarks and submit results to the evaluation server. Results are presented in Table 1.

On the MOT17-benchmark, the relation-aware tracker shows an improvement in all metrics compared to the Tracktor baseline. In particular, a higher IDF1 score indicates that our model robustly preserves the identities of the objects throughout the sequence, while also providing more accurate localization as indicated by the MOTP score. On the MOT20-benchmark, the relation-aware tracker demonstrates 1.3% increase in the overall HOTA score. Although the baseline tracker provides slightly higher localization precision as indicated by MOTP score, its relation-aware extension is more robust and is able to track targets longer as indicated by the higher percentage of mostly tracked objects (MT).

To investigate the ability of the relation-aware model to provide robust tracking in the dense scenes, we analyze the Association Accuracy (AssA) of the tracker over various localization thresholds (Figure 3). Low localization thresholds permit the association of loose bounding boxes. It can deteriorate the association quality when predicted bounding boxes are densely overlapped, which is a common case in crowded scenarios. Thus, the higher association accuracy of relation-aware tracker (Figure 3) under low localization thresholds indicates the better ability to preserve identities of densely interacting objects.

5. Discussion

In this work, we demonstrate that utilizing inter-object relations is important for robust multi-object tracking. We develop a plug-in relation encoding module, which encodes relational prior by running a message passing over a spatial-temporal graph of tracked instances. We experimentally demonstrate that extending a backbone multi-object tracker with relational cues improves tracking accuracy and robustness. We suppose that our approach would be the most useful in problems, where video analysis of crowded scenes is required.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 4

[2] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *J. Image Video Process.*, 2008, Jan. 2008. 4

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 4

[4] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes, 2020. 3

[5] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018. 2

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feed forward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR, 2010. 4

[7] Margret Keuper, Siyu Tang, Bjoern Andres, Thomas Brox, and Bernt Schiele. Motion segmentation multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):140–153, 2020. 1, 2

[8] Margret Keuper, Siyu Tang, Yu Zhongjie, Bjoern Andres, Thomas Brox, and Bernt Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects, 2016. 1, 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4

[10] Zhichao Lu, Vivek Rathod, Ronny Votel, and Jonathan Huang. Retinatrack: Online single stage joint detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3

[11] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, pages 1–31, 2020. 4

[12] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831, 2016. 3

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 2

[14] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. 2019. 4

[15] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5033–5041, 2015. 1, 2

[16] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *Computer Vision – ECCV 2016 Workshops*, pages 100–111, 2016. 1, 2

[17] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *CVPR*, pages 3701–3710, 2017. 1, 2

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. 3

[19] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-object tracking and segmentation. In *CVPR*, 2019. 1

[20] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019. 1, 2, 3

[21] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6787–6796, 2020. 1, 4

[22] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020. 1, 2, 3

[23] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ECCV*, 2020. 1, 2, 3

[24] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 2