International Journal of Computer Vision (2022) 130:3035–3053
https://doi.org/10.1007/s11263-022-01678-6

Multi-Object Tracking and Segmentation Via Neural Message Passing

Guillem Brasó[1] · **Orcun Cetintas**[1] · **Laura Leal-Taixé**[1]

Received: 6 October 2021 / Accepted: 27 August 2022 / Published online: 26 September 2022

Abstract
Graphs offer a natural way to formulate Multiple Object Tracking (MOT) and Multiple Object Tracking and Segmentation (MOTS) within the tracking-by-detection paradigm. However, they also introduce a major challenge for learning methods, as defining a model that can operate on such *structured domain* is not trivial. In this work, we exploit the classical network flow formulation of MOT to define a fully differentiable framework based on Message Passing Networks. By operating directly on the graph domain, our method can reason globally over an entire set of detections and exploit contextual features. It then jointly predicts both final solutions for the data association problem and segmentation masks for all objects in the scene while exploiting synergies between the two tasks. We achieve state-of-the-art results for both tracking and segmentation in several publicly available datasets. Our code is available at https://github.com/ocetintas/MPNTrackSeg

Keywords Multi-object tracking · Segmentation · Neural message passing · Graph neural networks

1 Introduction

Multiple object tracking (MOT) is the task of determining the bounding box trajectories of all object instances in a video. Multi-object tracking and segmentation (MOTS) (Voigtlaender et al., 2019) extends this task to pixel-level precision by forming trajectories with instance segmentation masks. Both tasks constitute fundamental problems in computer vision, with applications such as autonomous driving, biology, and video analysis.

Segmentation and tracking are naturally intertwined tasks. When occlusions among objects occur during tracking, masks can disambiguate their relative position in a more explicit manner than boxes. Even though we can expect improved performance from exploiting the synergies between the tasks, current state-of-the-art barely takes advantage of their interactions.

In recent years, *tracking-by-detection* has been the dominant paradigm among state-of-the-art methods in MOT. This two-step approach consists of first obtaining frame-by-frame object detections and then linking them to form trajectories. While the first task can be addressed with learning-based detectors (Ren et al., 2015; Redmon and Farhadi, 2017), the latter, data association, is generally formulated as a graph partitioning problem (Tang et al., 2017; Yu et al., 2007; Zhang et al., 2008; Leal-Taixé et al., 2011; Berclaz et al., 2011). In this graph view of MOT, a node represents an object detection, and an edge represents the connection between two nodes. An active edge indicates that two detections belong to the same trajectory. Solving the graph partitioning task, i.e., finding the set of active edges or trajectories, can also be decomposed into two stages. First, a cost is assigned to each edge in the graph encoding the likelihood of two detections belonging to the same trajectory. After that, these costs are used within a graph optimization framework to obtain the optimal graph partition.

Recent MOTS approaches often design a single model to perform both tasks. A common approach is to employ an additional segmentation module on top of the tracking pipeline and predict masks together with the trajectories (Voigtlaender et al., 2019; Porzi et al., 2020; Meinhardt et al., 2021; Qiao et al., 2021). Despite using a single net-

Communicated by Wanli Ouyang.

Guillem Brasó and Orcun Cetintas have contributed equally to this work.

✉ Orcun Cetintas
orcun.cetintas@tum.de

Guillem Brasó
guillem.braso@tum.de

Laura Leal-Taixé
leal.taixe@tum.de

[1] Technical University of Munich, Munich, Germany

work, these approaches use separate modules without any explicit interaction to perform segmentation and data association, preventing any synergies between these two tasks from arising. An alternative line of work specifically takes advantage of the segmentation masks to extract richer scene information and improve association performance (Luiten et al., 2020a; Xu et al., 2020b). However, these approaches require an independent segmentation model; hence, they do not simultaneously train for both tasks and therefore cannot adaptively extract segmentation features for data association. All in all, fusing tracking and segmentation cues with a unified model is an under-explored research direction.

We propose to jointly solve data association and segmentation with a unified learning-based solver that can extract and combine relevant appearance, geometry, and segmentation features and reason over the entire scene. This enables a framework in which association benefits from refined visual information provided by segmentation masks.

With this motivation, we exploit the classical network flow formulation of MOT (Zhang et al., 2008) to define our model. Instead of learning pairwise costs, using these within an available solver and independently predicting masks for every object, our method learns to propagate association and segmentation features across the graph in order to directly predict final partitions of the graph together with masks for every object. We perform learning directly in the graph domain with a message passing network (MPN) (Gilmer et al., 2017). We design a neural message passing framework that allows our model to learn to combine association and segmentation features into high-order information across the graph. Therefore, we exploit synergies among all tasks in a fully learnable manner while relying on a simple graph formulation. We show that our unified framework yields substantial improvements with respect to the state-of-the-art both in MOT and MOTS domains without requiring heavily engineered features.

This work builds upon our previous CVPR paper (Braso and Leal-Taixe, 2020) and extends it by 1) integrating an attentive module to our neural message passing scheme to yield a unified model for multi-object tracking and segmentation and 2) providing an extensive evaluation of our tracking model over three challenging datasets, including MOT20 (Dendorfer et al., 2020), KITTI Geiger et al. (2012a) and the recently proposed Human in Events dataset (Lin et al., 2020).

To summarize, we make the following **contributions**:

- We propose a multi-object tracking and segmentation solver based on message passing networks, which can exploit the natural graph structure of the tracking problem to perform both feature learning as well as final solution prediction.
- We propose a novel time-aware neural message passing update step inspired by classic graph formulations of MOT.
- We present a unified framework capable of performing joint tracking and segmentation by combining cues from both domains to improve the association performance. To this end, we propose an attentive message passing update that aggregates inferred temporal and spatial information.
- We achieve state-of-the-art results in eight public MOT and MOTS benchmarks.

2 Related Work

Most state-of-the-art MOT works follow the tracking-by-detection paradigm which divides the problem into two steps: (i) detecting pedestrian locations independently in each frame, for which neural networks are currently the state-of-the-art (Ren et al., 2015; Redmon and Farhadi, 2017; Yang et al., 2016), and (ii) linking corresponding detections across time to form trajectories.

Tracking as a Graph Problem. Data association can be done on a frame-by-frame basis for online applications (Breitenstein et al., 2009; Ess et al., 2008; Pellegrini et al., 2009) or track-by-track (Berclaz et al., 2006). For video analysis tasks that can be done offline, batch methods are preferred since they are more robust to occlusions. The standard way to model data association is by using a graph, where each detection is a node, and edges indicate possible links among them. The data association can then be formulated as a maximum flow (Berclaz et al., 2011) or, equivalently, a minimum cost problem with either fixed costs based on distance (Jiang et al., 2007; Pirsiavash et al., 2011; Zhang et al., 2008), including motion models (Leal-Taixé et al., 2011), or learned costs (Leal-Taixé et al., 2014). Both formulations can be solved optimally and efficiently. Alternative formulations typically lead to more involved optimization problems, including minimum cliques (Zamir et al., 2012) and lifted disjoint paths (Hornakova et al., 2020, 2021), general-purpose solvers, e.g., multi-cuts (Tang et al., 2017). A recent trend is to design ever more complex models which include other vision input such as reconstruction for multi-camera sequences (Leal-Taixé et al., 2012; Wu et al., 2011), activity recognition (Choi and Savarese, 2012), segmentation (Milan et al., 2015), keypoint trajectories (Choi, 2015) or joint detection (Tang et al., 2017).

Learning in Graph-based Tracking. It is no secret that neural networks are now dominating the state-of-the-art in many vision tasks since (Krizhevsky et al., 2012) showed their potential for image classification. The trend has also arrived in the tracking community, where learning has been used primarily to learn a mapping from image to optimal costs for the aforementioned graph algorithms. The authors of (Leal-Taixé et al., 2016) use a siamese network to directly learn the

costs between a pair of detections, while a mixture of CNNs and recurrent neural networks (RNN) is used for the same purpose in Sadeghian et al. (2017); Zhang et al. (2020). The authors of Ristani and Tommasi (2018) show the importance of learned ReIdentification (ReID) features for multi-object tracking. More recently, JDE and FairMOT (Wang et al., 2020; Zhang et al., 2021) explore the potential of jointly learning appearance features data association and detection, therefore, improving efficiency. All aforementioned methods learn the costs independently from the optimization method that actually computes the final trajectories. In contrast, (Kim et al., 2013; Wang and Fowlkes, 2015; Schulter et al., 2017) incorporate the optimization solvers into learning. The main idea behind these methods is that costs also need to be optimized for the solver in which they will be used. (Kim et al., 2013; Wang and Fowlkes, 2015; Frossard and Urtasun, 2018) rely on structured learning losses while (Schulter et al., 2017) proposes a more general bi-level optimization framework. These works can be seen as similar to ours in spirit, given our common goal of incorporating the full inference model into learning for MOT. However, we follow a different approach towards this end: we propose to directly learn a solver and treat data association as a classification task, while their goal is to adapt their methods to perform well with non-learnable solvers. Moreover, all these works are limited to learning either pairwise costs (Frossard and Urtasun, 2018; Schulter et al., 2017) or additional quadratic terms (Wang and Fowlkes, 2015; Kim et al., 2013) but cannot incorporate higher-order information as our method. Instead, we propose to leverage the common graph formulation of MOT as a domain in which to perform learning. Concurrent work also explores the potential of learning in the graph domain (Liu et al., 2020; Li et al., 2020); however, they do so within frame-by-frame settings, while we focus on a more general graph formulation. Moreover, our proposed approach has inspired recent graph neural network-based work to tackle proposal classification for Multiple Hypothesis Tracking (Dai et al., 2021) and graph-matching-based tracking (He et al., 2021).

Regression-Based Tracking. While graph-based methods are still actively used by several state-of-the-art trackers, several recent works achieve remarkable performance with a simpler formulation. The first of such works was Tracktor Bergmann et al. (2019) which performs tracking by exploiting the regressor head of a Faster R-CNN to sequentially predict the location objects in consecutive frames. TMOH Stadler and Beyerer (2021) improves Tracktor's performance under occlusion, and CTracker Peng et al. (2020) builds upon its idea with a framework that performs chained regression over consecutive pairs of frames and uses attention. In a similar fashion to these works, CenterTrack Zhou et al. (2020) uses center-points instead of bounding boxes and predicts offset heatmaps to regress consecutive object locations, and PermaTrack Tokmakov et al. (2021) improves its robustness to occlusions by predicting offsets over multiple future frames. In our approach, we leverage Tracktor as an initial preprocessing step to improve object detections before applying our neural solver over the result.

Deep Learning on Graphs. Graph Neural Networks (GNNs) were first introduced in Scarselli et al. (2009) as a generalization of neural networks that can operate on graph-structured domains. Since then, several works have focused on further developing and extending them by developing convolutional variants (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016). More recently, most methods were encompassed within a more general framework termed neural message passing (Gilmer et al., 2017) and further extended in Battaglia et al. (2018) as graph networks. Given a graph with some initial features for nodes and optionally edges, the main idea behind these models is to embed nodes (and edges) into representations that take into account not only the node's features but also those of its neighbors in the graph, as well as the overall graph topology. These methods show remarkable performance in a wide variety of areas, ranging from chemistry (Gilmer et al., 2017) to combinatorial optimization (Li et al., 2018). Within vision, they have been successfully applied to problems such as human action recognition (Guo et al., 2018), visual question answering (Narasimhan et al., 2018) or single object tracking (Gao et al., 2019).

Multi-Object Tracking and Segmentation. MOTS (Voigtlaender et al., 2019) was recently introduced as an extension of MOT to overcome the limitations of bounding boxes while also opening up the possibility to incorporate a richer visual cue for the association step. Prior works often solve both tasks by modifying strong segmentation models and extending them with an association head (Voigtlaender et al., 2019; Porzi et al., 2020; Qiao et al., 2021; Wu et al., 2021). TrackR-CNN (Voigtlaender et al., 2019) integrates temporal information into the MaskR-CNN framework with 3D convolutions, while MOTSNet (Porzi et al., 2020) employs a mask-pooling layer for filtering out the background information from the instance feature maps. Alternatively, recent works attempt to condition the association step on masks by first performing segmentation and then reconstructing objects in the 3D space (Luiten et al., 2020a) or denoting objects with 2D point clouds (Xu et al., 2020b). The motivation of this line of work is to utilize the strong visual cues provided by masks to improve the association performance of the model. However, these approaches disentangle the mask prediction step from the association. Thus, simultaneously solving both tasks by fusing tracking and segmentation cues remains an open question.

3 Tracking as a Graph Problem

Our method's tracking formulation is based on the classical min-cost flow view of MOT (Zhang et al., 2008). In order to provide some background and formally introduce our approach, we start by providing an overview of the network flow MOT formulation. We then explain how to leverage this framework to reformulate the data association task as a learning problem.

3.1 Problem Statement

In tracking-by-detection, we are given as input a set of object detections $\mathcal{O} = \{o_1, \dots, o_n\}$, where n is the total number of objects for all frames of a video. Each detection is represented by $o_i = (a_i, p_i, t_i)$, where a_i denotes the raw pixels of the bounding box, p_i contains its 2D image coordinates and t_i its timestamp. A trajectory is defined as a set of time-ordered object detections $T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$, where n_i is the number of detections that form trajectory i. The goal of MOT is to find the set of trajectories $\mathcal{T}_* = \{T_1, \dots, T_m\}$, that best explains the observations $\mathcal{O}$. The problem can be modelled with an undirected graph $G = (V, E)$, where $V := \{1, \dots, n\}$, $E \subset V \times V$, and each node $i \in V$ represents a unique detection $o_i \in \mathcal{O}$. The set of edges E is constructed so that every pair of detections, i.e., nodes, in different frames is connected, hence allowing to recover trajectories with missed detections. Now, the task of dividing the set of original detections into trajectories can be viewed as grouping nodes in this graph into disconnected components. Thus, each trajectory $T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$ in the scene can be mapped into a group of nodes $\{i_1, \dots, i_{n_i}\}$ in the graph and vice-versa.

3.2 Network Flow Formulation

In order to represent graph partitions, we introduce a binary variable for each edge in the graph. In the classical minimum cost flow formulation (Zhang et al., 2008), this label is defined to be 1 between edges connecting nodes that (i) belong to the same trajectory, and (ii) are temporally consecutive inside a trajectory; and 0 for all remaining edges.

A trajectory $T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$ is equivalently denoted by the set of edges $\{(i_1, i_2), \dots, (i_{n_i-1}, i_{n_i})\} \subset E$, corresponding to its time-ordered path in the graph. We will use this observation to formally define the edge labels. For every pair of nodes in different timestamps, $(i, j) \in E$, we define a binary variable $y_{(i,j)}$ as:

$$y_{(i,j)} := \begin{cases} 1 & \exists T_k \in \mathcal{T}_* \text{ s.t. } (i, j) \in T_k \\ 0 & \text{otherwise.} \end{cases}$$

An edge (i, j) is said to be *active* whenever $y_{(i,j)} = 1$. We assume trajectories in $\mathcal{T}$ to be node-disjoint, i.e., a node cannot belong to more than one trajectory. Therefore, y must satisfy a set of linear constraints. For each node $i \in V$:

$$\sum_{(j,i)\in E \text{ s.t. } t_i > t_j} y_{(j,i)} \leq 1 \tag{1}$$

$$\sum_{(i,k)\in E \text{ s.t. } t_i < t_k} y_{(i,k)} \leq 1 \tag{2}$$

These inequalities are a simplified version of the *flow conservation constraints* (Ahuja et al., 1993). In our setting, they enforce that every node gets linked via an active edge to, at most, one node in past frames and one node in upcoming frames.

3.3 From Learning Costs to Predicting Solutions

In order to obtain a graph partition with the framework we have described, the standard approach is to first associate a cost $c_{(i,j)}$ to each binary variable $y_{(i,j)}$. This cost encodes the likelihood of the edge being active (Leal-Taixé et al., 2014, 2016; Schulter et al., 2017). The final partition is found by optimizing:

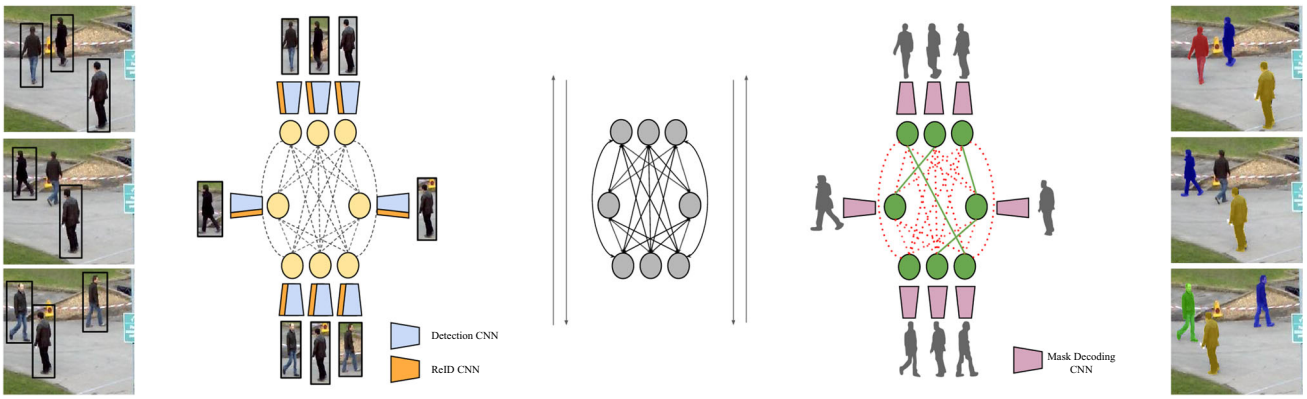
$$\begin{aligned} &\min_y && \sum_{(i,j)\in E} c_{(i,j)} y_{(i,j)} \\ &\text{Subject to:} && \text{Equation (1)} \\ &&& \text{Equation (2)} \\ &&& y_{(i,j)} \in \{0, 1\}, \quad (i, j) \in E \end{aligned}$$

which can be solved with available solvers in polynomial time (Berclaz et al., 2006; Ahuja et al., 1993).

We propose to, instead, directly learn to predict which edges in the graph will be active, i.e., predict the final value of the binary variable y. To do so, we treat the task as a classification problem over edges, where our labels are the binary variables y. Overall, we exploit the classical network flow formulation we have just presented to treat the MOT problem as a fully learnable task.

4 Learning to Track with Message Passing Networks

Our main contribution is to exploit the graph formulation described in the previous section to design a differentiable framework for joint tracking and segmentation. Given a set of detections, we design a neural message passing network that operates on its underlying graph and extracts contextual node and edge embeddings. We classify each edge embedding to predict the values of the binary *flow* variables y directly, and



(a) Input **(b)** Graph Construction + Feature Encoding **(c)** Neural Message Passing **(d)** Edge Classification + Mask Prediction **(e)** Output

Fig. 1 Overview of our method. **a** We receive as input a set of frames and detections. **b** We construct a graph in which nodes represent detections, and all nodes at different frames are connected by an edge. **c** We initialize node embeddings in the graph with two CNNs that encode appearance and mask features. Edge embeddings are initialized with an MLP encoding geometry information (not shown in the figure). **c** The information contained in these embeddings is propagated across the graph for a fixed number of iterations through neural message passing. **d** Once this process terminates, the embeddings resulting from neural message passing are used to predict masks and classify edges into active (colored with green) and non-active (colored with red). During training, we compute the cross-entropy loss of our predictions w.r.t. ground truth labels. **e** At inference, we follow a simple rounding scheme to binarize our classification scores and obtain final trajectories (Color figure online)

we exploit node embeddings to obtain a segmentation mask for every target. Our method is based on a novel message passing network (MPN) and is able to capture the graph structure of the MOT and MOTS problems. Within our proposed MPN framework, appearance, geometry, and segmentation cues are propagated across the entire set of detections, allowing our model to reason globally about the entire graph.

4.1 Overview

Our method consists of the following four main stages: **1. Graph Construction:** Given a set of object detections in a video, we construct a graph where nodes correspond to detections and edges correspond to connections between nodes (Sect. 3.2).
2. Feature Encoding: We initialize the node feature embeddings from two convolutional neural networks (CNNs) applied on every detection's Region of Interest (RoI), extracting appearance and mask features. For each edge, i.e., for every pair of detections in different frames, we compute a vector with features encoding their bounding box relative size, position, and time distance. We then feed it to a multilayer perceptron (MLP) that returns a *geometry* embedding (Sect. 4.5).
3. Neural Message Passing: We perform a series of message passing steps over the graph. Intuitively, for each round of message passing, nodes share appearance information with their connecting edges, and edges share geometric information with their incident nodes. This yields updated embeddings for nodes and edges containing *higher-order* information that depends on the overall graph structure (Sects. 4.2, 4.3 and 4.4).
4. Classification: We use the final edge embeddings to perform binary classification into active/non-active edges and node embeddings to classify each pixel of the RoIs into foreground/background. We train our model using the cross-entropy loss for both tasks (Sect. 4.6).

At test time, we use our model's prediction per edge as a continuous approximation (between 0 and 1) of the target *flow* variables. We then follow a simple scheme to round them and obtain the final trajectories. For a visual overview of our pipeline, see Fig. 1.

4.2 Message Passing Networks

In this section, we provide a brief introduction to MPNs based on the work presented in Gilmer et al. (2017); Kipf et al. (2018); Battaglia et al. (2016); Battaglia et al. (2018). Let $G = (V, E)$ be a graph. Let $h_i^{(0)}$ be a node embedding for every $i \in V$, and $h_{(i,j)}^{(0)}$ an edge embedding for every $(i, j) \in E$. The goal of MPNs is to learn a function to propagate the information contained in nodes and edge feature vectors across G. The propagation procedure is organized in embedding updates for edges and nodes, which are known as *message passing steps* (Gilmer et al., 2017). In (Battaglia et al., 2018; Kipf et al., 2018; Battaglia et al., 2016), each message passing step is divided, in turn, into two updates: one from nodes to edges ($v \rightarrow e$), and one from edges to

nodes ($e \rightarrow v$). The updates are performed sequentially for a fixed number of iterations L. For each $l \in \{1, \ldots, L\}$, the general form of the updates is the following (Battaglia et al., 2018):

$$(v \rightarrow e) \qquad h_{(i,j)}^{(l)} = \mathcal{N}_e \left([h_i^{(l-1)}, h_j^{(l-1)}, h_{(i,j)}^{(l-1)}] \right) \tag{3}$$

$$(e \rightarrow v) \qquad m_{(i,j)}^{(l)} = \mathcal{N}_v \left([h_i^{(l-1)}, h_{(i,j)}^{(l)}] \right) \tag{4}$$

$$h_i^{(l)} = \Phi \left(\left\{ m_{(i,j)}^{(l)} \right\}_{j \in N_i} \right) \tag{5}$$

$\mathcal{N}_e$ and $\mathcal{N}_v$ represent learnable functions, e.g., MLPs, that are shared across the entire graph. [.] denotes concatenation, $N_i \subset V$ is the set of adjacent nodes to i, and Φ denotes an order-invariant operation, e.g., a summation, maximum, or an average. Note, after L iterations, each node contains information about all other nodes at a distance L in the graph. Hence, L plays an analogous role to the receptive field of CNNs, allowing embeddings to capture context information.

4.3 Time-Aware Message Passing

The previous message passing framework was designed to work on arbitrary graphs. However, MOT graphs have a very specific structure that we propose to exploit. Our goal is to encode a MOT-specific inductive bias in our network, specifically in the node update step. Recall the node update depicted in Eqs. 4 and 5, which allows each node to be compared with its neighbors and aggregate information from all of them to update its embedding with further context. Recall also the structure of our flow conservation constraints (Eqs. 1 and 2), which imply that each node can be connected to, at most, one node in future frames and another one in past frames. Arguably, aggregating all neighboring embeddings at once makes it difficult for the updated node embedding to capture whether these constraints are being violated or not (see Sect. 5.4 for constraint satisfaction analysis). More generally, explicitly *encoding* the temporal structure of MOT graphs into our MPN formulation can be a useful prior for our learning task. Towards this goal, we modify Eqs. 4 and 5 into time-aware update rules by dissecting the aggregation into two parts: one over nodes in the past, and another over nodes in the future. Formally, let us denote the neighboring nodes of i in future and past frames by N_i^{fut} and N_i^{past}, respectively. Let us also define two different MLPs, namely, $\mathcal{N}_v^{fut}$ and $\mathcal{N}_v^{past}$. At each message passing step l and for every node $i \in V$, we start by computing *past* and *future* edge-to-node embeddings for all of its neighbors $j \in N_i$ as:

$$m_{(i,j)}^{(l)} = \begin{cases} \mathcal{N}_v^{past} \left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_{(i)}^{(0)}] \right) & \text{if} \quad j \in N_i^{past} \\ \mathcal{N}_v^{fut} \left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_{(i)}^{(0)}] \right) & \text{if} \quad j \in N_i^{fut} \end{cases} \tag{6}$$

Note, the initial embeddings $h_{(i)}^{(0)}$ have been added to the computation. This skip connection ensures that our model does not *forget* its initial features during message passing, and we apply it analogously with initial edge features in Eq. 3. After that, we aggregate these embeddings separately, depending on whether they were in future or past positions with respect to i:

$$h_{i,past}^{(l)} = \sum_{j \in N_i^{past}} m_{(i,j)}^{(l)} \tag{7}$$

$$h_{i,fut}^{(l)} = \sum_{j \in N_i^{fut}} m_{(i,j)}^{(l)} \tag{8}$$

Now, these operations yield *past* and *future* embeddings $h_{i,past}^{(l)}$ and $h_{i,fut}^{(l)}$, respectively. We compute the final updated node embedding by concatenating them and feeding the result to one last MLP, denoted as $\mathcal{N}_v$:

$$h_i^{(l)} = \mathcal{N}_v([h_{i,past}^{(l)}, h_{i,fut}^{(l)}]) \tag{9}$$

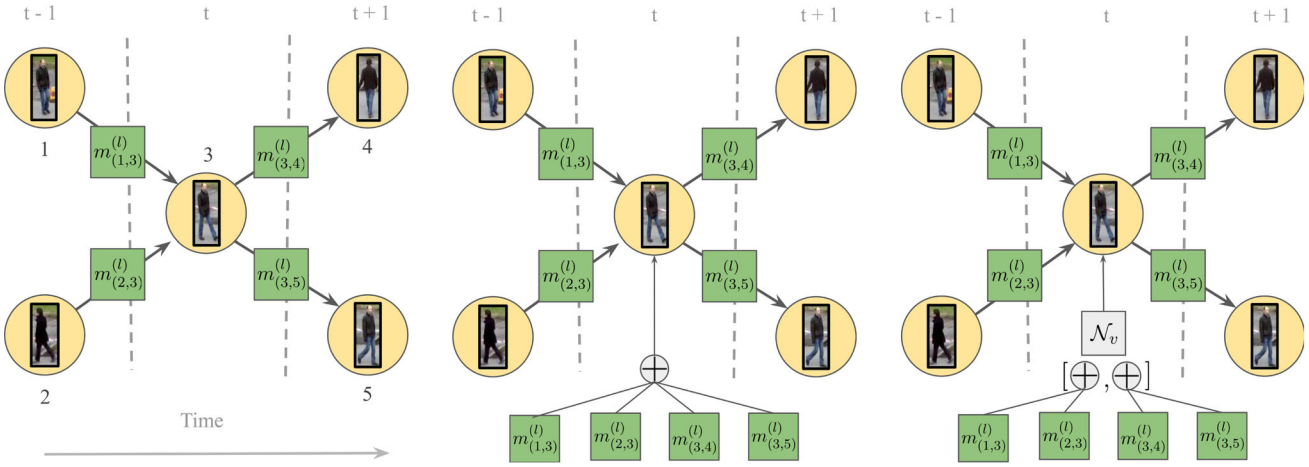
We summarize our time-aware update in Fig. 2c. As we demonstrate experimentally (see Sect. 5.4), this simple architectural design results in a significant performance improvement with respect to the *vanilla* node update of MPNs, shown in Fig. 2c.

4.4 Attentive Message Passing

Our time-aware message passing framework utilizes appearance and geometry feature vectors of dimension d for the association. Segmentation, on the other hand, is a dense prediction task that requires pixel-precise outputs. Preserving spatial information is crucial for this task, therefore we incorporate contextual mask features with additional spatial dimensions H and W (i.e., tensors in $\mathbb{R}^{H \times W \times d'}$) for every node into our message passing updates. H and W correspond, respectively, to the height and width at which each node's RoI is resized.

Our goal is to extract rich segmentation features encoding temporal information to be used for association. Towards this end, we leverage the temporal information encoded in the edge embeddings and use them to produce attention coefficients to guide the updates of mask features. The mask features will therefore be most influenced by neighbors in the graph belonging to the same trajectory.

Formally, let $\tilde{h}_i^{(0)} \in \mathbb{R}^{H \times W \times d}$ represent a secondary mask node embedding for every $i \in V$ encoding visual information. In addition, let $\mathcal{N}_e^w$ denote an MLP working on the edges of the graph. At each message passing step $l \geq 1$, we calculate the unnormalized attention weights for each edge of the graph:



(a) Initial Setting **(b)** Vanilla node update **(c)** Time-aware node update

Fig. 2 Visualization of node updates during message passing. Arrow directions in edges show time direction. Note the time division in $t-1$, t, and $t+1$. In this case, we have $N_3^{past} = \{1, 2\}$ and $N_3^{fut} = \{4, 5\}$. **a** shows the starting point after an edge update has been performed (Eq. 3), and the intermediate node update embeddings (Eq. 4) have been computed. **b** shows the standard node update in vanilla MPNs, in which all neighbors' embeddings are aggregated jointly. **c** shows our proposed update, in which embeddings from past and future frames are aggregated separately, then concatenated and fed into an MLP to obtain the new node embedding

$$w^{(l)}_{(i,j)} = \mathcal{N}^w_e\left(h^{(l-1)}_{(i,j)}\right) \tag{10}$$

The attention scores for the edge (i, j) are computed by respecting the time-aware message passing rules and normalizing the weights over the *past* and *future* neighbors of the node i with a softmax operator:

$$a^{(l)}_{(i,j)} = \begin{cases} \dfrac{\exp\left(w^{(l)}_{(i,j)}\right)}{\sum_{k \in N_i^{past}} \exp\left(w^{(l)}_{(i,k)}\right)} & \text{if} \quad j \in N_i^{past} \\ \dfrac{\exp\left(w^{(l)}_{(i,j)}\right)}{\sum_{k \in N_i^{fut}} \exp\left(w^{(l)}_{(i,k)}\right)} & \text{if} \quad j \in N_i^{fut} \end{cases} \tag{11}$$

The *past* and *future* context tensors for each node i are, then, computed as a weighted sum of the node features of its neighbors:

$$c^{(l)}_{i,past} = \sum_{j \in N_i^{past}} a^{(l)}_{(i,j)} \tilde{h}^{(l-1)}_j \tag{12}$$

$$c^{(l)}_{i,fut} = \sum_{j \in N_i^{fut}} a^{(l)}_{(i,j)} \tilde{h}^{(l-1)}_j \tag{13}$$

Finally, we obtain the updated node embeddings by concatenating initial embedding $\tilde{h}^{(0)}_i$ with the *past* and *future* context tensors and feeding the result to a 2-layer CNN, denoted as $\tilde{\mathcal{N}}_v$:

$$\tilde{h}^{(l)}_i = \tilde{\mathcal{N}}_v([c^{(l)}_{i,past}, c^{(l)}_{i,fut}, \tilde{h}^{(0)}_{(i)}]) \tag{14}$$

Note that high-dimensional secondary node embeddings $\tilde{h}_i$ and attentive update rules are only employed in the MOTS setting. Default node embeddings h_i are still present in this scenario and are updated according to Eqs. 6–9.

We illustrate our update scheme in Fig. 3. With our attentive message passing updates high-dimensional mask features can participate in the message passing steps, and we can train our model for tracking and segmentation jointly. It is worth highlighting that updates of these node features are directly governed by the edges of the graph. Hence, in our end-to-end differentiable framework, both the edge features $h^{(l)}_{(i,j)}$ (later classified for tracking) and node features $\tilde{h}^{(l)}_i$ (later used for segmentation) are guided by both the tracking and segmentation objectives. Overall, we obtain a joint framework in which segmentation features can guide association decisions. In Sect. 5.4, we empirically show that our unified pipeline improves upon our baseline in which segmentation and tracking are detached.

4.5 Feature Encoding

The initial embeddings that our MPN receives as input are produced by other backpropagatable networks.

Appearance and Mask Embeddings. We rely on a CNN, denoted as $\mathcal{N}^{enc}_v$, to *learn* to extract feature embeddings directly from RGB data. For every detection $o_i \in \mathcal{O}$, and its corresponding image patch a_i, we obtain o_i's corresponding node embedding by computing $h^{(0)}_i := \mathcal{N}^{enc}_v(a_i)$. Moreover,

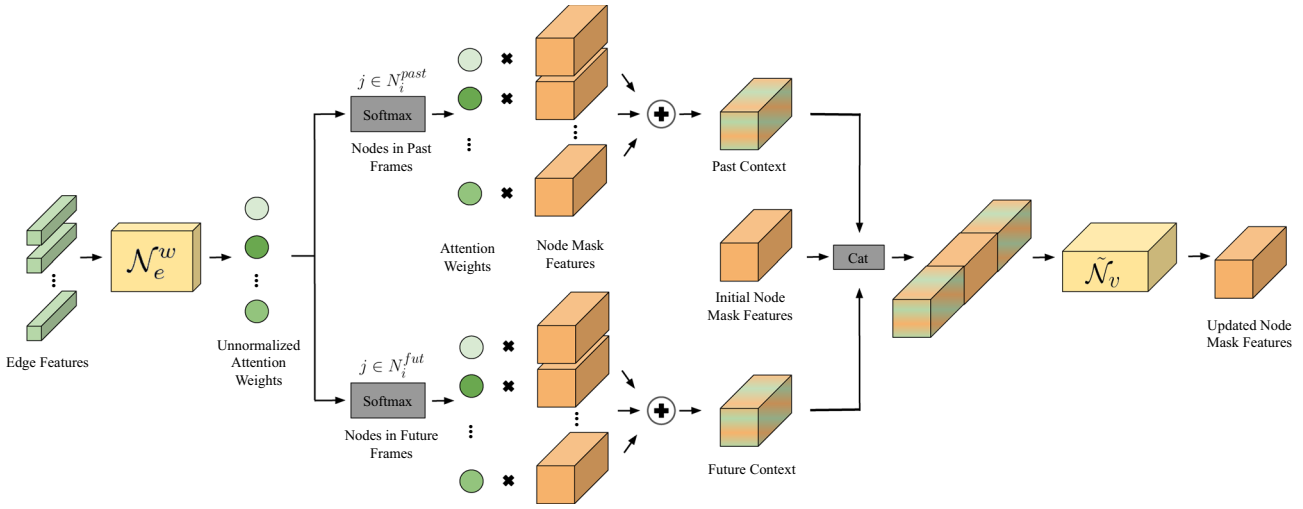


Fig. 3 Visualization of our proposed attentive message passing. Updates of the node mask features (orange) are guided by the attention weights, obtained from the edge features (green) (Color figure online)

we use an additional network composed of a CNN backbone and RoI Align (He et al., 2017) denoted as $\tilde{\mathcal{N}}_v^{enc}$ to obtain secondary node embeddings from each Region of Interest (RoI) r_i as $\tilde{h}_i^{(0)} := \tilde{\mathcal{N}}_v^{enc}(r_i)$.

Geometry Embedding. For each pair of detections in different frames, we seek to obtain a representation that encodes their relative position, size, as well as distance in time. For every pair of detections o_i and o_j with timestamps $t_i \neq t_j$, we consider their bounding box coordinates parameterized by top left corner image coordinates, height and width, i.e., (x_i, y_i, h_i, w_i) and (x_j, y_j, h_j, w_j). We compute their relative distance and size as:

$$\left(\frac{2(x_j - x_i)}{h_i + h_j}, \frac{2(y_j - y_i)}{h_i + h_j}, \log\frac{h_i}{h_j}, \log\frac{w_i}{w_j}\right)$$

We then concatenate this coordinate-based feature vector with the time difference $t_j - t_i$ and relative appearance $\|\mathcal{N}_v^{enc}(a_j) - \mathcal{N}_v^{enc}(a_i)\|_2$ and feed it to a neural network $\mathcal{N}_e^{enc}$ in order to obtain the initial edge embedding $h_{(i,j)}^{(0)}$.

4.6 Training and Inference

Training Loss. To classify edges, we use an MLP with a sigmoid-valued single output unit, sharing weights with $\mathcal{N}_e^w$. For training, we use the weighted binary cross-entropy of our predictions over the embeddings produced in the last m message passing steps, with respect to the target flow variables ($\mathcal{L}_t$). Since our classification problem is severely imbalanced, we weight positive terms in the loss with the inverse proportion of positive samples.

Masks are obtained by feeding the updated node features to a CNN that classifies each pixel of the RoI as foreground/background with a per-pixel sigmoid. We define $\mathcal{L}_s$ as the average binary cross-entropy loss over the pixels. Similar to $\mathcal{L}_t$, we compute this loss for the last m message passing steps. Finally, we define a multi-task loss as $\mathcal{L} = \mathcal{L}_t + \mathcal{L}_s$, which is the main objective of our training.

Tracking Inference. During inference, we interpret the set of output values obtained from our model at the last message passing step as the solution to our data association problem. An easy way to obtain hard 0 or 1 decisions is to binarize the output by thresholding. However, this procedure does not generally guarantee that the flow conservation constraints in Eqs. 1 and 2 are preserved. In practice, thanks to the proposed time-aware update step, our method will satisfy over 98% of the constraints on average when thresholding at 0.5. After that, a simple rounding scheme suffices to obtain a feasible binary output. We consider the significantly smaller subgraph consisting of nodes and edges involved in a violation of Eqs. 1 and 2 and solve a linear program with it to ensure the satisfaction of constraints 1 and 2. Given the high constraint satisfaction rate obtained by our model, the runtime of this procedure adds an insignificant computational overhead. As shown in Braso and Leal-Taixe (2020), the linear program can be replaced by a simple heuristic rounding procedure without loss of tracking performance.

Mask prediction. To obtain our final masks, we use a 7-layer CNN, denoted as $\tilde{\mathcal{N}}_v^{mask}$, to predict binary masks from the node mask features for each RoI. Our segmentation network $\tilde{\mathcal{N}}_v^{mask}$ receives two sets of features, namely the updated node features $\tilde{h}_i^{(l)}$ via attentive message passing, and the raw node features $\tilde{h}_i^{(0)}$ prior to message passing updates, which can be thought of as a skip connection:

$$mask_i = \tilde{\mathcal{N}}_v^{mask}([\tilde{h}_{(i)}^{(l)}, \tilde{h}_{(i)}^{(0)}]) \tag{15}$$

5 Experiments

In this section, we start by presenting ablation studies to understand the behavior of our model better. We then compare our model to the published methods on several datasets and show state-of-the-art results. For the MOT experiments, we use our default time-aware message passing framework, as introduced in our conference paper, and denote it as MPNTrack. For MOTS experiments, we extend our model to include also attentive message passing as explained in Sect. 4.4 and denote our model as MPNTrackSeg. Implementation details of the models are reported in Sect. 5.3.

5.1 Evaluation Metrics

To evaluate our method, we report the CLEAR MOT (Kasturi et al., 2009), IDF1 (Ristani et al., 2016) and HOTA (Luiten et al., 2020b), together with the percentage of mostly tracked (MT) and mostly lost targets (ML). For the CLEAR MOT metrics, we use the multiple object tracking accuracy (MOTA), which combines false positive detections (FP), false negative detections (FN), and identity switches (IDs) into a single score. Despite its widespread use, MOTA accounts mostly for the quality of detections and is not affected significantly by identity preservation errors (Ristani et al., 2016). IDF1, instead, is based on matching predicted trajectories to ground truth trajectories, instead of boxes, and therefore provides a better measure of data association performance. Lastly, the recently proposed HOTA score finds a balance between detection and data association performance by being decomposable into Detection Accuracy (DetA) and Association Accuracy (AssA). Hence, it can provide more clarity into the sources of errors committed by different trackers. (Voigtlaender et al., 2019) adapts CLEAR MOT metrics for MOTS by accounting for the segmentation masks. Specifically, they replace the bounding box IoU with a mask-based IoU and propose multi-object tracking and segmentation accuracy (MOTSA) and soft multi-object tracking and segmentation accuracy (sMOTSA). MOTSA utilizes the number of true positives that reaches an IoU of 0.5 whereas, sMOTSA accumulates the soft number of true positives to incorporate segmentation quality even more into MOTA.

5.2 Datasets

MOTChallenge. The multiple object tracking benchmark MOTChallenge consists of several challenging pedestrian tracking sequences, with frequent occlusions and crowded scenes. The challenge includes four datasets *2D MOT 2015* (Leal-Taixé et al., 2015), *MOT16* (Milan et al., 2016), *MOT17* (Milan et al., 2016) and *MOT20* (Dendorfer et al., 2020). They contain sequences lasting from 3 seconds to over 2 minutes, and with varying viewing angles, size, number of objects, camera motion, and frame rate. MOT20 is notable for its extreme crowdedness, with close to 150 pedestrians per frame, on average. MOTA and IDF1 scores are the most important metrics in this benchmark.

KITTI. The KITTI Vision Benchmark Suite (Geiger et al., 2012) focuses on robotics applications and includes sequences of autonomous driving scenarios captured both in rural and urban areas and on highways. KITTI accommodates challenging computer vision benchmarks, including optical flow, visual odometry, and multi-object tracking. The tracking benchmark contains 21 sequences for training and 29 for testing, recorded at 10 frames per second. Not all sequences contain pedestrians, and those that do are characterized by their low object density. KITTI has recently incorporated HOTA as the main metric, and it also makes use of MOTA.

MOTS20 and KITTI-MOTS (Voigtlaender et al., 2019). Recently, (Voigtlaender et al., 2019) extended both MOTChallenge and KITTI sequences for MOTS. MOTS20 consists of 4 training and 4 testing MOT16 sequences annotated with high-resolution instance masks. KITTI-MOTS adds segmentation masks for both pedestrians and cars to all KITTI sequences. As for metrics, sMOTSA and IDF1 are the main metrics to evaluate performance in MOTS20, whereas KITTI-MOTS bases its evaluation on domain-adapted HOTA that accounts for the mask quality.

Human in Events (HiEve) (Lin et al., 2020). The HiE dataset is a recently proposed benchmark focused on pedestrian tracking, detection, pose estimation, and action recognition in diverse surveillance scenarios, often characterized by heavy occlusions. It is the largest dataset currently available for pedestrian tracking, containing over 1.3 million annotated boxes across 19 sequences for training and 13 for testing, with an average trajectory length of over 480 frames. It uses two of the same metrics used in MOTChallenge, MOTA and IDF1, and does not accommodate HOTA.

5.3 Implementation Details

Network Models. For the network $\mathcal{N}_v^{enc}$ used to encode detections appearances (see Sect. 4.5), we employ a ResNet50 (He et al., 2016) architecture pretrained on ImageNet (Deng et al., 2009), followed by global average pooling and two fully-connected layers to obtain embeddings of dimension 256. We train the network for the task of ReIdentification (ReID) jointly on three publicly available datasets: Market1501 (Zheng et al., 2015), CUHK03 (Li et al., 2014) and DukeMTMC (Ristani et al., 2016). Note that using external ReID datasets is a common practice among MOT methods (Tang et al., 2017; Kim et al., 2018; Ma et al., 2019).

Once trained, three new fully connected layers are added after the convolutional layers to reduce the embedding size of $\mathcal{N}_v^{enc}$ to 32. To obtain secondary node features, we use a

COCO pretrained ResNet50-FPN backbone (He et al., 2016; Lin et al., 2017), referred as $\tilde{\mathcal{N}}_v^{enc}$.

Data Augmentation. To train our network, we sample batches of graphs. Each graph corresponds to small clips with a fixed number of frames. For MOT experiments, we use 15 frames per graph sampled at six frames per second for static sequences and nine frames per second for those with a moving camera. For MOTS experiments, on the other hand, we use 30 frames per graph without any sampling. We perform data augmentation by randomly removing nodes from the graph, hence simulating missed detections, and randomly shifting bounding boxes.

Training. We have empirically observed that additional training of the ResNet blocks provides no significant increase in performance, but carries a significantly larger computational overhead. Hence, during training, we freeze all convolutional layers and train jointly all of the remaining model components. We train for 15000 iterations for MOT and fifteen epochs for MOTS with a learning rate of $3 \cdot 10^{-4}$, weight decay term of 10^{-4} and an Adam Optimizer with β_1 and β_2 set to 0.9 and 0.999, respectively.

Batch Processing. We process videos offline in batches of n frames, with $n - 1$ of those overlapping, to ensure that the maximum time distance between two connected nodes in the graph remains stable across the whole graph. We prune graphs by connecting two nodes only if both are among the top-K mutual nearest neighbors according to the ResNet features. We set $n = 15$, $K = 50$ for MOT and $n = 30$, $K = 150$ for MOTS experiments. Each batch is solved independently by our network, and for overlapping edges and masks between batches, we average the predictions coming from all graph solutions. To fill gaps in our trajectories, we perform simple bilinear interpolation over the missing frames.

Baseline. Recently, (Bergmann et al., 2019) have shown the potential of detectors for simple data association, establishing a new baseline for MOT. We exploit this in our MOT version and preprocess all sequences by first running (Bergmann et al., 2019) on public detections, which allows us to be fully comparable to all methods on MOTChallenge. Note that this procedure has been adopted by several methods in the recent literature (Dai et al., 2021; Hornakova et al., 2020; He et al., 2021). One key drawback of (Bergmann et al., 2019) is its inability to fill in gaps, hence failing in properly recovering identities through occlusions. As we will show, this is exactly where our method excels. For the MOTS domain, however, we don't adopt this scheme with (Bergmann et al., 2019) and we use our model independently.

Detections For all MOT benchmarks in MOTChallenge and HiEve, we use the provided detections to ensure a fair comparison with other methods. For MOTS20 and KITTI benchmarks, we use bounding box detections obtained with a Mask R-CNN (He et al., 2017) with ResNeXt-152 backbone (Xie et al., 2017) trained on ImageNet and COCO (Lin et al., 2014), as these benchmarks do not accommodate public detections.

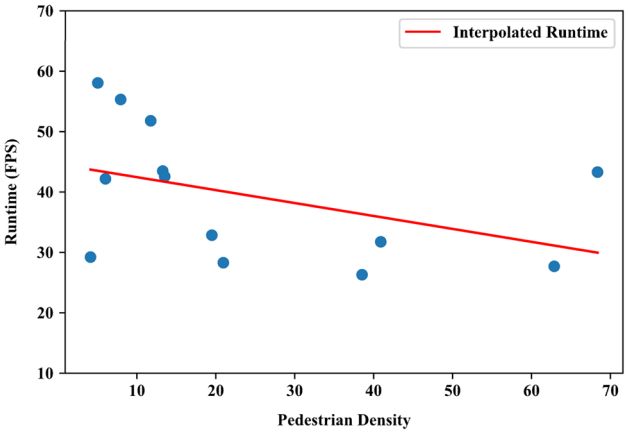


Fig. 4 We report our method's runtime in frames-per-second over sequences with varying pedestrian densities, as measured by average detections per frame

Runtime. We build our graph on the output of Bergmann et al. (2019) for MOT. Hence, we take also its runtime into account. Our method, on its own, runs at 35fps, while (Bergmann et al., 2019) without the added re-ID head runs at 8fps, which gives the reported average of 6.5fps on a single Nvidia P5000 GPU, running on a machine with 8 3.6GHz CPU cores. When we incorporate the attentive message passing scheme and segmentation head, our unified framework runs at 2.3fps on MOTS20. To provide further analysis, in Fig. 4 we report our method's runtime in fps over MOT15, MOT16, and MOT20 test sequences depending on their pedestrian density (in average detections per frame). While our method is naturally slower in denser sequences, it still shows a very competitive runtime in very crowded scenes containing over 60 pedestrians per frame.

5.4 Ablation Study

In this section, we aim to answer five main questions towards understanding our model. Firstly, we compare the performance of our time-aware neural message passing updates with respect to the time-agnostic vanilla node update described in Sect. 4.2. Secondly, we assess the impact of the number of message passing steps in network training to the overall tracking performance. Thirdly, we investigate how different information sources, namely, appearance embeddings from our CNN and relative position information, affect different evaluation metrics. Then, we quantify the impact of the attentive message passing explained in Sect. 4.4 by exploring the effect of the features used for mask prediction. Finally, we compare our tracking-only model and unified

Table 1 We investigate how our proposed update improves tracking performance with respect to a vanilla MPN

Arch.	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓	Constr. ↑
Vanilla	63.2	67.1	586	372	3239	119,853	917	83.2
T. aware	64.0	70.0	648	362	6169	114,509	602	98.8

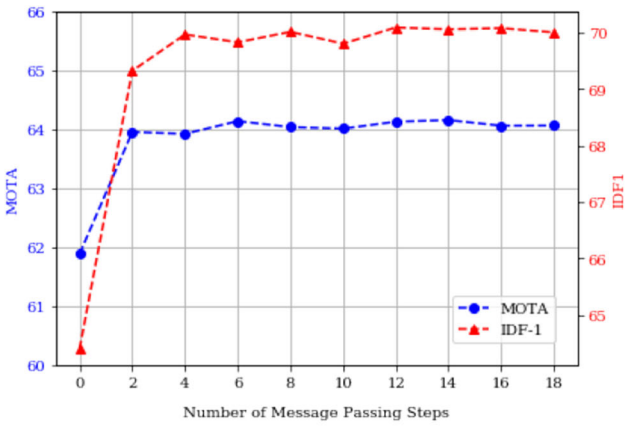
Vanilla stands for a basic MPN, *T. aware* denotes our proposed time-aware update. The metric *Constr* refers to the percentage of flow conservation constraints satisfied on average over entire validation sequences

model to demonstrate the effect of jointly training for tracking and segmentation.

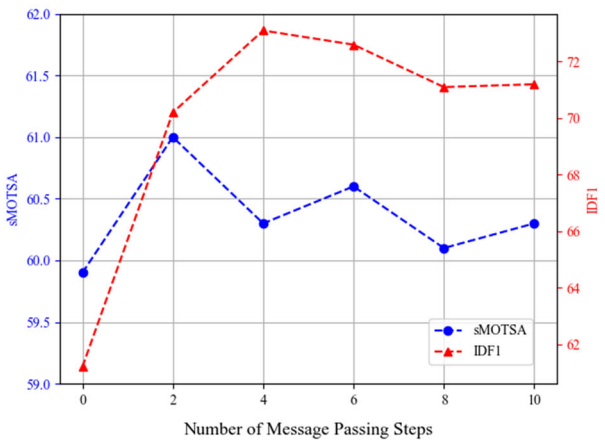
Experimental Setup. We conduct all of our experiments with the training sequences of the MOT15, MOT17, and MOTS20 datasets. To evaluate our models on MOT datasets, we split MOT17 sequences into three sets (sequences 2, 10 and 3; 4 and 11; and 5 and 9), and use these to test our models with 3-fold cross-validation. We then report the best overall MOT17 metrics obtained during validation. For MOTS results, we perform 4-fold cross-validation on MOTS20 and report the mean scores obtained in the best epochs by leaving one sequence out at a time.

Time-Aware Message Passing. We investigate how our proposed time-aware node update affects performance. For a fair comparison, we also add the skip connection with respect to initial edge features to the vanilla update, and increase the parameter count of its node update MLP to match the overall parameter count of the time-aware model. Still, we observe a significant improvement in almost all metrics, including close to 3 points in IDF1. As we expected, our model is particularly powerful at linking detections, since it exploits neighboring information and graph structure, making the decisions more robust, and hence producing significantly fewer identity switches. We also report the percentage of constraints that are satisfied when directly thresholding our model's output values at 0.5. Remarkably, our method with time-aware node updates is able to produce almost completely feasible results automatically, i.e., 98.8% constraint satisfaction (Table 1), while the baseline has only 83.2% satisfaction. This demonstrates its ability to capture the MOT problem structure.

Number of Message Passing Steps. Intuitively, increasing the number of message passing steps L allows each node and edge embedding to encode further context and gives edge predictions the ability to be iteratively refined. Hence, one would expect L values greater than zero to yield better-performing networks. We test this hypothesis in Fig. 5a by training networks with a fixed number of message passing steps, from 0 to 18. We use the case $L = 0$ as a baseline in which we train a binary classifier on top of our initial edge embeddings, and hence, no contextual information is used. As expected, we see a clear upward tendency for both IDF-1 and MOTA. Moreover, we observe a steep increase in both metrics from zero to two message passing steps, which demonstrates that the most improvement is obtained when switching from pairwise to high-order features in the graph. We also note that the upwards tendency stagnates after four message passing steps and shows no improvement after twelve message passing steps. We repeat the same experiment for MPNTrackSeg and report our findings in Fig. 5b. Similar to our previous findings, we observe an increase in IDF1 and sMOTSA from zero to two message passing steps. The upward trend continues until four steps, and the performance stagnates between four to six steps. Hence, we use $L = 12$ in our final configuration for MOT and $L = 4$ for MOTS.



(a) MPNTrack



(b) MPNTrackSeg

Fig. 5 We report the evolution of IDF1, MOTA and sMOTSA when training networks with an increasing number of message passing steps for MPNTrack and MPNTrackSeg

Effect of the Features. In the MOT setting, our model receives two main streams of information: (i) appearance information from a CNN and (ii) geometry features from an MLP encoding relative position between detections. We test their usefulness by experimenting with combinations of three groups of features for edges: time difference, relative position, and the euclidean distance in CNN embeddings between the two bounding boxes. Results are summarized in Table 2. We highlight the fact that relative position seems to be a key component of overall performance since its addition yields the largest relative performance increase. Nevertheless, CNN features are powerful to reduce the number of false positives and identity switches; hence, we use them in our final configuration.

Attentive Message Passing. We now evaluate the effect of our proposed attentive message passing updates for multi-object tracking and segmentation. As can be seen in Eq. 15, our model receives both updated node features from message passing and raw node features via a skip connection for mask prediction. In Table 3, we experiment with the combinations of these features. In addition, we compare these results with an additive message passing scheme that simply aggregates features from neighbors via summation instead of using attention. Note that only using the raw features via skip connections is equivalent to eliminating the message passing on mask features from our model and corresponds to using an independent segmentation network on top of our tracking-only version (MPNTrack), which serves as a baseline. Our graph-based framework already achieves competitive results while using skip connections with raw features alone. Performing message passing updates by simply summing neighboring node features is commonly used as an aggregation function in MPNs, however, this approach can not capture a rich visual representation required for our setting, as indicated by the lower sMOTSA score. We speculate that this drop is caused by the fact that incorporating segmentation features from neighboring nodes with equal weights harms our mask quality due to the outweighing caused by the large number of neighboring nodes belonging to different trajectories. With attention-based updates, we improve upon additive updates, highlighting the importance of taking identity information into account while performing message passing updates. Combining both raw features and attentive updates further boosts our performance, as indicated by the 1.1 points increase in IDF1 score and 15% drop in ML. Overall, while identity preservation scores get a significant boost from our attentive message passing module, we note that mask-quality related scores i.e., sMOTSA stay comparable to those of our baseline. We conclude that the main advantage of our joint framework lies in its ability to boost association performance by exploiting the rich appearance information that segmentation features provide.

Joint Training for Tracking and Segmentation. We compare MPNTrack and MPNTrackSeg in terms of data association performance to demonstrate the effect of joint training. For a fair comparison, we use the same parameters for both models and train them on MOTS20. Based on a shared set of ground truth, we use bounding box-based annotations for MPNTrack and mask-based annotations for MPNTrackSeg. After training, we disable the mask-related components of MPNTrackSeg. We base the evaluation on bounding boxes and report MOT metrics. As shown in Table 4, we observe +0.4 IDF1 and +0.2 MOTA improvements with our unified model. These observations suggest that association performance benefits from mask information and joint training within our unified framework.

5.5 Benchmark Evaluation

5.5.1 Multi-Object Tracking

MOTChallenge. In Table 5, we report our results in all four datasets in the MOTChallenge. We achieve competitive results and retain one of the fastest runtimes among published methods. We note that we are outperformed by two graph-based methods that were published after our approach: Lif_T and LPC_MOT. Lif_T, also follows a graph-based formulation, however, it uses a complex optimization scheme and additional engineered features such as DeepMatching (Revaud et al., 2016). As a result, it is over one order of magnitude slower than our approach. For LPC_MOT, we note that it builds over our work, and it also combines a message passing network with a graph-based approach. However, it replaces our network flow formulation with a significantly more involved multiple-hypothesis-based approach. We note that its runtime slows down significantly in crowded sequences such as those in MOT20, while our method retains the same speed and is, again, one order of magnitude faster. Lastly, we also are outperformed by a regression-based method that was published after our approach: TMOH. It follows a bounding box regression-based approach with a stronger backbone and is, therefore, able to reduce the amount of false negative detections in a way that is not available to us. Note, however, that it is also significantly slower than our method. Moreover, it follows an orthogonal direction to our approach, and potentially, both methods could be combined, similarly to how we integrated Tracktor into our approach.

KITTI. In Table 6 we report our pedestrian tracking results among published methods using 2D inputs on KITTI (i.e. no Lidar nor depth), and observe that we surpass all previous methods in terms of HOTA by a significant margin (+4.2 points). Our improvement is due to the data association capabilities of our model, as can be seen by a +5.7 improvement with respect to the previous best association accuracy

Table 2 We explore combinations of three sources of information for edge features: time difference in seconds (Time), relative position features (Pos) and the Euclidean distance between CNN embeddings of the two detections (CNN)

Edge Feats.	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
Time	58.8	52.6	529	372	13,127	122,800	2962
Time+CNN	62.3	64.5	641	363	10923	115,375	812
Time+Pos	63.6	68.7	631	365	6308	115,506	895
Time+Pos+CNN	64.0	70.0	648	362	6169	114,509	602

Table 3 We examine the contributions of the features and update schemes used for segmentation: initial node features before message passing updates (Raw) and updated node features (Upd.) after additive (Add.) and attentive (Att.) message passing steps

Method	Raw	Upd.	sMOTSA↑	IDF1↑	MOTSA↑	MT↑	ML↓	ID Sw.↓	MOTSP↑	TP↑
MPNTrack + Seg	✓		60.5	71.8	75.1	130	20	252	82.1	21,864
MPNTrackSeg-Add		Add.	57.3	72.2	73.2	127	21	227	80.2	21,651
MPNTrackSeg-Att		Att.	60.7	72.0	75.1	130	20	228	82.3	21,903
MPNTrackSeg-Att+R	✓	Att.	60.3	73.1	74.9	128	17	223	82.0	21,858

Table 4 We compare our tracking-only model (MPNTrack) and unified model (MPNTrackSeg) to demonstrate the effect of jointly training for tracking and segmentation

Edge Feats.	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
MPNTrack	62.1	68.2	134	17	5106	4867	232
MPNTrackSeg	62.3	68.6	129	20	4729	5143	271

We perform bounding-box-based evaluation and report MOT metrics

(AssA). We also note that our detection accuracy (DetA) is slightly lower than that of other newer methods, since we used off-the-shelf detections, and our method focuses solely on association. This can be further observed in our relatively lower MOTA, which is caused by a relatively larger number of False Negatives in the detections we used. However, as it can be seen from our overall HOTA score, our method has significantly superior overall tracking performance. These results show the versatility of our approach, and its ability to perform well in a setting for which it is not specialized: non-crowded autonomous driving scenes.

Human in Events. In Table 7 we report our results on the HiEve dataset. For a fair comparison, we exclude from this table competitors of the MM20' Grand Challenge (Lin et al., 2020), as they used multiple additional datasets for training, as well as model ensembles. Among published methods, we observe that our model achieves state-of-the-art performance in terms of key IDF1, Mostly Tracked, Mostly Lost, and ID switches. We note that these are the key measures of data association, which proves that our model excels at this task in the particularly crowded scenarios present in the HiEve dataset. We also note that our MOTA is below that of SiamMOT. This can be explained due to the fact that SiamMOT makes use of a heavy-weight Faster R-CNN detector based on DLA-169 (Yu et al., 2018) that is trained on an additional large-scale dataset, CrowdHuman (Shao et al., 2018), and therefore can find a better tradeoff between false positives and false negatives. This leads to an improved MOTA, despite having higher ID Switches. Instead, we use Tracktor for preprocessing with a ResNet50 backbone, and we use exclusively the provided training data in HiEve.

5.5.2 Multi-Object Tracking and Segmentation

Mask R-CNN MOTS Baselines. Following a similar approach to (Voigtlaender et al., 2019), we take several top-performing MOT methods that use public detections on the MOT17 benchmark and generate segmentation masks on their pre-computed outputs with a MaskR-CNN trained on COCO that shares the same backbone with $\tilde{N}_v^{enc}$. We report the results on the MOTS20 training set in Table 8. For a fair comparison, we run our model with MOT17 public detections in this experiment. Note that we use a stronger segmentation model with the baselines compared to Voigtlaender et al. (2019), and we improve on the original scores reported by them. It is worth highlighting that we report our 4-fold cross-validation results, whereas the baselines are already trained on these sequences together with additional sequences from the MOT17 training set. Remarkably, despite this disadvantage, our method outperforms the baselines. These results provide strong evidence that our unified approach that performs joint tracking and segmentation can improve over methods that detach segmentation from tracking.

MOTS20. We present our results on MOTS20 train and test sets in Table 9. For a fair comparison on the training set, we follow TrackFormer's 4-fold cross-validation scheme. In this setting, we observe 1.2 sMOTSA and 3.7 MOTSA improvements over TrackFormer, and 1.8 and 4.9

Table 5 Comparison of our method with state-of-the art on the **MOTChallenge test** sets

Method	On/Off	MOTA ↑	IDF1 ↑	HOTA ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓	Hz ↑
2D MOT 2015 (Leal-Taixé et al., 2015)										
JointMC (Keuper et al., 2020)	Offline	35.6	45.1	34.6	23.2	39.3	10580	28,508	457	0.6
AMIR15 (Sadeghian et al., 2017)	Online	37.6	46.0	33.2	15.8	26.8	7933	29,397	1026	1.9
STRN (Xu et al., 2019)	Online	38.1	46.6	33.5	11.5	33.4	5451	31,571	1033	13.8
DeepMOT† (Xu et al., 2020a)	Online	44.1	46.0	36.3	17.2	26.6	6085	26,917	1347	1.6
Tracktor++v2† (Bergmann et al., 2019)	Online	46.6	47.6	37.6	18.2	27.9	4624	26,896	1290	1.8
Lif_T† (Hornakova et al., 2020)	*Offline*	***52.5***	***60.0***	***46.0***	***33.8***	***25.8***	***6837***	***21,610***	*730*	*1.5*
MPNTrack† (Ours)	Offline	**51.5**	**58.6**	**45.0**	**31.2**	**25.9**	**7620**	**21,780**	**375**	6.5
MOT16 (Milan et al., 2016)										
FWT (Henschel et al., 2017)	Offline	47.8	44.3	35.7	19.1	38.2	8886	85,487	852	0.6
STRN (Xu et al., 2019)	Online	48.5	53.9	39.7	17.0	34.9	9038	84,178	747	13.5
LMP (Tang et al., 2017)	Offline	48.8	51.3	41.0	18.2	40.1	6654	86,245	481	0.5
BLSTM_MTP_O (Kim et al., 2021b)	*Online*	*48.3*	*53.5*	*39.7*	*17.0*	*38.7*	*9792*	*83707*	*735*	*21.0*
Tracktor++v2† (Bergmann et al., 2019)	Online	56.2	54.9	44.8	20.7	35.8	**2394**	76,844	617	1.8
LPC_MOT† (Dai et al., 2021)	*Offline*	*58.8*	***67.6***	*51.7*	***27.3***	*35.0*	*6167*	*68,432*	*435*	*4.3*
Lif_T† (Hornakova et al., 2020)	*Offline*	*61.3*	*64.7*	***50.8***	*27.0*	*34.0*	*4844*	*65,401*	*389*	*0.5*
TMOH† (Stadler and Beyerer, 2021)	*Online*	***63.2***	*63.5*	*50.7*	*27.0*	***31.0***	*3122*	***63,376***	*635*	*0.7*
MPNTrack† (Ours)	*Offline*	**58.6**	**61.7**	**48.9**	**27.3**	**34.0**	4949	**70,252**	**354**	6.5
MOT17 (Milan et al., 2016)										
jCC (Keuper et al., 2020)	Offline	51.2	54.5	42.5	20.9	37.0	25,937	247,822	1802	1.8
FAMNet (Chu and Ling, 2019)	Online	52.0	48.7	–	19.1	33.4	14,138	25,3616	3072	–
JBNOT (Henschel et al., 2019)	Offline	52.6	50.8	41.3	19.7	35.8	31,572	232,659	3050	5.4
Tracktor++v2† (Bergmann et al., 2019)	Online	56.3	55.1	44.8	21.1	35.3	**8866**	235,449	1987	1.8
LPC_MOT† (Dai et al., 2021)	*Offline*	*59.0*	***66.8***	***51.5***	***29.9***	*33.9*	*23,102*	*206,948*	***1122***	*4.8*
Lif_T† (Hornakova et al., 2020)	*Offline*	*60.5*	*65.6*	*51.3*	*27.0*	*33.6*	*14,966*	*206,619*	*1189*	*0.5*
CenterTrack (Zhou et al., 2020)	*Online*	*61.5*	*59.6*	*48.2*	*26.4*	*31.9*	*14,076*	***200,672***	*2583*	*17.0*
TMOH† (Stadler and Beyerer, 2021)	*Online*	***62.1***	*62.8*	*50.4*	*26.9*	***31.4***	*10,951*	*201,195*	*1897*	*0.7*
MPNTrack† (Ours)	Offline	**58.8**	**61.7**	**49.0**	**28.8**	**33.5**	17413	**213594**	**1185**	6.5
MOT20 (Dendorfer et al., 2020)										
SORT (Bewley et al., 2016)	Online	42.7	45.1	36.1	16.7	26.2	27,521	264,694	4470	57.3
Tracktor++v2† (Bergmann et al., 2019)	Online	52.6	52.7	42.1	29.4	26.7	**6930**	236,680	1548	1.8
LPC_MOT† (Dai et al., 2021)	*Offline*	*56.3*	***62.5***	***49.0***	*34.1*	*25.2*	*11726*	*213,056*	*1562*	*0.7*
TMOH† (Stadler and Beyerer, 2021)	*Online*	***60.1***	*61.2*	*48.9*	***46.7***	***15.2***	*38,043*	***165,899***	*2342*	*0.6*
MPNTrack† (Ours)	Offline	**57.6**	**59.1**	**46.8**	**38.2**	**22.5**	16953	**201384**	**1210**	6.5

Methods written in italic were published after our CVPR2020 work. Bold is used to indicate the best scores for each metric, separately for methods published before and after our CVPR2020 work. Methods with † after their name also used Tracktor-based preprocessing on their input boxes

Table 6 Comparison of our method with state-of-the-art on **KITTI-tracking test** set

Method	HOTA ↑	DetA ↑	AssA ↑	MOTA ↑	MT (%) ↑	ML(%) ↓	FP ↓	FN ↓	ID Sw. ↓
2D Methods									
NOMT (Choi, 2015)	36.3	31.9	41.6	36.5	19.2	43.6	12319	**2249**	**127**
CenterTrack (Zhou et al., 2020)	*40.4*	*44.5*	*35.4*	*53.8*	*21.3*	*34.7*	*8061*	*2201*	*425*
Quasi-Dense (Pang et al., 2021)	*41.1*	***44.8***	***38.1***	*55.6*	*31.3*	*20.3*	*8493*	***1309***	*487*
MPNTrack (Ours)	**45.3**	**43.7**	**47.3**	**46.2**	**44.0**	**10.3**	**6956**	5096	397
3D Methods									
CIWT (Osep et al., 2017)	33.9	**34.0**	34.1	42.1	14.1	**35.1**	11821	**1149**	433
JRMOT (Shenoi et al., 2020)	34.1	29.6	**39.5**	**45.3**	13.1	47.4	**10207**	1822	**179**
AB3DMOT (Weng et al., 2020)	**35.6**	33.0	38.6	38.9	**17.2**	41.2	11744	2135	259
EagerMOT (Kim et al., 2021a)	***39.4***	***40.6***	***38.7***	***49.8***	***27.5***	***24.1***	***8959***	*2161*	*496*

Methods written in italic were published after our CVPR2020 work

Table 7 Comparison of our method with state-of-the-art on the **Human in Events test** dataset

Method	MOTA ↑	IDF1 ↑	MT(%) ↑	ML (%) ↓	FP↓	FN ↓	ID Sw. ↓
DeepSORT (Wojke et al., 2017)	27.2	28.6	8.5	41.4	**5894**	426,68	2220
CenterTrack (Zhou et al., 2020)	*31.1*	*41.8*	*8.6*	*27.9*	*10014*	*35253*	*2767*
TPM (Peng et al., 2020)	*33.6*	*37.7*	*10.7*	*31.2*	*6595*	*35,395*	*4287*
GMPHD-ReId (Baisa, 2021)	*31.3*	*37.7*	***36.0***	*24.3*	*17,309*	*26,158*	*4392*
STPP (Wang et al., 2020)	*37.5*	*40.2*	*20.4*	*29.8*	*7395*	*31,638*	*4536*
SiamMOT (Shuai et al., 2021)	***53.2***	*51.7*	*26.7*	*27.5*	***2837***	*28,485*	*1308*
MPNTrack (Ours)	**48.0**	**53.3**	**33.6**	**23.8**	7756	**27,236**	**1243**

Methods written in italic were published after our CVPR2020 work.

Table 8 Comparison of our method with the baselines obtained from MOT methods on **MOTS20 train** set

Method	Val.	sMOTSA ↑	IDF1 ↑	MOTSA ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
MOTS20 Train Set - Public									
jCC (Keuper et al., 2020)	×	48.8	61.6	63.3	81	32	2514	7159	202
MHT_DAM (Kim et al., 2015)	×	51.4	62.6	65.4	82	34	2266	6877	164
FWT (Henschel et al., 2018)	×	51.5	54.0	65.4	81	33	1835	7213	249
Lif_T (Hornakova et al., 2020)	×	53.6	**73.3**	67.5	**89**	23	2026	**6615**	**92**
CenterTrack (Zhou et al., 2020)	×	53.9	58.4	67.5	80	34	1488	6975	279
Tracktor++ (Bergmann et al., 2019)	×	54.6	62.7	67.7	72	24	1036	7509	154
MPNTrackSeg (Ours)	✓	**55.4**	68.2	**67.8**	76	39	**955**	7592	102

We report our *cross-validation* scores, whereas baselines are obtained from the corresponding method's *training set* results.

MOTSA improvements over PointTrack. On the MOTS20, test set, our model establishes a new state-of-the-art by significantly improving on all metrics over the top published methods. Specifically, we outperform TrackFormer, TraDeS, and TrackR-CNN by 3.7, 7.8, and 18.2 points in sMOTSA and 5.2, 10.1, and 16.4 points in IDF1, respectively. In addition, our model achieves the highest track coverage among all methods and reduces the number of identity switches by 25% compared to the closest method.

KITTI MOTS. We compare our method against the top published methods on KITTIMOTS in Table 10. We achieve the best performance among the methods that only utilize 2D information and surpass the previous state-of-the-art, PointTrack, by 1.1 HOTA. Our approach excels in association accuracy (AssA) and MT, in which we outperform the closest method with 4 points and 20% more track coverage. In fact, our 2D approach obtains highly competitive results even when compared to the methods that make use of 3D information, such as EagerMOT and MOTSFusion. Specifically, we

Table 9 Comparison of our method with state-of-the-art on **MOTS20 train and test** sets

Method	sMOTSA ↑	IDF1 ↑	MOTSA ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
MOTS20 Train Set								
TrackR-CNN (Voigtlaender et al., 2019)	52.7	–	66.9	–	–	–	–	–
MOTSNet (Porzi et al., 2020)	56.8	–	69.4	–	–	–	–	–
PointTrack (Xu et al., 2020b)	58.1	–	70.6	–	–	–	–	–
TrackFormer (Meinhardt et al., 2021)	58.7	–	–	–	–	–	–	–
MPNTrackSeg (Ours)	**59.9**	71.9	**74.3**	126	17	1679	5017	210
MOTS20 Test Set								
TrackR-CNN (Voigtlaender et al., 2019)	40.6	42.4	55.2	127	71	1261	12,641	567
TraDeS (Wu et al., 2021)	50.8	58.7	65.5	162	60	1474	9169	492
TrackFormer (Meinhardt et al., 2021)	54.9	63.6	–	–	–	2233	**7195**	278
MPNTrackSeg (Ours)	**58.6**	**68.8**	**73.7**	**207**	**26**	**1059**	7233	**202**

Table 10 Comparison of our method with state-of-the-art on **KITTI MOTS test** set

Method	HOTA ↑	DetA ↑	AssA ↑	sMOTSA ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
2D Methods									
TrackR-CNN (Voigtlaender et al., 2019)	41.9	53.8	33.8	47.3	123	36	5355	1171	482
PointTrack (Xu et al., 2020b)	54.4	**62.3**	48.0	**61.5**	132	**25**	4341	**344**	176
MPNTrackSeg (Ours)	**55.5**	60.5	**52.0**	57.3	**152**	26	**3743**	857	**162**
3D Methods									
MOTSFusion (Luiten et al., 2020a)	54.0	60.8	49.5	58.8	128	42	4868	463	279
EagerMOT (Kim et al., 2021a)	57.7	60.3	56.2	58.1	117	37	5056	**458**	270
VIP-DeepLab (Qiao et al., 2021)	**64.3**	**70.7**	**59.5**	**68.8**	**199**	**7**	**2265**	731	**209**

obtain a significantly higher MT while also preserving 50% more tracks. Furthermore, our model produces the fewest number of identity switches among all 2D and 3D methods.

6 Discussion

Overall, we have demonstrated strong results in both box-based and mask-based tracking in a variety of datasets. Given our graph-based formulation, our method's strongest ability is data association, as it can be observed from its high IDF1 and AssA scores, together with its fast runtime (Table 5). Moreover, as shown in Tables 8, 9 and 10, our method achieves top scores with our proposed attentive message passing module on MOTS benchmarks by blending tracking and segmentation cues, demonstrating the potential of solving complementary tasks jointly within a unified framework.

We also note, however, that our method falls short in overall tracking performance when compared to some of the more recent tracking methods that either use more complex optimization schemes (Hornakova et al., 2020), or are regression-based (Stadler and Beyerer, 2021). Regarding the former, we believe that our method's significantly improved runtime offers a favorable efficiency tradeoff. Moreover, we note that LPC_MOT (Dai et al., 2021) has been inspired by our work to use a message passing neural network within a multiple-hypothesis graph formulation.

Regarding regression-based approaches, we would like to note that they offer a different performance profile. Our method, since it is graph-based, focuses entirely on data association. Regression-based methods, instead, have the ability to track an increased number of boxes, and hence can improve MOTA significantly, but fall short in terms of data association performance, as can be seen from the relatively lower IDF1 scores of CenterTrack (Zhou et al., 2020) and TMOH (Stadler and Beyerer, 2021). We believe there is a need to investigate how these two lines of work can be combined to get the benefits of both of them. We hope future work will be able to develop such integration.

7 Conclusion

We have introduced a fully differentiable framework based on message passing networks that can exploit the under-

lying graph structure of the tracking problem. Our unified architecture reasons over the entire graph and performs data association and segmentation jointly by merging appearance, geometry, and mask features. Our approach achieves state-of-the-art results in both multi-object tracking and segmentation tasks on several benchmarks. We hope that our approach will open the door for future efforts in marrying graph-based methods with deep learning approaches and exploiting synergies between tracking and segmentation.

Funding. Open Access funding enabled and organized by Projekt DEAL. This project was partially funded by the Sofja Kovalevskaja Award of the Humboldt Foundation and by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036B. The authors of this work take full responsibility for its content.

Declarations

Conflicts of interest Not applicable.

Availability of data and material. All datasets used are publicly available.

Code availability Our code is publicly available.

References

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms and applications*. Upper Saddle River, NJ, USA: Prentice Hall.

Baisa, N. L. (2021). Occlusion-robust online multiobject visual tracking using a GM-PHD filter with CNN-based re-identification. *Journal of Visual Communication and Image Representation., 80*, 103279.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., & Gulcehre C. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 .

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., & Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*.

Berclaz, J., Fleuret, F., & Fua, P. (2006). Robust people tracking with global trajectory optimization. In *IEEE conference on computer vision and pattern recognition*.

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. In *IEEE transactions on pattern analysis and machine intelligence*.

Bergmann, P., Meinhardt, T., & Leal-Taixé, L. (2019). Tracking without bells and whistles. In *International conference on computer vision*.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *IEEE international conference on image processing*.

Braso, G., & Leal-Taixe, L. (2020). Learning a neural solver for multiple object tracking. In *IEEE conference on computer vision and pattern recognition*.

Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., & van Gool, L. (2009). Robust tracking-by-detection using a detector confidence particle filter. In *International conference on computer vision*.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 .

Choi, W. (2015). Near-online multi-target tracking with aggregated local flow descriptor. In *International conference on computer vision*.

Choi, W., & Savarese, S. (2012). A unified framework for multi-target tracking and collective activity recognition. In *European conference on computer vision*.

Chu, P., & Ling, H. (2019). Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *International conference on computer vision*.

Dai, P., Weng, R., Choi, W., Zhang, C., He, Z., & Ding, W. (2021). Learning a proposal classifier for multiple object tracking. In *IEEE conference on computer vision and pattern recognition*.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixé, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. arXiv preprint arXiv:2003.09003 .

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hi-erarchical Image Database. In *IEEE conference on computer vision and pattern recognition*.

Ess, A., Leibe, B., Schindler, K., & van Gool, L. (2008). A mobile vision system for robust multi-person tracking. In *IEEE conference on computer vision and pattern recognition*.

Frossard, D., & Urtasun, R. (2018). End-to-end learning of multi-sensor 3d tracking by detection. In *IEEE International Conference on Intelligent Robots and Systems*.

Gao, J., Zhang, T., & Xu, C. (2019). Graph convolutional tracking. In *IEEE conference on computer vision and pattern recognition*.

Geiger, A., Lenz, P., & Urtasun, R. (2012a). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on computer vision and pattern recognition (cvpr)*.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE conference on computer vision and pattern recognition*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*.

Guo, M., Chou, E., Huang, D.-A., Song, S., Yeung, S., & Fei-Fei, L. (2018). Neural graph matching networks for fewshot 3d action recognition. In *European conference on computer vision*.

He, J., Huang, Z., Wang, N., & Zhang, Z. (2021). Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In *IEEE conference on computer vision and pattern recognition*.

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask r-cnn. In *International conference on computer vision*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*.

Henschel, R., Leal-Taixé, L., Cremers, D., & Rosenhahn, B. (2017). Improvements to frank-wolfe optimization for multi-detector multi-object tracking. In *IEEE conference on computer vision and pattern recognition*.

Henschel, R., Leal-Taixe, L., Cremers, D., & Rosenhahn, B. (2018). Fusion of head and full-body detectors for multi-object tracking. In *IEEE Conference on computer vision and pattern recognition workshops*.

Henschel, R., Zou, Y., & Rosenhahn, B. (2019). Multiple people tracking using body and joint detections. In *IEEE Conference on computer vision and pattern recognition workshops*.

Hornakova, A., Henschel, R., Rosenhahn, B., & Swoboda, P. (2020). Lifted disjoint paths with application in multiple object tracking. In *IEEE International conference on machine learning*.

Hornakova, A., Kaiser, T., Rolinek, M., Rosenhahn, B., Swoboda, P., Henschel, R., & equal contribution (2021). Making higher order mot scalable: An efficient approximate solver for lifted disjoint paths.

Jiang, H., Fels, S., & Little, J. (2007). A linear proramming approach for multiple object tracking. In *IEEE conference on computer vision and pattern recognition*.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Boonstra, M., & Zhang, J. (2009). Framework for performance evaluation for face, text and vehicle detection and tracking in video: data, metrics, and protocol. In *IEEE transactions on pattern analysis and machine intelligence*.

Keuper, M., Tang, S., Andres, B., Brox, T., & Schiele, B. (2020). Motion segmentation multiple object tracking by correlation co-clustering. In *IEEE transactions on pattern analysis and machine intelligence*.

Kim, A., Ošep, A., & Leal-Taixé, L. (2021). Eagermot: 3d multi-object tracking via sensor fusion. In *International conference on intelligent robots and systems*.

Kim, C., Fuxin, L., Alotaibi, M., & Rehg, J. M. (2021). Discriminative appearance modeling with multitrack pooling for real-time multi-object tracking. In *IEEE conference on computer vision and pattern recognition*.

Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *International conference on computer vision*.

Kim, C., Li, F., & Rehg, J. M. (2018). Multi-object tracking with neural gating using bilinear lstm. In *European conference on computer vision*.

Kim, S., Kwak, S., Feyereisl, J., & Han, B. (2013). Online multi-target tracking by large margin structured learning. In *Asian Conference on Computer Vision*.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. In *International conference on machine learning*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 .

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing*.

Leal-Taixé, L., Canton-Ferrer, C., & Schindler, K. (2016). Learning by tracking: Siamese cnn for robust target association. In *IEEE conference on computer vision and pattern recognition workshops*.

Leal-Taixé, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., & Savarese, S. (2014). Learning an imagebased motion context for multiple people tracking. In *European conference on computer vision*.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., & Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. arXiv preprint arXiv:1504.01942 .

Leal-Taixé, L., Pons-Moll, G., & Rosenhahn, B. (2011). Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *International conference on computer vision workshops*

Leal-Taixé, L., Pons-Moll, G., & Rosenhahn, B. (2012). Branch-and-price global optimization for multiview multi-target tracking. In *IEEE conference on computer vision and pattern recognition*.

Li, J., Gao, X., & Jiang, T. (2020). Graph networks for multiple object tracking. In *IEEE winter conference on applications of computer vision* (pp. 708–717). https://doi.org/10.1109/WACV45572.2020.9093347

Li, W., Zhao, R., Xiao, T., & Wang, X. (2014). Deepreid: Deep filter pairing neural network for person re-identification. In *IEEE conference on computer vision and pattern recognition*.

Li, Z., Chen, Q., & Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in neural information processing systems*.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *IEEE conference on computer vision and pattern recognition*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*.

Lin, W., Liu, H., Liu, S., Li, Y., Qian, R., Wang, T., Xu, N., Xiong, H., Qi, G.J., & Sebe, N. (2020). Human in events: A large-scale benchmark for human-centric video analysis in complex events. arXiv preprint arXiv:2005.04490 .

Liu, Q., Chu, Q., Liu, B., & Yu, N. (2020). Gsm: Graph similarity model for multi-object tracking. In C. Bessiere (Ed.), *Proceedings of the twentyninth international joint conference on artificial intelligence, IJCAI-20* (pp. 530-536). International Joint Conferences on Artificial Intelligence Organization. (Main track) 7

Luiten, J., Fischer, T., & Leibe, B. (2020). Track to reconstruct and reconstruct to track. In *IEEE Robotics and Automation Letters*.

Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., & Leibe, B. (2020). Hota: A higher order metric for evaluating multi-object tracking. In *International conference on computer vision*.

Ma, L., Tang, S., Blakc, M., & van Gool, L. (2019). Customized multi-person tracker. In *Asian conference on computer vision*.

Meinhardt, T., Kirillov, A., Leal-Taixe, L., & Feichtenhofer, C. (2021). Trackformer: Multiobject tracking with transformers. arXiv preprint arXiv:2101.02702 .

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 .

Milan, A., Leal-Taixé, L., Schindler, K., & Reid, I. (2015). Joint tracking and segmentation of multiple targets. In *IEEE conference on computer vision and pattern recognition*.

Narasimhan, M., Lazebnik, S., & Schwing, A. G. (2018). Out of the box: Reasoning with graph convolution nets for factual visual question answering. arXiv preprint arXiv:1811.00538 .

Osep, A., Mehner, W., Mathias, M., & Leibe, B. (2017). Combined image- and world-space tracking in traffic scenes. In *International conference on intelligent robots and systems*

Pang, J., Qiu, L., Li, X., Chen, H., Li, Q., Darrell, T., & Yu, F. (2021). Quasi-dense similarity learning for multiple object tracking. In *IEEE conference on computer vision and pattern recognition*.

Pellegrini, S., Ess, A., Schindler, K., & van Gool, L. (2009). You'll never walk alone: modeling social behavior for multi-target tracking. In *International conference on computer vision*.

Peng, J., Wang, C., Wan, F., Wu, Y., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., & Fu, Y. (2020). Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *European conference on computer vision*.

Peng, J., Wang, T., Lin, W., Wang, J., See, J., Wen, S., & Ding, E. (2020). Tpm: Multiple object tracking with tracklet-plane matching. *Pattern Recognition, 107*, 107480.

Pirsiavash, H., Ramanan, D., & Fowlkes, C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE conference on computer vision and pattern recognition*.

Porzi, L., Hofinger, M., Ruiz, I., Serrat, J., Bulo, S. R., & Kontschieder, P. (2020). Learning multi-object tracking and segmentation from automatic annotations. In *IEEE conference on computer vision and pattern recognition*

Qiao, S., Zhu, Y., Adam, H., Yuille, A., & Chen, L.-C. (2021). Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation. In *IEEE conference on computer vision and pattern recognition*.

Redmon, J., & Farhadi, A. (2017). Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*.

Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. (2016). Deepmatching: Hierarchical deformable dense matching. In *International conference on computer vision*.

Ristani, E., Solera, F., Zou, R. S., Cucchiara, R., & Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision workshop*.

Ristani, E., & Tommasi, C. (2018). Features for multi-target multi-camera tracking and reidentification. In *IEEE conference on computer vision and pattern recognition*.

Sadeghian, A., Alahi, A., & Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *International conference on computer vision*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. In *Transactions on neural networks*.

Schulter, S., Vernaza, P., Choi, W., & Chandraker, M. (2017). Deep network flow for multi-object tracking. In *IEEE conference on computer vision and pattern recognition*.

Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., & Sun, J. (2018). Crowdhuman: A benchmark for detecting human in a crowd. arXiv preprint arXiv:1805.00123 .

Shenoi, A., Patel, M., Gwak, J., Goebel, P., Sadeghian, A., Rezatofighi, H., & Savarese, S. (2020). Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset. In *International conference on intelligent robots and systems*.

Shuai, B., Berneshawi, A., Li, X., Modolo, D., & Tighe, J. (2021). Siammot: Siamese multi-object tracking. In *IEEE conference on computer vision and pattern recognition*.

Stadler, D., & Beyerer, J. (2021). Improving multiple pedestrian tracking by track management and occlusion handling. In*IEEE conference on computer vision and pattern recognition*.

Tang, S., Andriluka, M., Andres, B., & Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *IEEE conference on computer vision and pattern recognition*

Tokmakov, P., Li, J., Burgard, W., & Gaidon, A. (2021). Learning to track with object permanence. In *International conference on computer vision* (p. 1084).

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., & Leibe, B. (2019). Mots: Multi-object tracking and segmentation. In *IEEE conference on computer vision and pattern recognition*.

Wang, S., & Fowlkes, C. (2015). Learning optimal parameters for multi-target tracking. In *The British Machine Vision Conference (BMVC)*

Wang, T., Chen, K., Lin, W., See, J., Zhang, Z., Xu, Q., & Jia, X. (2020). Spatio-temporal point process for multiple object tracking. *IEEE transactions on neural networks and learning systems*.

Wang, Z., Zheng, L., Liu, Y., & Wang, S. (2020). Towards real-time multi-object tracking. In *European conference on computer vision*

Weng, X., Wang, J., Held, D., & Kitani, K. (2020). 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE international conference on intelligent robots and systems*.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and real-time tracking with a deep associametric. In *IEEE international conference on image processing*.

Wu, J., Cao, J., Song, L., Wang, Y., Yang, M., & Yuan, J. (2021). Track to detect and segment: An online multi-object tracker. In *IEEE conference on computer vision and pattern recognition*.

Wu, Z., Kunz, T., & Betke, M. (2011). Efficient track linking methods for track graphs using networkflow and set-cover techniques. In *IEEE conference on computer vision and pattern recognition*

Xie, S., Girshick, R., Dollar, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *IEEE conference on computer vision and pattern recognition*

Xu, J., Cao, Y., Zhang, Z., & Hu, H. (2019). Spatialtemporal relation networks for multi-object tracking. In *International conference on computer*

Xu, Y., Osep, A., Ban, Y., Horaud, R., Leal-Taixe, L., & Alameda-Pineda, X. (2020). How to train your deep multi-object tracker. In *IEEE conference on computer vision and pattern recognition*

Xu, Z., Zhang, W., Tan, X., Yang, W., Huang, H., Wen, S., & Huang, L. (2020). Segment as points for efficient online multi-object tracking and segmentation. In *European conference on computer vision*.

Yang, F., Choi, W., & Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE conference on computer vision and pattern recognition*.

Yu, F., Wang, D., Shelhamer, E., & Darrell, T. (2018). Deep layer aggregation. In *IEEE Conference on computer vision and pattern recognition*.

Yu, Q., Medioni, G., & Cohen, I. (2007). Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *IEEE conference on computer vision and pattern recognition*.

Zamir, A., Dehghan, A., & Shah, M. (2012). Gmcptracker: Global multi-object tracking using generalized minimum clique graphs. In *European conference on computer vision*.

Zhang, L., Li, Y., & Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *IEEE conference on computer vision and pattern recognition*.

Zhang, Y., Sheng, H., Wu, Y., Wang, S., Lyu, W., Ke, W., & Xiong, Z. (2020). Long-term tracking with deep tracklet association. *IEEE Transactions on Image Processing, 29*, 6694–6706.

Zhang, Y., Wang, C., Wang, X., Zeng, W., & Liu, W. (2021). Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision, 129*, 3069–3087.

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., & Tian, Q. (2015). Scalable person re-identification: A benchmark. In *International conference on computer vision*.

Zhou, X., Koltun, V., & Krähenbühl, P. (2020). Tracking objects as points. In *European Conference on Computer Vision*.