PolarMOT: How Far Can Geometric Relations Take Us in 3D Multi-Object Tracking?

Aleksandr Kim, Guillem Brasó, Aljoša Ošep, Laura Leal-Taixé
Technical University of Munich, Germany
{aleksandr.kim, guillem.braso, aljosa.osep, leal.taixe}@tum.de

Abstract

Most (3D) multi-object tracking methods rely on object-level information, e.g. appearance, for data association. By contrast, we investigate how far we can get by considering only geometric relationships and interactions between objects over time. We represent each tracking sequence as a multiplex graph where 3D object detections are nodes, and spatial and temporal pairwise relations among them are encoded via two types of edges. This structure allows our graph neural network to consider all types of interactions and distinguish temporal, contextual and motion cues to obtain final scene interpretation by posing tracking as edge classification. The model outputs classification results after multiple rounds of neural message passing, during which it is able to reason about long-term object trajectories, influences and motion based solely on initial pairwise relationships. To enable our method for online (streaming) scenarios, we introduce a technique to continuously evolve our graph over long tracking sequences to achieve good performance while maintaining sparsity with linear complexity for the number of edges. We establish a new state-of-the-art on the nuScenes dataset.

1 Introduction

Intelligent agents such as autonomous vehicles need to understand dynamic objects in their surroundings to safely navigate the world. 3D multi-object tracking (MOT) is, therefore, an essential component of autonomous intelligent systems.

State-of-the-art methods leverage the representational power of neural networks to learn appearance models [41, 48] or regress velocity vectors [46] as cues for data association. While powerful, such methods need to be trained for the specific environments in which they are deployed. Methods such as [40, 7] rely on motion as the key cue for association and can thus generalize across different environments. However, performance-wise they lag behind data-driven methods as they treat individual objects in isolation and do not consider their interactions.

In this work, we investigate *how far we can get* by learning to track objects given *only geometric cues* in the form relative pose differences between 3D bounding boxes *without* relying on any appearance information. This approach is not coupled to any specific object detector, sensor modality, or region-specific appearance models. As a result, it generalizes well across environments and geographic locations, as we experimentally demonstrate. By contrast to prior work [40] that rely on individual object motion as the main association cue, our method can learn how objects move as a group, taking into account their interactions to better adapt to dynamic environments and crowded scenarios.

As graphs are a natural representation for such long-term agent interactions, we represent our scene as a *sparse*, *multiplex* graph, which is then processed by a graph neural network (Fig. 2). We encode 3D detection as nodes, while edges represent their *spatial* and *temporal* relations and denote possible associations and influences. After several message passing steps in this graph, our neural network

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

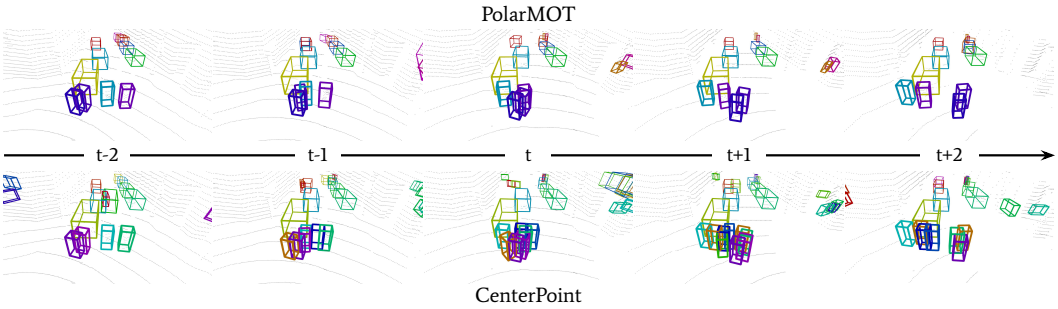


Figure 1: *PolarMOT* tracks objects in 3D (offline and online) using a graph neural network that learns to associate 3D bounding boxes over time solely based on their relative geometric features and spatio-temporal relationships. Considering such object interactions improves *PolarMOT*'s ability to handle scenarios with high occlusions (pedestrian crossing) compared to others, *e.g.* CenterPoint.

outputs binary classifications for all *temporal* edges. All nodes connected with a positive edge form a track and thus get assigned a consistent track ID [5].

One of our main insights is that encoding pairwise geometric relations, *i.e.*, edge features, in *localized polar* instead of global Cartesian space is the key to good generalization and robustness and enables our model to effectively learn to track by *solely* relying on geometric cues. The pair-specific nature of the features makes them invariant to global transforms, so features between different detections of a consistently moving object will be *stable*, regardless of the route geometry or point of reference. Such polar representation also naturally encodes non-holonomic motion prior, *i.e.*, a motion that is constrained by heading direction and is well parametrized by heading angle and velocity.

We evaluate our method, *PolarMOT*, on KITTI [10] and nuScenes datasets [6]. Our ablations reveal that our proposed graph structure and edge feature parametrization are pivotal for our final model achieving 66.4 average AMOTA, setting a new state-of-the-art on nuScenes MOT dataset among lidar-based methods. More importantly, we show that our learned tracker, which relies only on geometry leads to strong generalization across different geographic regions (Boston, Singapore, Karlsruhe) and datasets (nuScenes, KITTI), without fine-tuning. We are not suggesting to *not use* appearance cues, but point out that we can get very far with a minimalistic, graph-based tracking, solely relying on geometric cues and inferring object properties implicitly from interactions.

To **summarize**, (i) we propose a minimalistic, graph-based 3D tracker that relies only on geometric cues and, without any bells and whistles or image/lidar input, establish new state-of-the-art on the nuScenes dataset; (ii) we suggest a graph-based representation of the scene that encodes temporal and spatial relations via a localized polar representation. This is not only to achieve state-of-the-art performance on benchmarks but, more importantly, is the key to strong generalization; (iii) as efficiency and online operation are crucial for robotic/AV scenarios, we construct a sparse graph

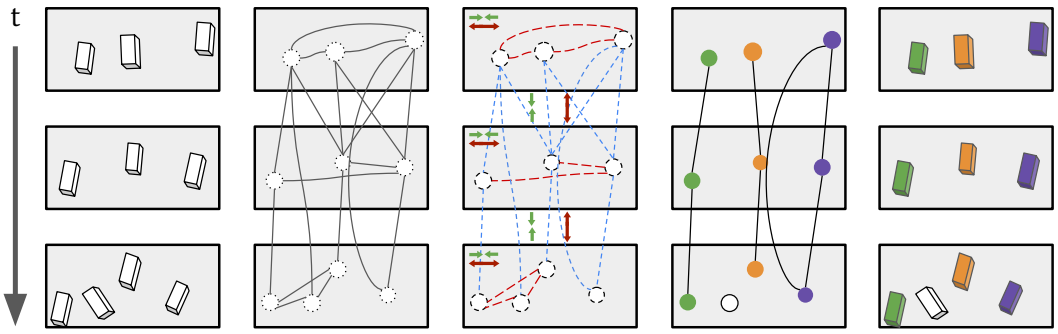
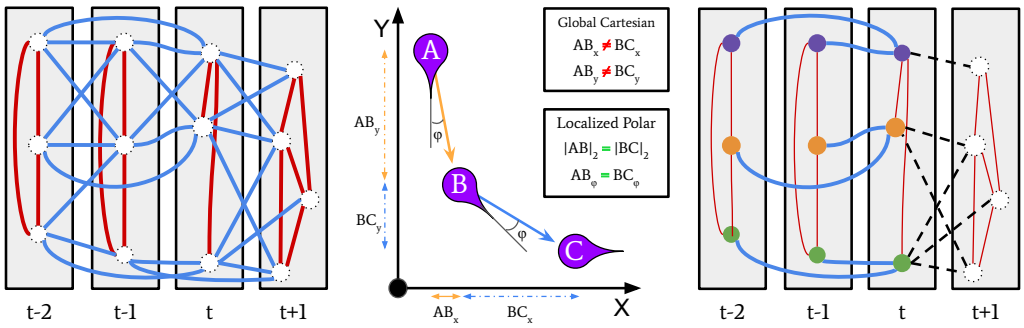


Figure 2: Given a set of 3D bounding box detections in a sequence, *PolarMOT* constructs a graph encoding objects as nodes and their geometric relations as *spatial* and *temporal* edges. After implicitly deriving node features and refining edge features via message passing with wider spatial and temporal context, we classify edges to obtain track predictions.



(a) Full (sparse) graph. (b) Relational features. (c) Online construction.

Figure 3: The key contributions of our work: multiplex input graph with *inter-frame (temporal)* and *intra-frame (spatial)* edges, relative geometric features in localized polar coordinates and continuously evolving online graph construction.

and only establish links that are feasible based on maximal possible velocity and show how such sparse graphs can be constructed in an online fashion. We hope our code and models, available at `polarmot.github.io`, will run on friendly future robots!

2 Related Work

This section reviews relevant related work in 2D and 3D multi-object tracking (MOT) based on the well-established *tracking-by-detection* paradigm.

In the era of deep learning, the community has been focusing on learning strong appearance models [18, 39, 35], future target locations [2, 44] or predicting offset vectors as cues for association [49]. Recently, we have witnessed a resurgence in graph-based approaches for tracking. MPNTrack [5] encodes 2D object detections as nodes while edges represent possible associations among them. Then, a message passing [11] graph neural network (GNN) [12] is used to update node/edge representation with temporal context. Just as [5], we (i) encode detections as nodes, while edges represent hypothetical associations, and we (ii) learn to classify edges to obtain the final set of tracks. Different to [5], we tackle 3D MOT using *only* geometric cues. Our model encodes both, temporal and spatial relations and, importantly, encodes them via *localized polar coordinates*, that encode non-holonomic motion prior. Moreover, beyond *offline* use-case, we show how to construct *sparse* graphs in *online* fashion with streaming input, as needed in mobile robotics.

3D MOT. Early methods for 3D detection and tracking [8, 28] model vehicles as cuboids or rely on bottom-up point cloud segmentation [36, 24, 13] and track objects using (extended) Kalman filter. Thanks to developments in deep point-based representation learning [30, 31, 37, 50], we nowadays have strong backbones, as needed for 3D object detection [29, 33, 50, 45, 17, 46], tracking [9, 40, 7] and segmentation [1]. AB3DMOT [40] shows that well-localized lidar-based 3D object detections work very well in conjunction with a simple Kalman filter based tracking framework [4]. Association can be performed based on 3D bounding box overlap or centroid-based Mahalanobis distance [7] to increase robustness to lower frame rates. State-of-the-art CenterPoint [46] learns to detect objects as points and regresses velocity vectors needed for the association. Similar to ours, OGR3MOT[47] also tackles 3D MOT using message passing networks based framework, proposed by [5]. However, it represents detections *and* tracks as two distinct types of nodes, thus effectively maintaining two subgraphs. *Different* to that, our *PolarMOT* implicitly derives node embeddings from relational (edge) features parametrized via proposed local polar coordinates. A recent body of work proposes to fuse image and lidar data [14, 48, 41]. GNN3DMOT [41] utilizes GNNs to learn appearance and motion features for data association jointly. This approach relies on Hungarian algorithm to perform data association based on the learned features. *Different* to that, we *only* use 3D geometric cues and perform association directly via edge classification in end-to-end manner.

3 PolarMOT

In this section, we provide a high-level overview of our *PolarMOT*, followed by a detailed discussion of our key ideas and components.

3.1 Method Overview

PolarMOT encodes the overall geometric configuration of detected objects and the relative pose changes between them as primary cues for tracking.

Sparse graph-based scene representation. We use a graph to represent object detections as nodes, and their geometric relations as edges. We encode both *temporal* and *spatial* relations among objects in two levels of our *sparse multiplex* graph (Fig. 3a). The *temporal* level connects nodes across different frames with inter-frame edges, while the *spatial* level connects nodes of the same frame with intra-frame edges. For sparsity, we *only* link nodes that are mutually reachable based on the maximal velocity of their semantic class.

Localized relational polar encoding We encode geometric relations among nodes via a localized polar-based representation (Fig. 3b). Our parametrization is not based on a shared global coordinate frame but is specific to each pair's local frame. This makes our parametrization *invariant* to the reference world frame, and induces a *non-holonomic* (directional) motion prior.

Learning representation via message passing. We follow a message passing procedure [11] to iteratively update our features via edge/node updates (Fig. 2, *center*). Node features are not extracted from the input detections, instead, they are learned *implicitly* through mutual object interactions. For more details on the procedure, please refer to the appendix.

Edge classification. To obtain object tracks, we classify our *temporal* edges using a multi-layer perceptron based on final edge representations (Fig. 2, *right*).

Online graph construction. For *online* tracking, we continuously evolve our input graph after each frame, while maintaining both sparsity and high connectivity (Fig. 3c). For online processing, we connect past track predictions directly to the most recent detections, thus allowing our network to infer historical context effectively via message passing on the graph topology.

3.2 Message Passing on a Sparse Multiplex Graph

Representation. Following [5], we represent each individual object detection as a *node* of the graph with *edges* encoding relations between objects. More precisely, we model *temporal* and *spatial* relations between objects (graph nodes) by encoding our 3D detections via a *sparse*, undirected *multiplex* graph with two distinct levels / edge types (Fig. 3a). In the *first level* we model temporal connections via undirected *inter-frame* edges between nodes across frames [5], which describe potential motion of objects. Each edge connects two distinct detections across frames. The edge features denote the likelihood of an object moving from pose/node A to pose/node B in the elapsed time. In the *second level*, we introduce undirected *intra-frame* edges (*i.e.*, links between nearby detections from the same frame) to model spatial context. These edges act as pathways to exchange frame-specific context and are meant to express the mutual influence of moving targets' motion patterns. This should intuitively help with difficult and ambiguous cases that arise in crowded scenarios. Both inter- and intra-frame edges connect the same set of nodes, but convey semantically different information, effectively turning our graph into a multiplex network with two distinct levels. *Intra-frame* edges are excluded from edge classification.

Table 1: Results of state-of-the-art methods for 3D multi-object tracking on the NuScenes test set.

Method name	Input modality	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
Ours	3D	**242**	**70.2**	**66.4**	**85.3**	**80.6**	34.9	70.8	**65.6**	**67.3**	60.2
OGR3MOT [47]	3D	288	69.2	65.6	81.6	78.7	**38.0**	**71.1**	64.0	**67.1**	59.0
CenterPoint [46]	3D	684	68.0	65.0	81.8	78.0	33.1	71.5	58.7	69.3	**62.5**

Sparse graph construction. To handle occlusions, we connect nodes across any number of frames. Doing this naively would potentially result in a prohibitively dense graph. To ensure graph sparsity, we take full advantage of the 3D domain, and rely on physical constraints to establish only *relevant*, physically-plausible spatial or temporal relations. For *inter-frame edges*, if the physical distance between two detections is greater than the maximal velocity of the detected class, we consider it impossible for them to belong to the same object and do not form an edge. For *intra-frame edges*, we allow distance up to twice the maximal velocity to connect objects that could collide in the next frame if moving towards each other and, therefore, influence each other's movement.

Tracking via edge classification. After graph construction and message passing, we classify *inter-frame* edges, which encode *temporal* relations. Positive classification implies the same object identity for both detections (*i.e.*, a consistent track ID). Any edge connecting nodes of the same track should be labeled as positive, regardless of the time difference between the connected nodes.

3.3 Localized Relational Polar Encoding

3D sensors do not provide comprehensive appearance information (compared to cameras) and instead provide accurate distance measurements to objects. Therefore, for 3D MOT, we fully focus on efficiently representing relative poses via edge features and learn node features implicitly. This leads to the question of how to efficiently represent geometric relations between objects? We start by representing each 3D detection as an oriented point with two planar coordinates x, y and orientation ϕ around s vertical axis z. We ignore elevation, tilt, and spatial dimensions of objects as their values are similar among objects of the same class.

Global Cartesian coordinates. Traditionally, spatial geometric relations between objects are parametrized in Cartesian coordinates relative to some reference frame, encoding an edge $h_{i,j}^{(0)}$ connecting nodes o_i and o_j as:

$$h_{i,j}^{(0)} = \Delta(o_i, o_j) = \begin{bmatrix} \Delta x \\ \Delta y \\ \cdots \end{bmatrix} = \begin{bmatrix} o_i^x - o_j^x \\ o_i^y - o_j^y \\ \cdots \end{bmatrix}. \tag{1}$$

As shown in Fig. 3b, for *identical* pose changes $\overrightarrow{AB}$ and $\overrightarrow{BC}$, global Cartesian features are *not identical*, as they depend on the orientation of the reference frame. Thus, with such representation, *identical motion non-intuitively leads to different relational features.*

Localized polar coordinates.

We propose a different representation, which depends (i) only on the relevant pair of objects, o_i and o_j, and (ii) is more suitable for encoding directional non-holonomic motion. Our parametrization (Fig. 3b) expresses differences in poses A and B through a velocity vector, expressed in *polar coordinates*, where the center of the first detection A is the *pole* (*i.e.*, the origin of the polar coordinate frame), and its heading direction (downward) is the *polar axis*. This vector includes two components: *velocity* $|AB|$ (*i.e.*, distance between objects by detection time difference), and *polar angle* φ, *i.e.*, the angle between $\overrightarrow{AB}$ and the polar axis of A (downward vector). We also include differences in object orientation (o^ϕ) and detection time difference (o^t):

$$h_{i,j}^{(0)} = \Delta(o_i, o_j) = \begin{bmatrix} v \\ \varphi_{i,j} \\ \Delta\phi \\ \Delta t \end{bmatrix} = \begin{bmatrix} \frac{\|o_i - o_j\|_2}{\Delta t} \\ \angle\left(\vec{o}_i^{\,head}, \vec{o_j} - \vec{o_i}\right) \\ o_i^\phi - o_j^\phi \\ o_i^t - o_j^t \end{bmatrix}. \tag{2}$$

We remove the dependency on an arbitrary reference frame by computing each feature relative to individual localized frames. *Now, identical motion between $\overrightarrow{AB}$ and $\overrightarrow{BC}$ leads to identical features.*

Table 2: Online tracking on the nuScenes validation set [6]

Method name	Input modality	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
Ours	3D	**439**	**72.46**	**67.27**	81.26	78.79	49.38	82.76	67.19	45.80	65.70
CenterPoint	3D	562	70.62	65.91	84.23	77.29	43.70	80.16	59.16	51.47	65.39

Table 3: Ablation on parametrization of geometric relations on nuScenes validation set.

Localized polar	Normalized by time	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
✓	✓	430	**62.12**	**57.96**	**85.16**	**80.80**	**44.02**	**80.68**	45.83	**5.41**	**63.83**
✓	✗	652	55.07	52.17	81.51	80.14	06.66	79.53	53.82	0	63.51
✗	✓	1321	41.06	40.41	78.77	78.80	0	67.62	0	0	57.66

Table 4: Ablation for intra-frame edges on the nuScenes validation set

Intra-frame connections	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motorcycle	trailer	truck
✓	213	**75.14**	**71.14**	**85.83**	**81.70**	**54.10**	**87.36**	**72.32**	**48.67**	68.03
✗	**198**	72.74	70.09	85.44	80.51	52.88	86.78	69.87	46.61	**68.54**

Polar coordinates explicitly encode the change in heading angle (φ). This intuitively encodes a *smooth, non-holonomic* motion prior. For example, polar features for trajectory $\overrightarrow{KL}$ and an (improbable) trajectory $\overrightarrow{MN}$ significantly differ, while their Cartesian coordinates do not. These key characteristics allow *PolarMOT* to generalize well even when trained with a small number of labeled samples (Sec. 4.4), especially important for rarely observed object classes.

3.4 Online Graph Construction

In this section, we propose an online graph construction approach that makes our method applicable to *online* applications. We maintain a *single* input graph for the whole sequence, which we continuously evolve with each incoming frame. This construction is identical to the *offline* setting, as discussed in Sec. 3.2. In each frame, edges are classified based on past-frame information *only*. There are multiple ways of maintaining a single input graph over a sequence of frames.

MPNTrack++: dense. First approach is a simple extension of offline GNN methods [5, 19], where previous track associations do not influence graph structure. At each frame, new nodes and edges are added to a continuously growing *dense* graph. *Such model is unaware of past track estimates.*

Prune inactive: consecutive. Past track estimates are a valuable cue to resolve ambiguous associations. To this end, [47] propose to simply prune all past (negative) edges. This option encodes track history and maintains sparsity and due to frame-by-frame processing, preserved track edges always connect temporally *consecutive* nodes in each track. However, over time, temporal distance from the current frame to early nodes is increasing, making them unreachable within a limited number of message passing iterations. *This approach thus has a limited temporal receptive field.*

Ours: prune + skip. We propose a solution that maintains graph sparsity and a global temporal receptive field. After each frame, we remove all *past negative* edges *and* ensure we have, for each track, an edge between *each* most-recent track node and *all* past nodes from its track, see Fig. 3c. Moreover, new nodes at each frame are *only* connected to the most-recent track node for each existing track. By continuously evolving the graph in this manner we maintain sparsity, provide previous tracking decisions to the model through the input graph topology (this makes our model autoregressive) and keep all nodes reachable during message passing: any two connected nodes have at most two edges between them. *This allows our model to learn historical context directly via message passing to make well-informed edge classification, as needed for long-term tracking.*

4 Experimental Evaluation

This section outlines our evaluation setting: datasets, metrics and input detections (Sec. 4.1). Next, in Sec. 4.2, we discuss our *offline* tracking results on the nuScenes test set and *online* and *offline* performance on the validation set, compared to the current state-of-the-art. Then, we justify our design decisions and discuss the benefits of our contributions via thorough ablation studies (Sec. 4.3). Finally, we demonstrate the generalization capabilities of *PolarMOT* by successfully applying it to different datasets and locations without fine-tuning (Sec. 4.4).

Table 5: Sparse graph construction: the impact of reducing/increasing the maximal velocity threshold on online tracking (nuScenes validation set)

Max edge distance	IDs ↓ total	Recall ↑ average	AMOTP ↓ average	AMOTA ↑ car	class-specific AMOTA ↑ ped	bicycle	bus	motorcycle	trailer	truck	
0.5x	1123	65.14	0.718	58.61	76.90	46.66	46.89	77.35	62.73	40.09	59.63
1.0x	**439**	**72.46**	**0.595**	**67.27**	**81.26**	**78.79**	**49.38**	82.76	**67.19**	**45.80**	**65.70**
2.0x	467	69.74	0.642	65.42	**81.28**	72.15	46.84	**83.45**	63.57	44.94	**65.70**

Table 6: Online graph construction analysis (nuScenes validation set)

Track connectivity	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	pedestrian	bicycle	bus	motorcycle	trailer	truck
Ours	**439**	**72.46**	**67.27**	**81.26**	**78.79**	49.38	**82.76**	**67.19**	**45.80**	**65.70**
Consecutive	485	69.90	66.03	81.04	77.37	48.84	82.45	66.58	40.66	65.23
Dense	1024	68.58	61.39	71.08	75.20	**49.70**	79.78	52.03	41.15	60.79

4.1 Evaluation setting

Datasets. We evaluate our method on nuScenes [6] and KITTI [10] tracking datasets. NuScenes was recorded in four locations across two cities: Boston, USA, and Singapore. It contains 150 scenes, recorded with a 32-beam lidar sensor, and provides two scans per second (2Hz). KITTI tracking dataset was recorded in Karlsruhe, Germany, using a 64-beam lidar sensor at a 10Hz frame rate. Unless otherwise specified, we follow the official train/validation/test splits.

Evaluation metric. On the nuScenes dataset, we follow the official evaluation protocol that reports per-class and average AMOTA [40], which averages the CLEAR-MOT [3] MOTA metric across different recall thresholds. On KITTI, we also report sAMOTA [40] and HOTA [23] metrics, which allows us to analyze detection and association errors separately. We also report the number of identity switches (IDs) at the best performing recall. In the supplementary, we provide extended tables and evaluations that we omitted for brevity.

Object detections. On nuScenes, for a fair comparison with CenterPoint [46], we use their provided 3D detections (without the estimated velocity vectors). On KITTI, we use PointGNN [34] detections provided by EagerMOT [14].

4.2 Benchmark results

We compare our method to state-of-the-art 3D MOT on the official nuScenes benchmark. In Tab. 1 (*top*) we compare methods that rely *only* on the 3D input. As can be seen, our model ranks highest overall (66.4 avg. AMOTA compared to 2nd best 65.6). Our method only lags behind methods that additionally rely on rich visual signal (AlphaTrack and EagerMOT). While our focus was on effectively leveraging geometric relations, this hints that our approach could potentially further benefit from sensor fusion, which we leave for future work.

Online tracking. To confirm that our approach successfully handles online scenarios, we evaluate it on the nuScenes validation set with streaming inputs. In Tab. 2 we compare our *offline* and *online* versions, and compare both to *online* state-of-the-art CenterPoint [46] using the same 3D detections.

Not surprisingly, our *offline* model achieves the top performance at 71.14 avg. AMOTA. When switching to *online* inference, we achieve 67.27 avg. AMOTA (-3.87), outperforming CenterPoint by $+1.36$ avg AMOTA (from 65.91). As expected, there is a noticeable difference between our *offline* and *online* variants, as the online version is effectively exposed to only half of the context available to its offline counterpart, which observes both past and future detections.

4.3 Model ablation

Edge parametrization. To measure the impact of our proposed representation of geometric relations, we train three models with different relative feature parametrizations. The main advantage of our *proposed polar* representation is in the inductive bias that helps the model better understand long

trajectories and non-holonomic motion. We conduct this experiment in the low data regime using the official nuScenes mini-split (1% of training data) and evaluate them on the (non-overlapping) validation set (150 seq.) and report results in Tab. 3.

Intra-frame connections. *Does spatial context matter?* Tab. 4 shows that adding spatial *intra-frame* edges improves avg. AMOTA by $+1.05$ and significantly improves recall ($+2.4\%$) at a marginal increase in IDs ($+15$). For better contextual awareness, we aggregate messages from temporal and spatial edges separately. An ablation study of this technique is in the supplementary material.

Sparse graph construction. *What is the impact of our sparse graph construction on tracking performance?* In Tab. 5 we compare results obtained by models using half (0.5x) and double (2.0x) of the measured maximal velocity values to constrain edges with the ones from our default model (1x). Reducing maximum allowed velocity (0.5x) removes valuable edges that hypothesize valid associations, while permitting higher values (2x) introduces irrelevant edges. In both cases, we observe a significant performance drop across all metrics (recall, precision, AMOTA), due to message passing becoming less complete and noisier and edge classification more challenging. This confirms that our data-driven physics-based sparse graph construction is the optimal approach.

Evolving online graph connectivity. To validate our proposed approach to online graph connectivity, we present ablation experiments that confirm the advantage of our technique over alternatives discussed in Sec.3.4. As shown in Tab. 6, our experimental results align with our expectations based on the theoretical properties of each option. *Dense* (MPNTrack++) linking (61.39 avg. AMOTA) completely ignores past trajectories and produces the lowest results. Next, *consecutive* (prune inactive) chaining introduces a significant improvement with 66.03 avg. AMOTA. Finally, our proposed *prune + skip* connectivity leads to a significant improvement over both alternatives with 67.27 avg. AMOTA.

4.4 Generalization study

In this section, we show how *PolarMOT* generalizes across geographic locations and datasets. To this end, we (i) use the nuScenes dataset, recorded in Singapore and Boston, to train the model on one city and evaluate on the other and (ii) evaluate our model, trained on nuScenes, on KITTI data without any fine-tuning.

Cross-city generalization: Boston $\leftrightarrow$ Singapore. In this cross-city evaluation we compare our method to CenterPoint (we use detections from the corresponding re-trained CenterPoint model). As shown in Tab. 10, our method generalizes significantly better: $+3.41$ avg. AMOTA on Boston (trained) $\rightarrow$ Singapore (evaluated) and $+3.33$ avg. AMOTA on Singapore $\rightarrow$ Boston. In both cases we also observe a significant improvement in both recall and number of ID-switches. This confirms that our method generalizes very well across regions, which we believe is largely due to our feature parametrization (Sec. 3.3).

Cross-dataset generalization: nuScenes $\rightarrow$ KITTI. We report our results on the unofficial KITTI 3D MOT benchmark [40] in Tab. 8, and on the official KITTI 2D MOT benchmark in Tab. 9. We *only* evaluate our nuScenes-trained model on KITTI dataset, which we consider an ultimate generalization experiment, as KITTI dataset was recorded using a different sensor under a different frame rate in a different geographical location (Karlsruhe, Germany). For 2D MOT evaluation we simply project our estimated 3D tracks to the image plane. We note that entries are not directly comparable, as different methods use different input detections. However, we observe our method is on-par with EagerMOT [15], which uses lidar and cameras, and the same set of 3D detections. *PolarMOT* is

Table 7: CenterPoint (CP) [46] and our method when trained on training data from one city, and evaluated on the validation data from another

Train city → eval city	Tracking model	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
Boston → Singapore	Ours	**145**	**64.48**	**63.12**	**82.26**	**72.81**	**31.49**	77.70	**43.17**	0	**71.28**
	CP	306	61.02	59.71	79.26	67.47	20.52	**78.86**	41.13	0	71.04
Singapore → Boston	Ours	**104**	52.30	**50.28**	**78.60**	**82.59**	**36.70**	**71.22**	**28.71**	11.31	42.83
	CP	314	**53.32**	47.06	77.01	76.18	34.86	71.07	13.54	**13.18**	**43.55**

Table 8: Unofficial KITTI 3D MOT validation set benchmark [40]. Our model was trained **only** on nuScenes dataset

Method name	3D input	2D input	IDs ↓ car	IDs ↓ ped	sAMOTA ↑ car	sAMOTA ↑ ped	MOTA ↑ car	MOTA ↑ ped	Recall ↑ car	Recall ↑ ped
Ours *online*	✓	✗	31	9	94.32	**94.08**	93.93	**93.48**	94.54	**93.66**
PC-TCNN [43]	✓	✗	**1**	–	**95.44**	–	unpub	–	unpub	–
EagerMOT [14]	✓	✓	**2**	36	94.94	92.95	**96.61**	93.14	**96.92**	**93.61**
GNN3DMOT [41]	✓	✓	10	–	93.68	–	84.70	–	unpub	–
AB3DMOT [40]	✓	✗	**0**	**1**	91.78	73.18	83.35	66.98	92.17	72.82

Table 9: KITTI 2D MOT test set benchmark. Our model was trained **only** on nuScenes dataset

Method name	3D input	2D input	IDs ↓ car	IDs ↓ ped	HOTA ↑ car	HOTA ↑ ped	AssA ↑ car	AssA ↑ ped	AssRe ↑ car	AssRe ↑ ped	MOTA ↑ car	MOTA ↑ ped
Ours *online*	✓	✗	462	270	75.16	43.59	76.95	48.12	80.00	51.95	85.08	46.98
PC-TCNN [43]	✓	✗	**37**	–	**80.90**	–	**84.13**	–	**87.46**	–	**91.70**	–
PermaTrack [38]	✗	✓	258	403	78.03	48.63	78.41	45.61	81.14	49.63	91.33	65.98
PC3T [42]	✓	✗	225	–	77.80	–	81.59	–	84.77	– 5	88.81	–
Mono_3D_KF [32]	✗	✓	162	267	75.47	42.87	77.63	46.31	80.23	52.86	88.48	45.44
EagerMOT [14]	✓	✓	239	496	74.39	39.38	74.16	38.72	76.24	40.98	87.82	49.82
SRK_ODESA [25]	✗	✓	380	511	68.51	**50.87**	63.08	48.78	65.89	53.45	87.79	**68.04**
3D-TLSR [27]	✗	✓	–	**175**	–	46.34	–	**51.32**	–	**54.45**	–	53.58
MPNTrack [5]	✗	✓	–	397	–	45.26	–	47.28	–	52.18	–	46.23

top-performer on the 3D benchmark and among top-4 on the 2D benchmark, performing consistently well on *car* and *pedestrian* classes.

5 Conclusion

We presented *PolarMOT* for 3D multi-object tracking that solely relies on 3D bounding boxes as input without appearance/shape information. Our key contribution is a GNN that encodes spatial and temporal geometric relations via localized polar coordinates. This parametrization enables us to effectively learn to understand long-range temporal and spatial context via message passing and, solely from object interactions, learn a scene representation suitable for tracking via edge classification. We also propose an online graph construction technique to apply *PolarMOT* to streaming data. Our method establishes a new state-of-the-art on the nuScenes dataset among methods that do not rely on image data and, more importantly, generalizes well across geographic regions and datasets.

References

[1] Aygün, M., Ošep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., Leal-Taixé, L.: 4d panoptic lidar segmentation (2021)

[2] Bergmann, P., Meinhardt, T., Leal-Taixé, L.: Tracking without bells and whistles (2019)

[3] Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: The clear mot metrics pp. 1–10 (2008)

[4] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: ICIP (2016)

[5] Braso, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking (2020)

[6] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving (2020)

[7] Chiu, H.k., Prioletti, A., Li, J., Bohg, J.: Probabilistic 3d multi-object tracking for autonomous driving (2021)

[8] Dellaert, F., Thorpe, C.: Robust car tracking using kalman filtering and bayesian templates. In: Conference on Intelligent Transportation Systems (1997)

[9] Frossard, D., Urtasun, R.: End-to-end learning of multi-sensor 3d tracking by detection (2018)

[10] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite (2012)

[11] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International conference on machine learning. pp. 1263–1272. PMLR (2017)

[12] Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: IJCNN (2005)

[13] Held, D., Levinson, J., Thrun, S., Savarese, S.: Combining 3d shape, color, and motion for robust anytime tracking (2014)

[14] Kim, A., Ošep, A., Leal-Taixé, L.: Eagermot: 3d multi-object tracking via sensor fusion (2021)

[15] Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Video panoptic segmentation (2020)

[16] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2015)

[17] Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds (2019)

[18] Leal-Taixé, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: Siamese cnn for robust target association. CVPR Workshops (2016)

[19] Li, J., Gao, X., Jiang, T.: Graph networks for multiple object tracking (2020)

[20] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection (2017)

[21] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond (April 2020)

[22] Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts (2017)

[23] Luiten, J., Ošep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., Leibe, B.: Hota: A higher order metric for evaluating multi-object tracking (2020)

[24] Moosmann, F., Stiller, C.: Joint self-localization and tracking of generic objects in 3d range data (2013)

[25] Mykheievskyi, D., Borysenko, D., Porokhonskyy, V.: Learning local feature descriptors for multiple object tracking. In: ACCV (2020)

[26] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML (2010)

[27] Nguyen, U., Heipke, C.: 3d pedestrian tracking using local structure constraints. ISPRS Journal of Photogrammetry and Remote Sensing **166**, 347–358 (2020)

[28] Petrovskaya, A., Thrun, S.: Model based vehicle detection and tracking for autonomous urban driving **26**, 123–139 (2009)

[29] Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data (2017)

[30] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation (2017)

[31] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space (2017)

[32] Reich, A., Wuensche, H.J.: Monocular 3d multi-object tracking with an ekf approach for long-term stable tracks. In: FUSION (2021)

[33] Shi, S., Wang, X., Li, H.: PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud (2019)

[34] Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud (2020)

[35] Son, J., Baek, M., Cho, M., Han, B.: Multi-object tracking with quadruplet convolutional neural networks (2017)

[36] Teichman, A., Levinson, J., Thrun, S.: Towards 3D object recognition via classification of arbitrary object tracks (2011)

[37] Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds (2019)

[38] Tokmakov, P., Li, J., Burgard, W., Gaidon, A.: Learning to track with object permanence. In: ICCV (2021)

[39] Voigtlaender, P., Krause, M., Ošep, A., Luiten, J., Sekar, B., Geiger, A., Leibe, B.: MOTS: Multi-object tracking and segmentation (2019)

[40] Weng, X., Wang, J., Held, D., Kitani, K.: 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics (2020)

[41] Weng, X., Wang, Y., Man, Y., Kitani, K.: Gnn3dmot: Graph neural network for 3d multi-object tracking with multi-feature learning (2020)

[42] Wu, H., Han, W., Wen, C., Li, X., Wang, C.: 3d multi-object tracking in point clouds based on prediction confidence-guided data association. IEEE TITS (2021)

[43] Wu, H., Li, Q., Wen, C., Li, X., Fan, X., Wang, C.: Tracklet proposal network for multi-object tracking on point clouds. In: IJCAI (2021)

[44] Xu, Y., Ošep, A., Ban, Y., Horaud, R., Leal-Taixé, L., Alameda-Pineda, X.: How to train your deep multi-object tracker (2020)

[45] Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)

[46] Yin, T., Zhou, X., Krähenbühl, P.: Center-based 3d object detection and tracking (2021)

[47] Zaech, J.N., Liniger, A., Dai, D., Danelljan, M., Van Gool, L.: Learnable online graph representations for 3d multi-object tracking (2022)

[48] Zeng, Y., Ma, C., Zhu, M., Fan, Z., Yang, X.: Cross-modal 3d object detection and tracking for auto-driving (2021)

[49] Zhou, X., Koltun, V., Krähenbühl, P.: Tracking objects as points (2020)

[50] Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection (2018)

A Appendix

A.1 Message Passing Networks for Multi-Object Tracking

Our work is inspired by MPNTrack [5], an image-based MOT method that we summarize for completeness. MPNTrack models detections as graph nodes and represents possible associations via edges. After propagating features via neural message passing [11], edges are classified as active/inactive.

MPNTrack processes a clip of frames with detected objects, and outputs classifications of links between them. For each clip, input is a set of object detections $\mathcal{O} = \{o_i\}_{i=1}^{n}$, represented via an (appearance) embedding vector, 2D position and timestamp. Then, a graph $G = (V, E)$ where $V = \mathcal{O}$ and $E \subset \mathcal{O} \times \mathcal{O}$. MPNTrack encodes only inter-frame edges and heuristically prunes them for sparsity. Each node $o_i \in V$ and edge $e_{ij} \in E$ (connecting o_i, o_j) have corresponding initial embeddings $h_i^{(0)}$ and $h_{(i,j)}^{(0)}$ obtained from appearance and position cues. These embeddings are propagated across the graph via neural message passing for a fixed number of iterations L to obtain updated edge features $h_{(i,j)}^{(1,\dots,L)}$.

More specifically, edge embeddings are updated at each message passing step based on embeddings from the previous step of the edge itself and its neighboring nodes. Nodes are then updated based on previous embeddings of the node and neighboring edges, which are aggregated separately for forward and backward directions in time. After L message passing updates, edges are classified based on their embeddings from all steps, $h_{(i,j)}^{(1,\dots,L)}$, where positive classification implies that two connected detections are part of the same track (identity).

Message passing. Following the general message passing framework, our model performs a fixed number of alternating *edge* and *node* updates to obtain enhanced node/edge representation, as needed for edge classification.

Edge update: At every message passing iteration $l = 1, \dots, L$, we first update edge embeddings by learning to fuse the edge and connected node embeddings from the previous message passing iteration. More precisely, we update edge embedding $h_{(i,j)}$ between nodes o_i and o_j as follows:

$$h_{(i,j)}^{(l)} = \text{MLP}_{\text{edge}}\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}\right]\right). \tag{3}$$

During each update, the edge embedding from the previous message passing iteration $l-1$, *i.e.*, $h_{(i,j)}^{(l-1)}$, is concatenated with embeddings $(h_i^{(l-1)}, h_j^{(l-1)})$ of the connected nodes. The result is then fused by MLP_{edge}, a Multi-Layer Perceptron (MLP), which produces the fused edge features $h_{(i,j)}^{(l)}$ (Eq. 3). This edge embedding update is identical for all edge types and directions.

Node update: To update nodes, first, we construct messages for each edge $h_{(i,j)}^{(l)}$ via learned fusion of *its updated edge* and *neighboring nodes'* features:

$$m_{(i,j)}^{(l)} = \begin{cases} \text{MLP}_{\text{past}}\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_j^{(l-1)}]\right) & \text{if } t_j < t_i, \\ \text{MLP}_{\text{pres}}\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_{(j)}^{(l-1)}]\right) & \text{if } t_i = t_j, \\ \text{MLP}_{\text{fut}}\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_j^{(l-1)}]\right) & \text{if } t_j > t_i. \end{cases} \tag{4}$$

Via *inter-frame edges* we produce two messages for each temporal direction: we produce *past* features via MLP_{past} and *future* features via MLP_{fut} (only in offline tracking settings). At the same time,

Table 10: CenterPoint (CP) [46] and our method when trained on training data from one city, and evaluated on the validation data from another

Train city → eval city	Tracking model	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
Boston → Singapore	Ours	**145**	**64.48**	**63.12**	**82.26**	**72.81**	**31.49**	77.70	**43.17**	0	**71.28**
	CP	306	61.02	59.71	79.26	67.47	20.52	**78.86**	41.13	0	71.04
Singapore → Boston	Ours	**104**	52.30	**50.28**	**78.60**	**82.59**	**36.70**	**71.22**	**28.71**	11.31	42.83
	CP	314	**53.32**	47.06	77.01	76.18	34.86	71.07	13.54	**13.18**	**43.55**

we produce *present* features via MLP_{pres} for each *intra-frame*, spatial, edge. Using separate MLPs allows our model to learn different embeddings for each type of relationship.

Node aggregation: Then, each node $h_i^{(l)}$ *aggregates* incoming messages using an element-wise max operator, separately for past, present and future edges to maintain contextual awareness. The resulting aggregated vectors are concatenated and fused via MLP_{node}:

$$h_i^{(l)} = \text{MLP}_{\text{node}}([\max_{t_j<t_i} m_{(i,j)}^{(l)}, \max_{t_i=t_j} m_{(i,j)}^{(l)}, \max_{t_j>t_i} m_{(i,j)}^{(l)}]). \tag{5}$$

As we do not use any object-specific information (*e.g.* appearance), we rely on the model to implicitly learn node features from their interactions.

Initialization: We initialize edge embeddings with relational features (as detailed in Sec. 3.3) processed by MLP_{edge_init} (Eq. 6). These are then directly aggregated to produce *initial node embeddings* via MLP_{node_init} (Eq. 7):

$$\text{(initial edge)} \qquad h_{(i,j)}^{(1)} = \text{MLP}_{\text{edge\_init}}(h_{(i,j)}^{(0)}), \tag{6}$$

$$\text{(initial node)} \qquad h_i^{(1)} = \text{MLP}_{\text{node\_init}}([\max_{t_j<t_i} h_{(i,j)}^{(1)}, \max_{t_i=t_j} h_{(i,j)}^{(1)}, \max_{t_j>t_i} h_{(i,j)}^{(1)}]). \tag{7}$$

A.2 Implementation details

Network structure. In *PolarMOT*, we use only fully connected (FC) layers with the leaky ReLU [26] nonlinearity, and the dimensionality of the edge and node features is 16 and 32 respectively. Our MLP_{node} and MLP_{node_init} consist of 3 FC layers. Remaining MLPs consist of 2 layers with 70k parameters in total.

Training and augmentation. We train *PolarMOT* only on keyframes from the nuScenes [6] dataset, which we augment with noise dropout to mimic real detection performance. To mimic occlusions and false negatives, we randomly drop full frames, individual bounding boxes, and edges. To mimic false positives, we randomly add bounding boxes. We also perturb boxes by adding Gaussian noise to 3D positions and orientat sion. We train *PolarMOT* with focal loss [20], which is well suited for our imbalanced binary classification case. We use Radam optimizer [21, 16] using cosine annealing with warm restarts [22] for 180 epochs with batch size of 64.