Department of Informatics
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Spatio-Temporal Graph Neural Networks for Multiple Object Tracking

Räumlich-Zeitliche Graphische Neuronale Netze für Tracking mehrerer Objekte

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Emeç Ercelik, M.Sc.

Author Maximilian Listl

Date June 15, 2022 in Garching

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, June 15, 2022

(Maximilian Listl)

Abstract

Multiple object tracking (MOT) is an important aspect for autonomous robotic applications, such as autonomous driving. Current research regarding MOT is mainly based on 2D object detections. However, there is a recent shift towards 3D MOT based on 3D object detections. Nevertheless, most of the state-of-the-art 3D MOT methods still rely on a combination of Kalman filters and association metrics, to match consecutive 3D object detections from different time frames.
Instead we propose to represent the tracking problem as a spatio-temporal graph. Each node represents a detection. Each node is connect to nodes from the same time frame over spatial edges. The temporal edges allow the representation of detection sequences.
In addition, we propose to use a graph neural network(GNN) to associate nodes. We show that our GNN benefits from the spatio-temporal structure. By including the spatial edges, the GNN can encode the spatial context for each node. It improves the on average tracking performance by 1.398% percentage points, in terms of average multiple object tracking accuracy (AMOTA), when using ground truth annotations from the Nuscenes dataset as detections.Additionally, we evaluate our MOT method on publicly available CenterPoint detections and compare it to state of the art MOT methods. There we show that our method suffers from its inability to filter out false positive detections.

Zusammenfassung

Das Tracking von mehreren Objekten ist einer der zentralen Aspekte für autonome Robotikanwendungen, wie zum Beispiel beim autonomen Fahren. Aktueller Stand der Technik sind hier Tracking-Methoden mit 2D basierten Objekterkennungsalgorithmen. Allerdings lässt sich in der Forschung hinsichtlich des Trackens mehrerer Objekte eine zunehmende Verschiebung hin zu 3D basierenden Algorithmen beobachten. Dabei wird meist auf eine Kombination von Kalman Filter und Assoziierungsmetriken gesetzt. Dies ermöglicht es, die zu unterschiedlichen Zeitpunkten erkannten Objekte in Beziehung zueinander zu setzen.
Alternativ schlagen wir vor das Tracking-Problem als räumlich-zeitlichen Graphen darzustellen. Dabei wird jedes erkannte Objekt durch einen Knoten repräsentiert. Jeder Knoten wird mit anderen Knoten des gleichen Zeitpunkts über räumliche Kanten verbunden. Zeitliche Kanten ermöglichen es, die Sequenz der Objekterkennung darzustellen.
Eine weitere Komponente unseres alternativen Ansatzes ist es, ein graphisches neuronales Netz (GNN) zu benutzen, um die Knoten zu assoziieren. Wir zeigen, dass unser GNN von der räumlich-zeitlichen Struktur des Graphen profitiert. Durch die Aufnahme von räumlichen Kanten, kann das GNN den räumlichen Kontext für jeden Knoten bestimmen. Im Durchschnitt verbessern wir unsere durchschnittliche Tracking-Genauigkeit um 1.398% Prozentpunkte, wenn wir die Ground Truth Objekterkennungen des Nuscenes Datensatzes tracken. Zusät-

zlich verproben wir unseren Tracking-Algorithmus mit öffentlich zugänglichen Objekterkennungen des CenterPoint-Algorithmus und vergleichen unsere Ergebnisse mit den Ergebnissen anderer Tracking-Algorithmen des Stands der Technik. Dort zeigen wir auf, dass unsere Methode unter der fehlenden Fähigkeit falsch positive Objekterkennungen zu filtern leidet.

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous driving is one of the most prominent robotic applications. It is highly researched under different aspects. One of these aspects is the robotic perception. The perception of the surrounding environment of a robot is essential for autonomous agents interacting with other agents in a dynamic environment.
The perception is composed of two main tasks, the 3D object detection and the 3D multiple object tracking (MOT). Firstly, the 3D object detection, which provides the 3D position of an object in a 3D world frame. The 3D object detection can possibly yield also the orientation, the class and its certainty of an object detection. Secondly, MOT involves the associations of given object detections, to compute the 3D trajectory of each object in the surrounding environment.
Due to MOT, an autonomous car has better scene understanding, which allows it to make better decisions. Currently, MOT has been widely researched for 2D images provided by RGB-cameras. This is because cameras were among the cheapest sensors compared to LIDAR and RADAR. However, in recent years there has been a push in research to perform 3D object detection[YZK21] and 3D MOT directly on LIDAR point clouds. Most of the current 3D MOT methods use Kalman filters [Wen+20a], [Chi+20], [PLW], to predict the future position of an object and a some association metric to match detections from different time frames. There are some MOT methods that use neural networks to either learn to fuse 2D and 3D features from images and LIDAR point clouds [Wen+20b] for computing a better association metric, or the neural network tries to directly associate detections given the raw features[Zae+22]. However, current 3D MOT methods only regard the temporal dependencies between object detections [Zae+22]. Therefore, they disregard valuable information given by the spatial relations between object detections from the same time frame. A group of cars that move simultaneously in the same direction will present a consistent distance to each other over time, which can be used as additional information feature. As a result the spatial context between object detections has not been explored yet for 3D MOT. This opens up a chance to be exploited.
We propose to build a spatio-temporal graph from object detections from different time frames, to better represent the spatial and temporal relationships between object detections. In addition, we propose to use a graph neural network, that uses the previously build spatio-temporal graph, to associate detections on the graph.

1.2 Outline

This thesis has two main tasks. Firstly, to build a spatio-temporal graph out of 3D object detection in a coherent way. Secondly, to train and design a graph neural network which matches the detections from different time frames, to infer the trajectory of each object instance. This thesis is divided into five chapters. The first chapter presents our motivations and introduces our idea. The second chapter presents related works, regarding graph neural networks, MOT with graphs and the current state-of-the-art 3D MOT methods. Afterwards, we explain our method in detail in chapter three. In chapter four we present our experiments and explain certain implementation details. In the last chapter, we conclude ours ideas and interpret our results seen in chapter four.

Chapter 2

Related Work

2.1 Graph Neural Networks

Nowadays, convolutional neural networks(CNN) [KSH12] and recurrent neural networks (RNN) [HS97] are popularly used to analyse images [KSH12] or text [Sin+17]. These data structures reside in the euclidean domain, which allows them to perform certain computations more efficiently. In the case of images and CNNs, the fixed sized filter is allows efficient encoding of visual features.
However, the emergence of non-euclidean data, e.g. point clouds [SR20], graphs of human poses [Zen+], graphs of social networks [Mon+], graphs of molecule structure [Gil+17], has increased the demand for a unified way to handle non-euclidean data [Wu+21]. Therefore, there is an increased demand for the use and development of graph neural networks (GNN). [Bro+17] tries to summarise these new emerging techniques in deep learning for non-euclidean data under the name of geometric deep learning.
The basis for Graph Neural Networks (GNN) has been continuously developed by researchers. Among those are works by [SS97],[GMS05],[Sca+09], [GM10], who were one of the first to lay down the foundations for GNNs.
Several works have tried to further develop and extend of learning on graphs by introducing convolutional variants, called graph convolutional networks (GCN) [Bru+; DBV16; TM17].
Recently, some works have adopted a more generalized framework for learning on graphs, called neural message passing network or message passing network (MPN), as seen in [Gil+17; BL20]. Furthermore, [Bat+] tries to generalize the framework even further to a graph network.
In general GNNs are very versatile, by having a variety of use cases. [SWL19] uses a GNN to compute point features that are used later on for 3D object detection in the downstream task. [BL20] uses graphs to associate 2D object detections in the image-domain between different frames. [Gil+17] uses a MPN to predict properties of molecules. Moreover, GNNs can also be used to detect fake news in social networks [Mon+]. [Wu+21] proposes a new taxonomy for GNNs, to categorize the current advancements of geometric deep learning into four categories:

- Recurrent GNNs
- Convolutional GNNs
- Graph Autoencoders
- Spatio-Temporal GNNs (STGNN)

Since most real world problems depend on time as well as on space, recent works try to encompass both domains into one graph and subsequently one GNN. [BCY22] and [LYS18] use

STGNNs for traffic prediction. Furthermore, [Gao+21] use STGNNs to predict the evolution of pandemics. Moreover, STGNNs are popular in human pose estimation as seen in [Sof+], [Cai+19] and [Zen+] due to the existence of a given apriori known graph that tries to represent the human joints. Lastly, spatio-temporal approaches also have been used recently for video instance segmentation [Wan+21b].
Our method also proposes to use a STGNN to solve the 3D multiple object tracking problem.

2.2 Graph-based Multi-Object Tracking

One typical way to perform tracking is by modelling the data association step with a graph, where each node represents either object detections or tracklets and each edge symbolises the possible linkage among them. However, there are multiple ways to take advantage of the graph description. Some works use a graph to formulate multiple object tracking (MOT) as max-flow problem, or equivalently the minimum cost problem [BL20],[LFS16],[Ber+11]. Others use graphs to perform data association over graph matching[He+21]. [Zae+22] uses graphs to jointly represent detection and tracklet instances in a unified way. While most approaches use GNNs as optimization step for optimizing the similarity value between a pair of detection, [Wen+20a], [Jia+], [LGJ20] uses the two GNNs to compute directly the appearance and motion similarities separately. [Dai+21] uses a GNN to evaluate the quality score of its tracking proposals.

2.2.1 Affinity Matrix-based Methods

A graph can be represented by an affinity matrix, which is a structured way of representing the aforementioned similarity values between pairs of detections, as seen in [Wen+20a], [SAS17], [LGJ20], [Jia+]. In this case the affinity matrix describes the weights of a weighted bipartite graph. This can also be formulated as an assignment problem [SSW05], which in turn is often solved by the hungarian algorithm [Kuh55], as seen in [Wen+20a], [Wen+20b], [SAS17], [LGJ20], [Jia+].

2.2.2 Maximum Flow Formulation

As mentioned before the MOT-problem can also be formulated as maximum flow problem, or minimum cost problem. This can be solved by a linear programming solver such as the simplex algorithm [Dan63] as seen in [LFS16]. However, [BL20] introduces a neural solver as opposed to the linear programming solver. This neural solver consists of a message passing network (MPN) that directly learns to associate detections by edge classification. In addition, a novel time-aware node and edge updating step is introduced to encourage the conservation of flows. This allows the MPN to consider interactions between detections in past and future time-frames.

2.2.3 Unified Graph Representations

[Zae+22] uses a graph to combine detections and tracklet-instances inside of a single representation. The detections are represented by detection nodes. In parallel tracking nodes are generated for each timeframe if new tracks are initialized. Detection nodes of different timeframes are connected by detection nodes, while the tracking nodes are connected to detection

nodes with tracking edges. This heterogeneous graph allows [Zae+22] to train a GNN on three different edge and node classification tasks, which aim to find a time-consistent and robust solution. Firstly, it classifies the detection edges to associate detections over time. Then it classifies the tracking edges to assign detections to a unique track. Lastly, the detection nodes are classified to detect false positive detections.

2.2.4 Trajectory Proposal Generation

[Dai+21] proposes to firstly generate multiple trajectory proposals given multiple frames and their detections. Thereafter, [Dai+21] uses a graph convolutional network to evaluate the quality score of each proposed trajectory. This "iterative graph clustering method reduces the computational cost while maintaining the quality of the generated proposals." [Dai+21, p. 1] To ensure that each detection is not assigned to multiple tracks, [Dai+21] uses adopted a simple de-overlapping strategy.

2.2.5 Graph Matching Methods

[He+21] presents a novel learnable graph matching method to address association issues in MOT. Similar to [Wen+20b], this work uses a graph for detection association. However, it considers intra-detection and intra-tracklet relationships by firstly building a detection graph out of the detections appearance features and a tracklet graph, by averaging each tracks past appearance features over time. It formulates the association problem as a general graph matching between the detection graph and the tracklet graph. These intra-detection and intra-tracklet relationships are in spirit similar to this work's spatial-connections between detections.

2.3 3D Multi-Object Tracking

Nowadays, MOT is dominated by the "tracking by detection"-paradigm, in 2D MOT [LFS16; SAS17; Voi+19; BML19; Zha+19] as well as 3D MOT [Wen+20b; Wen+20a; YZK21; KOL21; Wan+21a; Zae+22]. This is the result of resent advances in object detection in 2D [Ren+15; RF18; ZWK] and 3D object detections [YZK21; SR20; SWL19; Zhu+; Lan+19; Che+15]. In addition, there has also been in improvements in point representation learning [Qi+17a; Qi+17b; Wan+].

2.3.1 Segmentation-based Methods

However, earlier methods [MS13], [TLT11] would firstly use segmentation methods, such as the local convexity criterion [MPS09], to generate 3D object proposals. Then they associate the proposals and finally assign them to tracks.

2.3.2 Kalman Filter-based Methods

[Wen+20a] proposes a 3D Kalman filter to predict the future pose of tracked objects followed by an association step using the 3D Intersection over Union (IoU). This simple yet effective

method proposed to be a baseline for future 3D MOT methods. [Chi+20] replaces the IoU metric with the Mahalanobis distance and outperforms [Wen+20a] on the Average Multi-Object Tracking Accuracy (AMOTA) metric. However, both, the IoU-based and the distance metric-based, association metrics have their own failure cases, as mentioned by [PLW]. IoU-based methods fail to match predicted tracklets to new detections that are close in distance but not overlapping. Due to a lack of discrimination on orientations, Distance-based methods are prone to associate the predicted tracklets to false positive object detections [Pan+]. To support this [Pan+] illustrates the case where a false positive detection has a smaller distance to the tracklet than the a true positive detection, even though the false positive detection has a different orientation than the tracklet. To handle both of these failure cases [PLW] proposes the Generalized 3D IoU (GIoU) as association metric. The GIoU fuses the IoU with the volume of a enclosing convex hull of the Union of the associated bounding boxes. [Wan+21a] also uses IoU and GIoU for data association. However, [Wan+21a] proposes to never terminate the tracks. Instead [Wan+21a] continues to predict the trajectory of tracklets, even if they are not matched to any new detections from the current frame. As a result, [Wan+21a] shows that a significant amount of identity switches are caused by premature tracklet termination.

2.3.3 Velocity-based Methods

[YZK21] proposes an object detector, called CenterPoint, which also estimates a 2D velocity vector for each detection. As a result, [YZK21] uses the velocity vector to project new detections back to the previous time frame. The association is done by matching tracked objects to the projected detections with the smallest distance.

2.3.4 Learning-based Methods

As mentioned before [Wen+20b] predicts an affinity matrix to match detections to tracklets. However, it uses a GNN firstly to learn to fuse 2D and 3D appearance and motion features. After the fusion they build a graph, taking each detection and tracklet feature as node. The GNN's feature aggregation method lets the features of detections and tracklets interact with each other. The iterative aggregation leads to the refinement of the affinity matrix.
As mentioned in 2.2.3 [Zae+22] learns to perform data association directly on its unified graph representation of the detections and tracks as separate entities.

2.3.5 Fusion-based Methods

Some methods join 3D and 2D features to leverage on the domain specific advantage. Image detections are more robust for far away objects, which is more challenging for current 3D object detectors [KOL21]. 3D detections already contain 3D pose information, which makes the need for projection from the 2D image plane to 3D space obsolete. As a result, [KOL21] takes advantage of the higher detection robustness from 2D object detections to further improve the 3D tracking. [KOL21] firstly fuses existing 3D and 2d tracklets to create a fused instance. The association is performed in two stages. Firstly, they associate the 3D detections to existing 3D tracklets in a greedy fashion, using the scaled distance. All unmatched 3D tracklets are send to the the second stage. At the second stage, the fused instances of the unmatched 3D tracklets are matched to new 2D detections. This allows the method to recover from short term occlusions.

Recently [Bai+22] has proposed to use transformer-architectures to fuse 2D with 3D features. Their work introduces a soft-association mechanism that gives the 2D detection head a bias towards locations of high interest, based on the 3D features. It also uses a velocity-based approach similar to CenterPoint [YZK21]. However, it currently outperforms most methods on the Nuscenes [Cae+20] tracking challenge.

2.3.6 Tracking-Score-based Methods

[BSZ21] introduces the the idea of confidence-based MOT methods as opposed to the count-based methods which count up to a specific number of time steps before terminating an unmatched tracklet. In contrast, [BSZ21] proposes to use a tracklet score to manage the initialization and termination of tracklets. A score update module updates each tracklet's score by the detection score if it was matched to a new detection. Additionally, the score decreases continually over time, to simulate the lack of associations to new detections. If the tracklet score decreases bellow a certain threshold, it is terminated. By applying the score updating module on top of state-of-the-art 3D MOT methods, [BSZ21] reaches state-of-the-art performance on the Nuscenes [Cae+20] tracking challenge. It either uses a combination of CenterPoint [YZK21] as 3D object detector and PointTracker from [ZKK20] as tracking method or a combination of CenterPoint with a Kalman filter for association.

Chapter 3

Method

3.1 Problem Definition

3.1.1 Multiple Object Tracking represented with Graphs

In the "tracking-by-detection"-paradigm multiple object tracking (MOT) is the problem of associating a given set of object detections $\mathcal{O} = \{o_1, ..., o_n\}$, where n is total number of objects for all time frames, over time. This allows us to follow the trajectory of each unique object instance over time.
In our cases we want to associate 3D object detections from past time frames with future time frames and give the detection pair a unique tracking ID w_i, also known as instance ID. Each 3D object detection o_i is composed of a 3D pose which is defined by the 3D translation $x_i \in \mathbb{R}^3$ and a 3D rotation matrix $R_i \in \mathbb{SO}(3)$ relative to the global world frame, the 3D bounding box dimensions $b_i \in \mathbb{R}^3$, object detection score $s_i \in [0,1]$, a timestamp $\tau_i \in \mathbb{R}$ and the corresponding class-id $c_i \in [0, C-1]$, where C is the total number of unique classes available. This results in a tuple:

$$o_i = (x_i, R_i, b_i, s_i, \tau_i, c_i) \tag{3.1}$$

As a result, we can describe tracks or tracklets as a set of time-ordered object detections $T_i = \{o_{i_1}, ..., o_{i_{n_i}}\}$ where n_i is the number of detections which define the i-th tracklet [BL20].
The aim of MOT methods is to find a set of tracklets $\mathcal{T}_* = \{T_1, \ldots, T_m\}$, that explain our detections $\mathcal{O}$ in a coherent way.
Similar to [BL20], we want to model this problem with an undirected graph $G = (V, E)$, $V := \{1, \ldots, n\}$, $E \subset V \times V$ and each node $i \in V$ is a virtual representation for each observation $o_i \in \mathcal{O}$. Note, that in the computational implementation, a undirected graph is represented by having for each edge connection, two directed edges in opposite directions. A directed edge is defined by a source node i and a target node j as a tuple (i, j). Therefore, the flow convention is $i \longrightarrow j$. As a result, for a connection between nodes i and j we define a set of two edge instances (i, j) and (j, i).
The set of edges E that we construct between nodes, represent a possible link to each other in terms of tacking. Therefore, usually the edges are only built in a way such that only nodes from different time frames are connected. We call these types of edges "temporal edges". However, unlike [BL20], we also allow the construction between nodes from the same time frame, so called "spatial edges". Therefore, the set of all edges E consists of the set of all temporal edges E_{temp} and the set of all spatial edges $E_{spatial}$.

$$E = \{E_{temp}, E_{spatial}\} \tag{3.2}$$

The MOT problem as a graph can be viewed as grouping nodes into disconnected components [BL20]. As a result, we can represent each trajectory $T_i = \left\{o_{i_1}, ..., o_{i_{n_i}}\right\}$ as a group of nodes $\left\{i_1, \ldots, i_{n_i}\right\}$

3.1.2 Network Flow Formulation

To describe that two nodes belong to the same object instance w_i and therefore to the same trajectory T_i, we need to introduce a binary value $y_{(i,j)}$ for every edge $(i, j) \in E$. We adopt the convention presented by [BL20], if two nodes i and j belong to the same trajectory then $y_{(i,j)} = 1$, other wise it is 0. We describe a edge to be active if the binary value $y_{(i,j)}$ is equal to 1, otherwise they are inactive.

$$y_{(i,j)} = \begin{cases} 1, \exists T_k \in \mathcal{T}_* s.t. (i, j) \in T_k \\ 0, \text{otherwise} \end{cases} \tag{3.3}$$

Since nodes from the same time frame can never belong to the same tracklet, all of our spatial edges $(i, j) \in E_{spatial}$ are assigned a binary value $y_{(i,j)}$ equal to 0.
If we only take our temporal edges E_{temp} into account, then we can see that the binary values $y_{(i,j)}$ represent the flow labels, as seen in the maximum flow problem in [LFS16] and [BL20]. Later on we will use the set $y = \left\{y_{(i,j)}\right\}_{(i,j)\in E}$ as edge labels for the training of our graph neural network.
We can also leverage the use of the flow labels by introducing a simplified version of the flow conservation constraints, adopted from [BL20]. These flow conservation constraints allow us to impose linear constraints onto the assignment of inferred flow labels. This is the case when we want to infer a possible solution for our MOT problem based on edge predictions $\hat{y}$ from our model. The maximum flow problem, employs flow conservation constraints to ensure that a node cannot belong to more than one trajectory. We use the simplified version of the flow conservation constraints from [BL20]. For each node $i \in V$, the following holds:

$$\sum_{(j,i)\in E s.t. t_i > t_j} y_{(j,i)} \leq 1 \tag{3.4}$$

$$\sum_{(i,k)\in E s.t. t_i < t_k} y_{(i,k)} \leq 1 \tag{3.5}$$

However, this representation of the flow constrains assumes that none of the detections are false positives. For a more general formulation of the flow conservation constraints we refer the reader to [AMO93] and the supplemental material from [BL20].

3.1.3 Multiple Object Tracking with Spatio-Temporal Graphs

As opposed to other state of the art MOT methods, we see the tracking problem as a spatio-temporal problem. This means that we try to exploit the fact that 3D object detections contain the 3D pose to build a 3D Graph which implicitly contains time information due to its structure. In addition, the spatial edges $E_{spatial}$ allow us to include the spatial context of each node into the association process. Similar to [BL20] We propose to use a graph neural network (GNN) to directly infer associations between detections from consecutive time frames. However, since we firstly build a spatio-temporal graph we expand this idea to a spatio-temporal graph neural network. In addition, our method can run in an online manner, as opposed

to [BL20], which only can perform tracking as offline method. This means that our method can perform tracking while receiving new detections over time, while similar works such as [BL20] need the whole set of detections of a certain scene. Compared to [Zae+22] which also uses a GNN for detection association, we do not model tracks and detections separately for a unified graph representation. However, we can impose flow conservation constraints as seen in [BL20] to efficiently compute a coherent tracking solution.
In figure 3.1 you can see a high-level overview of our method. Our pipeline can be divided into 4 modules, which are going to be explained in detail in the following sections.

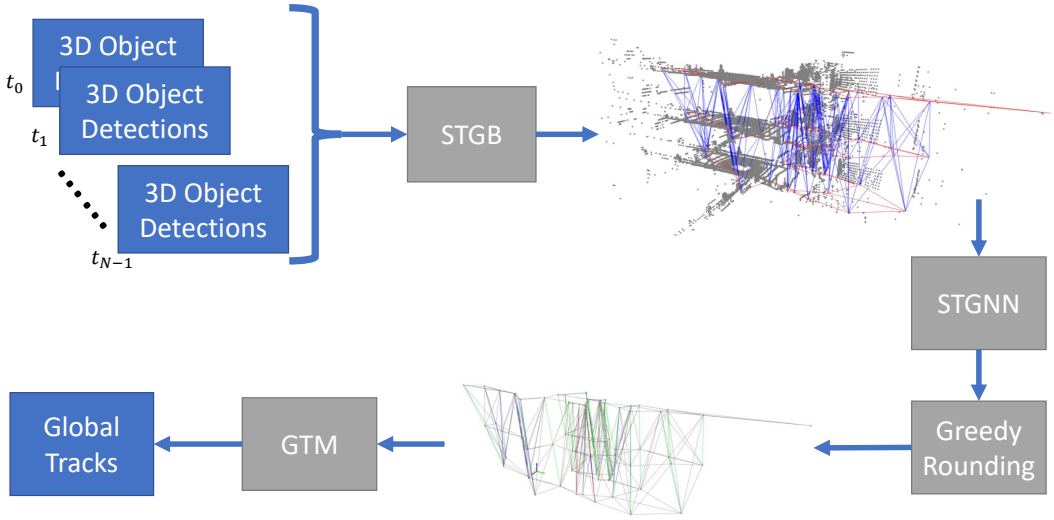

Figure 3.1: Visualization of pipeline. Firstly, we take all object detections from N consecutive frames and feed it to the spatio-temporal graph builder module (STGB). The STGB constructs a spatio-temporal graph and gives it to our spatio-temporal graph neural network (STGNN). This STGNN classifies the edges into edges that connect the same object instance or not. Afterwards we employ a greedy rounding method, adopted from [BL20], to handle inconsistent predictions from the STGNN regarding the multiple associations for the same detection. As a result, we can determine consistent tracks. Last but not least, we feed our local tracking results to a global tracking module (GTM) which associates the local tracks to the global tracks. This allows our method to work in a windowed manner, by stitching consecutive graphs together..

3.2 Graph Building

In our graph each node represents a detection and each edge represents either a temporal or a spatial relation to other detections. A temporal edge connects detections over time frames while a spatial edge connects nodes from the same time frame. Since the detections are given relative to the global world frame we first transform them into the corresponding LIDAR-frame of our LIDAR-sensor for each timeframe. As mentioned before we believe that the spatio-temporal graph can be represented in a 3-dimensional way. This means that we represent the graph mainly by the detections 3D pose, specifically by their translation, orientation is not taken into account for the graph building. However, since we only track objects that do not fly, e.g cars, pedestrians, trucks, this would mean that we place all detections on the same plane. This would overcrowd a 2D plane with detections from different time-

frames. In order to avoid this and to incorporate temporal information into a 3D graph, we propose to shift the detections from each frame by a certain shift value a_{shift} in an upwards direction. In our case the shift is performed in the z-direction, due to the conventions of the used dataset nuscenes [Cae+20]. As a result the detections are shifted by the constant value $a_{shift} \cdot t_i$, where t_i describes the i-th time frame contained in the graph and does not represent the total time or the exact timestamp. The shifting can be described as follows:

$$x_i' = x_i + \begin{bmatrix} 0 \\ 0 \\ a_{shift} \cdot t_i \end{bmatrix} \in \mathbb{R}^3 \tag{3.6}$$

with $t_i = \{0, ..., N-1\}$. As a result we receive a updated 3D translation x_i' for each detection. We display the resulting graph in figure 3.2

3.2.1 Spatial Edge Construction

A simple way to build the edges would be by connecting all nodes with each other. However, this would lead to a significant increase of computational cost. To avoid building a unnecessarily large amount of edges, we propose to use a K-nearest-neighbor (KNN) method to construct the spatial and temporal edges. We use specifically the ball-tree method [Ste89]. To perform KNN we use the shifted translations x_i' as feature. Given a set of detections O_{t_i} from time frame t_i we compute the $k_{spatial}$ neighbors that are nearest to each detection.

3.2.2 Temporal Edge Construction

As previously mentioned we want to build temporal edges to connect past detection with future detections over multiple time frames. This means that a past detection can be connected to detections in the following frame but also to detections in the subsequent frames. Our hypothesis is that by connecting detections directly, skipping multiple time frames, allows us to potentially recover from short occlusions. We limit the reach of the temporal connections by the value β, which describes the maximum temporal edge length.
As opposed to the spatial edge construction, we cannot directly apply the KNN-method directly on all detection within the graph. This is because in empirical experiments we saw that detections will tend to form connections with detections from the same time frame ,which is not our aim. As a result we propose a way to force the building of temporal edges. Given two sets of detections from different time frames O_{t_i} and O_{t_j} where $t_i < t_j$ and $\beta \geq (t_j - t_i)$, we compute edges for each detection from the past time frame O_{t_i} to the future. Each node connects to up to k_{temp} other nodes. Similar to 3.2.1 we use again the shifted translation x_i' as feature for our KNN-method. $O_{t_i} = \left\{o_{t_{i1}}, ..., o_{t_{ip}}\right\}$ consists of p observations and $O_{t_j} = \left\{o_{t_{j1}}, ..., o_{t_{jq}}\right\}$ consists of q observations. Each observation o_i has a associated shifted translation x_i'. We provide a detailed overview of the algorithm in 1.

3.3 Network Architecture

After passing the STGB-module we use our spatio-temporal graph to learn to assign flow labels according to the network flow formulation mentioned in 3.1.2. However, before we can perform any learning on our graph we need to define initial features for our nodes V and

Algorithm 1 Algorithm to build temporal edges

Require: $\beta \geq (t_j - t_i)$ and $t_i < t_j$
Given: k_{temp}
Given: $X'_{t_i} \leftarrow \left\{x'_{t_{i_1}}, ..., x'_{t_{i_p}}\right\}$
Given: $X'_{t_j} \leftarrow \left\{x'_{t_{j_1}}, ..., x'_{t_{j_q}}\right\}$
$A \leftarrow \{\}$ ▷ A is the empty set initially
for $x'_{t_{i_g}} \in X'_{t_i}$ **do**
$a_i \leftarrow knn\_method(X'_{t_j}, x'_{t_{i_g}}, k_{temp})$ ▷ a_i contains index pairs of the k_{temp} neighbors of detection o_i
$A \leftarrow A \cup a_i$
end for
return A ▷ Contains the index pairs of all k_{temp} neighbors for all detection from O_{t_i}. They represent the temporal edges.

edges E. Firstly, we assign for each edge $(i,j) \in E$ an initial edge feature vector $h^{(0)}_{(i,j)}$. Then we determine for each node $i \in V$ a initial node feature $h^{(0)}_i$.

3.3.1 Neural Message Passing

As mentioned in section 2.1 one popular way to aggregate information in a GNN is to use neural message passing (NMP). To understand the concept better we first provide a brief overview of message passing networks (MPN) before explaining our model.
The idea of NMP is to learn a function to propagate in formation from nodes and edges across the graph. The propagation is performed for L iterations. The higher maximum number of iterations L, the further the information can propagate over the graph. After propagating information over L iterations each node contains information from all other nodes at distance L in the graph. This distance is the number of edges that form a path between two nodes. This is analogous to the receptive field of CNNs[Wu+21]. Therefore high L leads to better context understanding. Note, that we want to take advantage of this concept to learn the spatial and temporal context for each node i.
The propagation step, also known as neural message passing step [Gil+17], consists of an initial edge-update step and a following node-update step. The edge update uses a learnable function $\mathcal{N}_e$ to compute an updated edge feature $h^{(l)}_{(i,j)}$, where $l \in \{1, ..., L\}$ symbolises the l-th iteration step. The edge update is as follows:

$$h^{(l)}_{(i,j)} = \mathcal{N}_e\left(\left[h^{(l-1)}_i, h^{(l-1)}_j, h^{(l-1)}_{(i,j)}\right]\right) \tag{3.7}$$

where $[\cdot]$ represent the concatenation all enclosing feature vectors. Afterwards we perform the node update which firstly involves the computation of messages $m^{(l)}_{(i,j)}$ for each node i [Gil+17]. The messages $m^{(l)}_{(i,j)}$ include information about the previous state of the node feature $h^{(l-1)}_i$ and the updated edge features $h^{(l)}_{(i,j)}$, where $j \in \mathbb{N}_i$. $\mathbb{N}_i$ represents the local neighborhood of node i. This means it contains all edges that are connected to node i. The message computation is as follows:

$$m^{(l)}_{(i,j)} = \mathcal{N}_v\left(\left[h^{(l-1)}_i, h^{(l-1)}_{(i,j)}\right]\right) \tag{3.8}$$

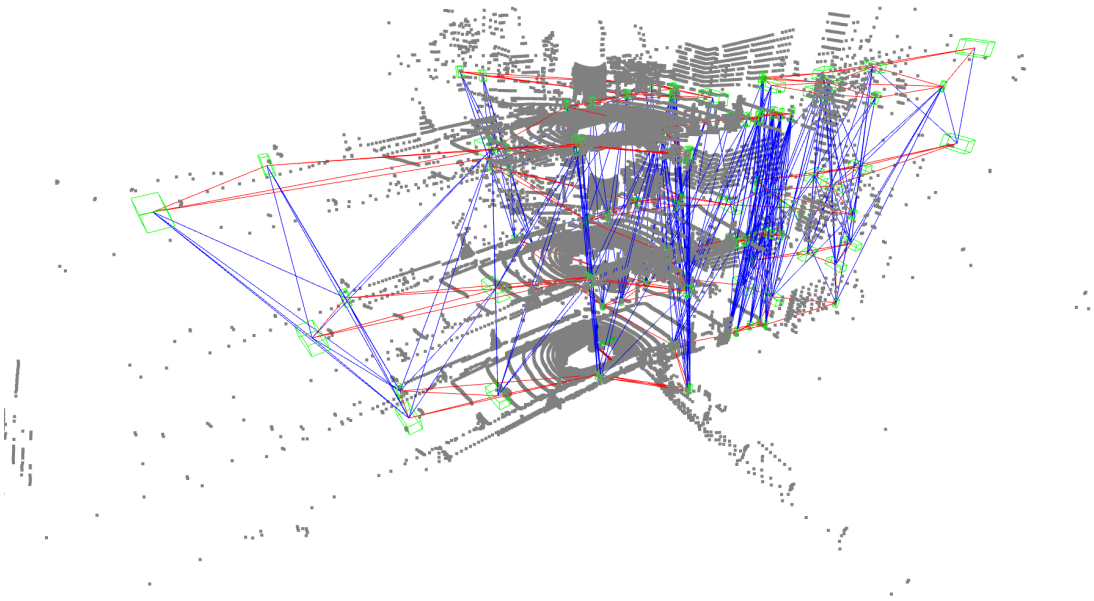

Figure 3.2: Visualization of spatio-temporal graph after being build by the STGB module.$k_{temp} = 3$, $k_{spatial} = 3$, $\beta = 2$. Blue edges represent temporal connections and red represent spatial connections. The green boxes are the corresponding bounding boxes of each detections o_i. The grey dots represent the point cloud, resulting from LIDAR-measurements.

where $\mathcal{N}_v$ is a learnable function. Then the messages $m_{(i,j)}^{(l)}$ that belong to node i are aggregated by a order invariant function Φ such as maximum, summation or the arithmetic mean.

$$h_i^{(l)} = \Phi\left(\left\{m_{(i,j)}^{(l)}\right\}_{j \in \mathbb{N}_i}\right) \tag{3.9}$$

The learnable functions $\mathcal{N}_v$ and $\mathcal{N}_e$ can be multilayer perceptrons (MLP) similar to the implementation of [BL20].

3.3.2 Time-Aware Message Passing with Spatio-Temporal Graphs

For our pipeline we adopt the MPN from [BL20], which uses a modified version of the message passing step to encode the temporal structure of the MOT graph in a explicit way. This modified version, called Time-Aware Message Passing separates all adjacent nodes $j \in \mathbb{N}_i$ of node i into two different sets $\mathbb{N}_i^{past}$ and $\mathbb{N}_i^{fut}$. $\mathbb{N}_i^{past}$ contains all adjacent nodes j from the past and $\mathbb{N}_i^{fut}$ contains the adjacent nodes from the future. Mainly the change is in the node update. It is done as follows:

$$m_{(i,j)}^{(l)} = \begin{cases} \mathcal{N}_v^{past}\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_i^{(0)}\right]\right), & \text{if } j \in \mathbb{N}_i^{past} \\ \mathcal{N}_v^{fut}\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_i^{(0)}\right]\right), & \text{if } j \in \mathbb{N}_i^{fut} \end{cases} \tag{3.10}$$

where $\mathcal{N}_v^{past}$ and $\mathcal{N}_v^{fut}$ represent learnable functions.
However, [BL20] only builds a graph with temporal edges. Therefore, the time-aware message passing will behave differently for our spatio-temporal graph.

$$m_{(i,j)}^{(l)} = \begin{cases} \mathcal{N}_v^{past}\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_i^{(0)}\right]\right), & \text{if } j \in \left\{\mathbb{N}_{i,temp}^{past} \cup \mathbb{N}_{i,spatial}^{flow\_in}\right\} \\ \mathcal{N}_v^{fut}\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_i^{(0)}\right]\right), & \text{if } j \in \left\{\mathbb{N}_{i,temp}^{fut} \cup \mathbb{N}_{i,spatial}^{flow\_out}\right\} \end{cases} \tag{3.11}$$

$\mathbb{N}^{past}_{i,temp}$ describes the set of all neighboring nodes from node i that form temporal connections to past frames. $\mathbb{N}^{fut}_{i,temp}$ analogously represents temporal connections to future frames. Due to the implementation of the model all spatial connections from node i are separated into two sets of neighboring nodes $\mathbb{N}^{flow\_in}_{i,spatial}$ and $\mathbb{N}^{flow\_out}_{i,spatial}$. The implementation of [BL20] assumes that all nodes i are sorted by a time, such that if two nodes i and j form an edge, the node with the higher index i or j is from the future. This works perfectly for graphs with purely temporal edges. As a result, for nodes j with spatial edges to node i the following holds:

$$j \in \mathbb{N}^{flow\_in}_{i,spatial}, \quad \text{if } j < i \tag{3.12}$$

$$j \in \mathbb{N}^{flow\_out}_{i,spatial}, \quad \text{if} \quad j > i \tag{3.13}$$

In figure 3.3 we visualize the problem for better understanding.
Afterwards the node update step aggregates the resulting messages separately.

$$h^{(l)}_{i,past} = \sum_{j \in \left\{\mathbb{N}^{past}_{i,temp} \cup \mathbb{N}^{flow\_in}_{i,spatial}\right\}} m^{(l)}_{(i,j)} \tag{3.14}$$

$$h^{(l)}_{i,fut} = \sum_{j \in \left\{\mathbb{N}^{fut}_{i,temp} \cup \mathbb{N}^{flow\_out}_{i,spatial}\right\}} m^{(l)}_{(i,j)} \tag{3.15}$$

Finally, we compute the updated node embedding $h^{(l)}_i$ by feeding both $h^{(l)}_{i,past}$ and $h^{(l)}_{i,fut}$ into a MLP $\mathcal{N}_v$.

$$h^{(l)}_i = \mathcal{N}_v\left(\left[h^{(l)}_{i,past}, h^{(l)}_{i,fut}\right]\right) \tag{3.16}$$

Note, that in our implementations all learnable functions $\mathcal{N}_e$, $\mathcal{N}_v$, $\mathcal{N}^{past}_v$, $\mathcal{N}^{fut}_v$ are MLPs.

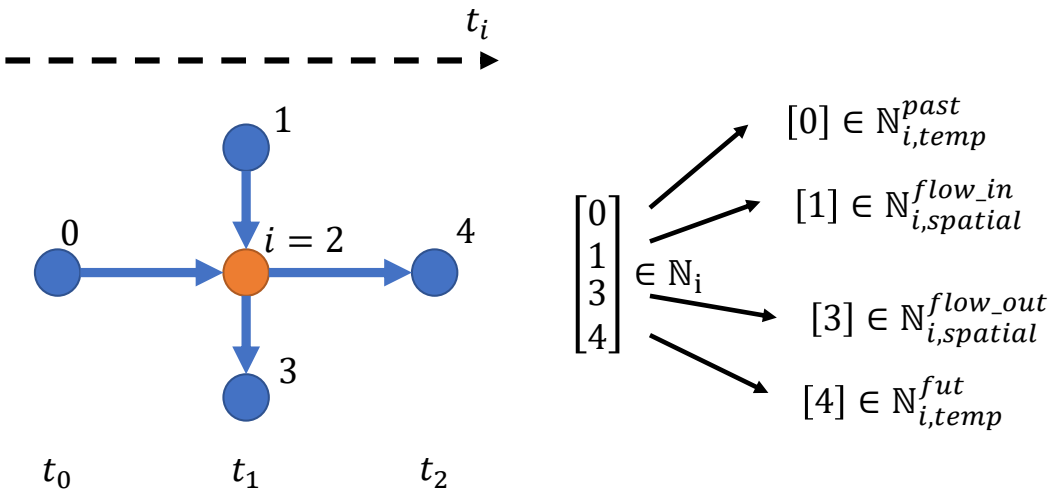

Figure 3.3: Visualization of indexing problem. Given node i with $i = 2$ and its adjacent nodes j with $j \in \mathbb{N}_i$..

3.3.3 Feature Selection

For our initial node features we selected the shifted translation of each detection x'_i and the time frame t_i.

$$h^{(0)}_i = \left[x'_i, t_i\right] \in \mathrm{R}^4 \tag{3.17}$$

Our initial edge features $h_{(i,j)}^{(0)}$ are composed of one binary value $q_{(i,j)}$, which describes the edge type. In order to increase the edge feature space, we concatenate the binary value twice. We define it as follows:

$$h_{(i,j)}^{(0)} = \left[q_{i,j}, q_{i,j}\right] \in \mathrm{R}^2 \tag{3.18}$$

with

$$q_{(i,j)} = \begin{cases} 0, & \text{if} \quad (i,j) \in E_{spatial} \\ 1, & \text{if} \quad (i,j) \in E_{temp} \end{cases} \tag{3.19}$$

Furthermore, with our method it is also possible to add appearance features from the 3D object detections and possibly also the orientation vector of the object detections, to the node features. It would also be possible to include a difference in bounding box dimensions to the edge features, as seen in [BL20].

3.3.4 Training and Inference

Our training is similar to [BL20] work. After performing l_0 message passing steps we start to classify all edges $(i,j) \in E$, temporal and spatial, by feeding the edge features $h_{(i,j)}^{(l)}$ into a classification head. This classification head is composed of a MLP $\mathcal{N}_e^{class}$ and a sigmoid function $\sigma(\cdot)$ for binary classification. The MLP $\mathcal{N}_e^{class}$ reduces the whole feature vector to a single value score. Afterwards the score is given to a sigmoid function to limit our edge predictions $\hat{y}_{(i,j)}^{(l)}$ value to a range between 0 and 1.

$$\hat{y}_{(i,j)}^{(l)} = \sigma\left(\mathcal{N}_e^{class}\left(h_{(i,j)}^{(l)}\right)\right) \in [0,1] \tag{3.20}$$

Since we perform binary classification we use the binary cross entropy loss of our edge predictions. We use the same loss as in [BL20], except we include the edge predictions of our spatial edges. The loss is as follows:

$$\mathcal{L} = \frac{-1}{|E|} \sum_{l=l_0}^{L} \sum_{(i,j)\in E} w \cdot y_{(i,j)} \log\left(\hat{y}_{(i,j)}^{(l)}\right) + (1 - y_{(i,j)}) \log\left(1 - \hat{y}_{(i,j)}^{(l)}\right) \tag{3.21}$$

where $y_{(i,j)}$ represents the binary edge label. $y_{(i,j)}$ is 1 for edges that connect nodes that represent the same object. In other words both nodes i and j need to be part of the same ground truth trajectory T_i. $y_{(i,j)}$ also represents the flow labels which is further explained in 3.1.2. Due to the label imbalance, there are more negative labels than positive, we use a positive scalar w to give positive predictions a higher loss. Therefore, the optimization algorithm will tend to minimize the loss of positive labels.
At inference time, we pick the edge predictions $\hat{y}_{(i,j)}^{(L)}$ from the L-th iteration, the last iteration, as final solution from the GNN-module. $\hat{y}_{(i,j)}^{(L)}$ is then passed to the greedy rounding module.

3.4 Greedy Rounding

After performing L iterations over our GNN, we want to use the last edge predictions $\hat{y}_{(i,j)}^{(l)}$ to infer associations between detections at different time frames. However, our predictions can lead to inconsistent association, such as a node with multiple incoming active edges. This would mean that the node is part of multiple tracklets which is impossible. As a result, our

predictions $\hat{y}$ are not valid flow formulations as long as they violated the flow conservation constraints, seen in 3.1.2. To satisfy the flow conservation constraints we adopt a greedy method that finds a coherent solution.

3.4.1 Greedy Method

By only regarding our temporal edges E_{temp} and all nodes V we can define a new Graph $G' = (V, E_{temp})$. This allows us to profit from the flow formulation and its flow conservation constraints. These constraints ensure that a node cannot belong to more that one trajectory. In order to counter this infringement, we adopt a simple greedy rounding scheme from [BL20]. To satisfy the flow conservation constraints, we set for each node the incoming edge prediction $\hat{y}^{(l)}_{(i,j*)}$ with the highest score to 1. All remaining incoming edge predictions are set to 0. The same procedure is performed analogously for outgoing edges. For a detailed explanation to the computational efficiency we refer to the supplement material of [BL20]. We depict the used method in algorithm 2 The a figure of resulting output graph can be seen in

Algorithm 2 Algorithm for Greedy Rounding based on Greedy Projection from [BL20]

Given: Graph $G = (V, E_{temp})$
Given: Edge Predictions $\hat{y}^L$
Return: feasible flow solution y
for $(i,j) \in E_{temp}$ **do**
  if $\hat{y}^L_{(i,j)} > 0.5$ **then**
    $y_{(i,j)} \leftarrow 1$
  else
    $y_{(i,j)} \leftarrow 0$
  end if
end for
for $i \in V$ **do**
  if Constraint 3.4 is violated **then**
    $j* \leftarrow \text{argmax}_{j \in \mathbb{N}^{past}_{i,temp}} \hat{y}^L_{(i,j)}$
    for $j \in \mathbb{N}^{past}_{i,temp} \setminus \{j*\}$ **do**
      $y_{(i,j)} \leftarrow 0$
    end for
  end if
  if Constraint 3.5 is violated **then**
    $j* \leftarrow \text{argmax}_{j \in \mathbb{N}^{fut}_{i,temp}} \hat{y}^L_{(i,j)}$
    for $j \in \mathbb{N}^{fut}_{i,temp} \setminus \{j*\}$ **do**
      $y_{(i,j)} \leftarrow 0$
    end for
  end if
end for

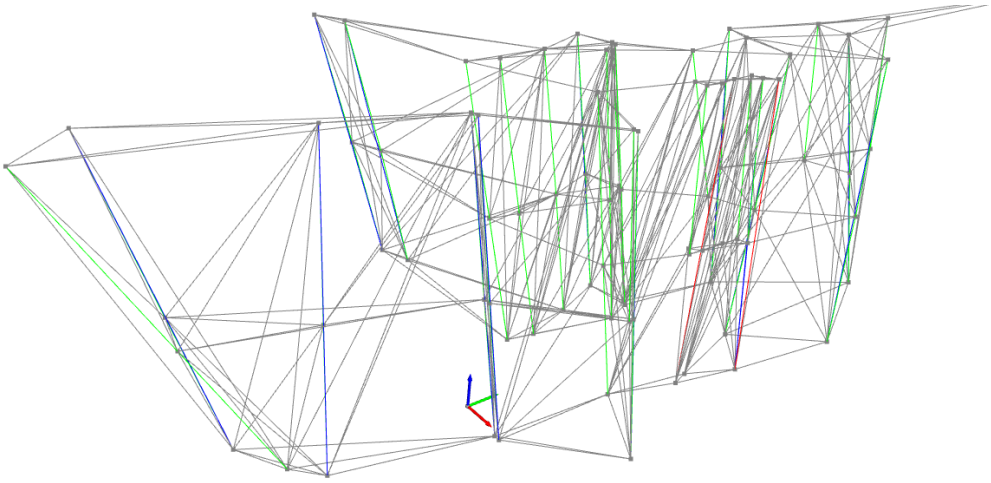

Figure 3.4: Visualization of spatio-temporal graph after greedy rounding. Green edges are true positive. Red edges are false positives. Blue edges represent false negatives. Grey edges represent either true negatives or spatial edges.

3.5 Tracker Routine

As mentioned before our MOT method works as online method. Therefore, it can only work with past and current detections at global time frame τ_i. As a result, our method performs tracking within a window of N time frames and then shifts by $N-1$ time frames. Note that τ_i represents the global time frame within a scene Γ, which is composed of time frames $\tau_i \in \Gamma = [1, \ldots, \tau_{end}]$. t_i is the local time frame indicator within a frame.

3.5.1 From Local to Global Tracking

We perform tacking within a window, that means we take all detections from the selected N time frames and proceed with the steps explained in 3.2, 3.3 and 3.4. Therefore, we build a local graph $G_{\tau_i} = (V_{\tau_i}, E_{\tau_i})$ for global time frame τ_i. Subsequently, we deduce for each node $i \in V_{\tau_i}$ a local tracking id w'_i, which is only valid within the window. To perform tracking over a whole scene Γ, we want to assign each node in each window the corresponding global tracking id w_i. Therefore, we stitch the local graphs together at a commmon timeframe τ_i. For stitching we are given the graph $G_a = (V_a, E_a)$ from the previous window and a current graph $G_b = (V_b, E_b)$ from the current window at time frame τ The stitching is done by determining the corresponding nodes $i \in V_a^{\tau_i}$ where $V_a^{\tau_i}$ represents all nodes from graph G_a that have timestamp τ_i to nodes $j \in V_b^{\tau_i}$, where $V_b^{\tau_i}$ represents all nodes from graph G_b that have timestamp τ_i. Both of these sets represent the same detections $\mathcal{O}_{\tau_i} \subset \mathcal{O}$, where $\mathcal{O}_{\tau_i}$ represents all detections from global time frame τ_i. By just comparing the underlying observation o_i of each node i and j, we can assign the global tracking ids from nodes $i \in V_a^{\tau_i}$ from the previous window to nodes $j \in V_b^{\tau_i}$ from the current window. In figure 3.5 we depicts the window shifting with the global and local time frame expressions for clarification.

3.5.2 Routine Explanation

At the beginning we define the first local tracking ids from the first window as global tracking ids for initialization. Afterwards we continue handing down the tracking ids over graph stitching. However, if there is a time frame without any detections we stop the graph stitching and terminate the associated tracklets T_i. Then we shift the window frame-by-frame until we find a frame that allows our SPGB-module to build a graph with N time frames without a lack of detections.
When reaching the end, we need to take care that we shift the window outside of the time frame space Γ. Therefore, if $\tau_{end} - \tau_i < (N-1)$ we perform tracking on the last available window before the scene ends. Then we assign the global tracking ids at time frame τ_i. This is visualized on the right hand side of figure 3.5.

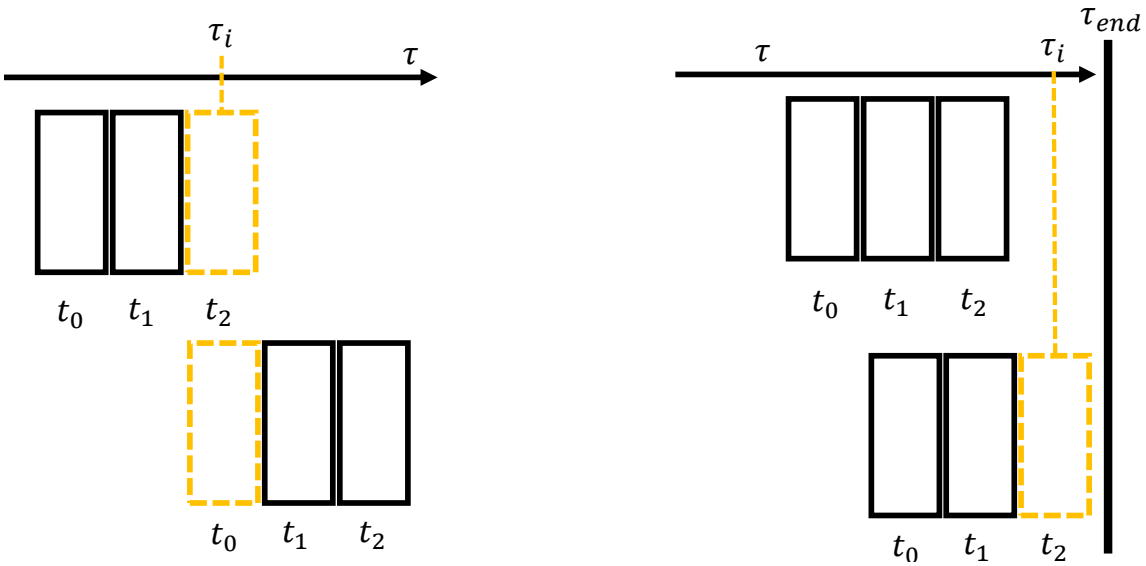

Figure 3.5: Visualization of window shifting of tracker on the left hand side. Visualization of graph stitching at the end of a scene on the right hand side. At the left hand side the upper window represents a past window and the bottom window is the current window which needs to be stitched at global time frame τ_i to adopt the global tracking ids. On the right hand side the previous window (upper) is less than N steps away from the end of the scene. Therefore, the current window must be initiated at global time $\tau_{end} - N$ and stitched at time τ_i .

Algorithm 3 Algorithm for Tracking over whole scene

Given: all time frames Γ
Given: all Observations $\left\{\mathcal{O}_{\tau_i}\right\}_{\tau_i \in \Gamma}$
$W \leftarrow \{\}$ ▷ Set of Global Tracking Ids
while $\tau_i \neq \tau_{end}$ **do**
  if $\mathcal{O}_{\tau_i} = \{\}$ or $(\tau_{end} - \tau_i < (N-1))$ **then**
    if $(\tau_{end} - \tau_i < (N-1))$ and $\mathcal{O}_{\tau_i} \neq \{\}$ **then** ▷ Last window reached
      $G_{\tau_{end-N}} = \text{build\_graph}(\{\mathcal{O}_{\tau_{end-N}}, \ldots, \mathcal{O}_{\tau_{end}}\})$
      $\{w'_i\}_{i \in V_{\tau_{end-N}}} = \text{perform\_tracking}(G_{\tau_{end-N}})$ ▷ Feeds graph into STGNN and greedy rounding module
      $\text{assign\_global\_ids}\left(G_{\tau_{end-N}}, G_{\tau_j}, \{w'_i\}_{i \in V_{\tau_{end-N}}}, \{w'_j\}_{j \in V_{\tau_j}}, W, \tau_i\right)$
    else
      $\{w'_j\}_{j \in V_{\tau_j}} \leftarrow \{\}$ ▷ local tracking Ids of previous window set to empty set
      $G_{\tau_j} \leftarrow \{\}$ ▷ Graph of previous window set to empty set
    end if
    $\tau_i \leftarrow \tau_i + 1$
  else
    $G_{\tau_i} = \text{build\_graph}(\{\mathcal{O}_{\tau_i}, \ldots, \mathcal{O}_{\tau_{N+i}}\})$
    $\{w'_i\}_{i \in V_{\tau_i}} = \text{perform\_tracking}(G_{\tau_i})$
    if $G_{\tau_j} \neq \{\}$ **then** ▷ normal graph stitching
      $\text{assign\_global\_ids}\left(G_{\tau_i}, G_{\tau_j}, \{w'_i\}_{i \in V_{\tau_i}}, \{w'_j\}_{j \in V_{\tau_j}}, W, \tau_i\right)$
    else ▷ No previous window available
      $\text{init\_new\_track\_ids}\left(\{w'_i\}_{i \in V_{\tau_i}}, W\right)$
    end if
    $\{w'_j\}_{j \in V_{\tau_j}} \leftarrow \{w'_i\}_{i \in V_{\tau_i}}$ ▷ assign current to previous window
    $G_{\tau_j} \leftarrow G_{\tau_i}$ ▷ assign current to previous window
    $\tau \leftarrow \tau + (N-1)$

  return W ▷ contains all global tracking ids for each node over the whole scene

Chapter 4

Experiments

In this chapter we are going to explain certain implementation details and we are going to present our results.

4.1 Implementation Details

4.1.1 Nuscenes Dataset

We use the Nuscenes dataset [Cae+20] for our experiments. It is used for autonomous driving tasks benchmarking. It provides 1000 driving scenes with a duration of 20 seconds captured in Boston and Singapore. It contains LIDAR, Camera and Radar signals. Therefore, it provides a wide variety a sensor inputs that can be used for fusion as seen in [KOL21]. It has 23 object classes annotated with 3D bounding boxes at a rate of 2Hz. However, these classes include barriers, debris and traffic cones which we do not take into account for the MOT challenge. For MOT we only take the following classes into account:

- bicycle
- motorcycle
- pedestrian
- bus
- car
- trailer
- truck
-

Some of these tracking classes are a union of the official Nuscenes classes. Pedestrian is a union of the classes:

- human.pedestrian.adult
- human.pedestrian.child
- human.pedestrian.construction_worker

- human.pedestrian.police_officer

Bus is a union of:

- vehicle.bus.bendy
- vehicle.bus.rigid

4.1.2 3D MOT Metrics

In order to evaluate MOT - methods, [BS08] and [Wen+20b] introduces metrics, which were specifically designed for MOT. These allow to compare different MOT methods over quantitative scores.
[BS08] introduces the CLEAR MOT metrics. They introduce the multiple object tracking accuracy (MOTA) and the multiple object tracking precision (MOTP).

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t} \tag{4.1}$$

FP_t represent the false positive tracking for time t. FN_t represents the number of missed detections which were not assigned to any tracklet T_i. IDS_t represent the number of missmatched tracklets for time t. That means that two tracklets T_i and T_j were assigned detection from the opposite tracklet at time t, also known as identity switches. GT_t represents the number of objects present at time t. Therefore, MOTA incorporates a number of failure cases, missed detections, falsely tracked detections and missmatches.

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t TP_t} \tag{4.2}$$

$d_{i,t}$ represents the position error of track i at time t. TP_t represents the number of correct matches at time t. Therefore, MOTP is computed of the total error in estimated position for all tracks i over all time frames t, averaged by the total number of correct matches. MOTP represents the trackers ability to estimate object positions precisely.
Even though MOTA and MOTP are regularly used for 2D MOT, they do not take the object detections, detection score s_i into account. Therefore, the thresholds for object detections with sufficently high certainty s_i must be manually set. To address this issue [Wen+20b] introduces the metrics average MOTA (AMOTA) and average MOTP (AMOTP). In addition [Wen+20b] propose the scaled AMOTA (sAMOTA), which normalises the metric between 0 and 1. However, Nuscenes defines the sAMOTA as AMOTA for their own evaluation metrics. Since, we use the evaluation method of the Nuscenes dataset, we adopt their convention to define the AMOTA and AMOTP [Cae+20].

$$MOTA_r = max(0, 1 - \frac{FN_r + FP_r + IDS_r - (1-r) \cdot P}{r \cdot P}) \tag{4.3}$$

$$AMOTA = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}} MOTA_r \tag{4.4}$$

r represents the recall. P refers to the number of ground-truth positives for the current class. n represents the number of interpolations.

$$AMOTP = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}} MOTP \tag{4.5}$$

AMOTA and AMOTP can be seen as integrals over the MOTA and MOTP curves at different recall values r. To integrate the curves a n-point interpolation is used. [Cae+20] and [Wen+20b] use a $n = 40$. To normalize the AMOTA between 0 and 1, $(1-r) \cdot P$ is subtracted from FN_r. In addition, the scaling factor r is added to the denominator and a maximum is added. These changes all together guarantee that AMOTA spans between $AMOTA \in [0,1]$ and the it brings the three failure cases into similar value ranges.

4.1.3 Graph Building

We use the Ball-Tree method [Ste89] implementation from [Ped+11] as KNN-method.

4.1.4 Training

We trained our model for 50 epochs. The batch size was 4. We train on a Nvidia Geforce RTX 2080. Our number of iterations $L = 5$ for the message passing steps. We begin classification at $l_0 = 3$. Our aggregation function is $\Phi = \sum$. For optimization we use a Adam-optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$ and a weight decay of 0.

- $k_{temp} = 3$
- $a_{shift} = 20$
- $k_{spatial} = 3$
- $\beta = 2$
- $N = 3$

Training only with graphs with object detections of class "car". We use the model from the 48 epoch, since it yields the best validation loss. Our training loss to validation loss can be seen in figure 4.1

4.2 Feasibility Study

To evaluate if our approach really works we first wanted to conduct a feasibility study. Therefore, we use the ground truth (GT) annotations from Nuscenes as detections. In addition, we do not use the GNN for our inference but use the computed edge labels y as edge predictions $\hat{y}$. In this section we evaluate all experiments on the official Nuscenes validation split. We track all official classes for the Nuscenes tracking challenge.

$$\hat{y} = y \tag{4.6}$$

In addition, we set the detections score s_i of each detection to 1 because they are given by the ground truth. Therefore the recall will display large values. We mainly want to evaluate three parameters:

- number of frames per graph N
- the temporal KNN parameter k_{temp}
- the spatial KNN parmeter $k_{spatial}$

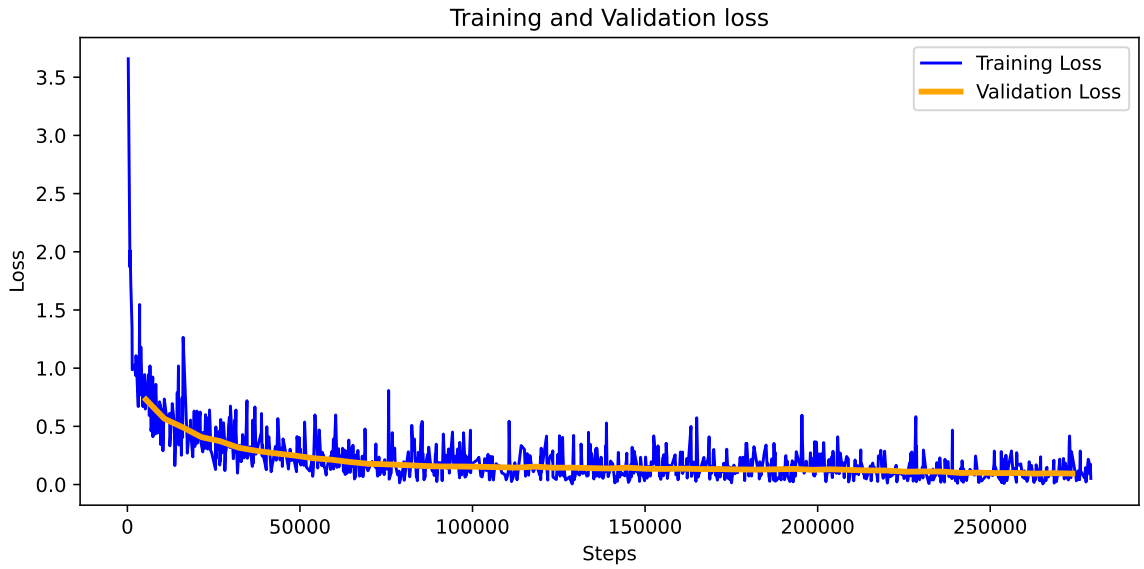

Figure 4.1: Visualization of training loss and validation loss over number of steps. These are the number iteration due to mini-batch processing.

- maximum temporal edge length β

For comparison purposes we define a base configuration:

- $N = 3$
- $k_{temp} = 3$
- $k_{spatial} = 3$
- $a_{shift} = 20$
- $\beta = 2$

4.2.1 Influence of Spatial Edges

Since we are given the GT edge labels y as edge predictions $\hat{y}$, we are not using the GNN in this experiment. Thus, the influence of spatial edges should be none for this configuration. Nevertheless, we want to show a comparison of results to prove our hypothesis, seen in table 4.1. We vary $k_{spatial}$, while fixing the remaining parameters to:

- $N = 3$
- $k_{temp} = 6$
- $a_{shift} = 20$
- $\beta = 2$

As we can see a increase in $k_{spatial}$ does not affect any of the presented metrics. As a result, the results confirm our hypothesis.

$k_{spatial}$	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
6	0.9487	0.0699	0.9954	99,766	2,717	738	1,393
3	0.9487	0.0699	0.9954	99,766	2,717	738	1,393

Table 4.1: MOT-results after varying the number of frames $k_{spatial}$. Using GT annotations from Nuscenes validation split and GT edge labels as predictions

4.2.2 Influence of Frames per Graph

Firstly, we vary the N while fixing the remaining parameters to:

- $k_{temp} = 3$
- $k_{spatial} = 3$
- $a_{shift} = 20$
- $\beta = 2$

The results can be seen in table 4.2. First of all, we can see that a $N = 3$ yields the highest AMOTA. The AMOTA decreases when increasing the number of frames per graph N. In addition, a increase of N also yields a higher AMOTP, which means that the tracker is less precise to estimate the objects position. We believe that by spanning large graphs with $N = 12$ the connectivity defined by $k_{temp} = 3$ is not sufficient to connect nodes from the same trajectory. This would explain the rapid increase in FN for $N = 12$. Even though the FP and IDS are lower for $N = 12$ which would lead to a better AMOTA, the FN increases more than 20 times the initial FP value for $N = 3$.
The increase in AMOTP is probably partly caused by the by the lower amount TP tracks. The position error should remain constant since we work with GT-annotations as detections.
In addition, we depict a close-up of a graph with $N = 12$ in figure 4.2. In figure 4.2 we can observe that a possible reason for the increase in AMOTP is the high amount of nodes on the left hand side of the image. Since $k_{temp} = 3$ it is not high enough to establish temporal connections between nodes which belong to the same GT-trajectory. In addition, on the rigth hand side of figure 4.2 we can observe the ideal case for tracking. Three objects appear jointly, and move in similar directions and since $k_{temp} = 3$ and $k_{spatial} = 3$ each node can encode the spatial and temporal context of this subgraph. We depict the entire graph in the appendix on figure B.3

N	AMOTA↑	AMOTP↓	RECALL↑	MOTA↑	TP↑	FP↓	FN↓	IDS↓
3	**0.9201**	**0.1268**	**0.995**	**0.9378**	**96,347**	2,729	**738**	4,812
6	0.9129	0.1411	0.986	0.9294	95,643	2,741	1,506	4,748
12	0.7672	0.4403	0.847	0.7985	82,608	**2,367**	15,048	**4,241**

Table 4.2: MOT-results after varying the number of frames N. Using GT annotations from Nuscenes validation split and GT edge labels as predictions

4.2.3 Influence of Temporal Edges

We vary k_{temp} while fixing the remaining parameters to:

- $N = 3$

- $k_{spatial} = 3$
- $a_{shift} = 20$
- $\beta = 2$

Table 4.3 shows us that a higher k_{temp} leads to a higher AMOTA. Additionally, TP increases while FP and FN decreases. In addition, the IDS also decrease significantly. We think this is the result of higher connectivity between nodes. This leads to more possible connections, and thus increases the probability of connecting to nodes that belong to the same trajectory T_i. Since we use the GT edge labels, we can automatically see if the graph connects nodes that are from the same Trajectory. As a result, less IDS happen due to more consistent tracking.

k_{temp}	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
3	0.9201	0.1268	0.9954	96,347	2,729	738	4,812
6	0.9487	0.0699	0.9954	99,766	2,717	738	1,393
12	**0.9558**	**0.0558**	0.9954	**100,780**	**2,714**	**737**	**380**

Table 4.3: MOT-results after varying k_{temp}. Using GT annotations from Nuscenes validation split and GT edge labels as predictions

4.2.4 Influence of Maximum Temporal Edge Length

Since we also connect nodes across multiple time frames, to allow better recovery from small occlusions, we wanted to evaluate the effectiveness of the maximum temporal edge length β. Even though a $N = 12$ yields poor results, we wanted to vary $beta$ over a large range, while fixing the remaining parameters to:

- $N = 12$
- $k_{temp} = 3$
- $k_{spatial} = 3$
- $a_{shift} = 20$

As we can see in table 4.4 both AMOTA and AMOTP only change minimally with increasing β. A possible explanation could be that the additional temporal edges due to a higher β do not connect to a node that is part of the GT trajectory. Since we limit $k_{temp} = 3$ to a low number, there are probably objects which do not form temporal edges with nodes from the same GT-trajectory. This is highly likely for objects that move quickly in between frames. As a result, the additional temporal edges only connect to nodes that belong to other GT trajectories.

β	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
2	**0.7672**	0.4403	0.8472	82,608	**2,367**	15,048	4,241
6	0.7668	0.4403	**0.8477**	82,759	2,439	14,957	4,181
11	0.7662	0.4403	**0.8477**	**82,761**	2,464	**14,956**	**4,180**

Table 4.4: MOT-results after varying β. Using GT annotations from Nuscenes validation split and GT edge labels as predictions

4.3 Learned Tracker Results

After seeing promising results in the feasibility study. We want to continue our evaluation using our GNN model for inference of active edges. Firstly, we continue evaluating on the GT annotation, to see the gap between the feasibility study and our model. Then we use publicly available CenterPoint detections [YZK21] to compare our method to state-of-the-art methods, CBMOT [BSZ21] and EagerMOT[KOL21] that also use the same detections.

4.3.1 Inference on the Ground Truth Annotations

For the inference on the GT Annotations, we want to analyze the following parameters:

- $k_{spatial}$
- N
- k_{temp}
- $k_{spatial}$ and k_{temp} together

For comparison purposes we again use the same base configuration as previously in 4.2:

- $N = 3$
- $k_{temp} = 3$
- $k_{spatial} = 3$
- $a_{shift} = 20$
- $\beta = 2$

Influence of Spatial Edges

Since we use the GNN model we can now test the extend of our hypothesis of having better prediction results due to our spatial features. We vary $k_{spatial}$ while fixing the remaining parameters to:

- $N = 3$
- $k_{temp} = 3$
- $\beta = 2$
- $a_{shift} = 20$

In table 4.5 we can see that an increase in $k_{spatial}$ leads to a slight increase in AMOTA while also minimally decreasing the AMOTP. With a increasing $k_{spatial}$ we observe a decrease in FP and FN which leads to a better AMOTA. Therefore, we believe that the spatial context helps to predict active edges. However, due to the small feature space of R^2 given to spatial edges, the effect is probably limited. In addition, the mentioned indexing problem of our GNN architecture might also be a source of error. As seen in 3.3.2, the spatial edges are encoded together with the temporal edges with different MLPs $\mathcal{N}_v^{past}$ and $\mathcal{N}_v^{fut}$. Since they are not encoded separately, with a hypothetical MLP for spatial information $\mathcal{N}_v^{spatial}$, some of the spatial information can be lost, in the time-aware message passing step. In conclusion, $k_{spatial}$ positively affects AMOTA and AMOTP.

$k_{spatial}$	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
3	0.8581	0.2244	0.9933	**16.702**	92,278	5,024	8,752
6	**0.8657**	**0.2149**	0.9933	14.803	92,392	4,484	**8,646**
12	0.8617	0.2239	0.9933	13.711	**91,622**	**4,194**	9,423

Table 4.5: MOT-results after varying $k_{spatial}$. Using GT annotations from Nuscenes validation split but GNN edge predictions

Influence of Frames per Graph

Similarly to subsection 4.2.2 we wanted to evaluate the influence of N on the MOT performance of our GNN, with the same configurations. Therefore, we vary N while fixing the remaining parameters to:

- $k_{spatial} = 3$
- $k_{temp} = 3$
- $\beta = 2$
- $a_{shift} = 20$

As seen previously, the AMOTA decreases with increasing N, while AMOTP increases. However, the AMOTA from 4.2.2 for each configuration is higher than the AMOTA from the corresponding configuration with the GNN predictions. This shows the gap between the predictive performance of our GNN and the ideal case with perfectly assigned active edges. This gap is reinforced by a higher count of FP and IDS compared to 4.2.2. Moreover the AMOTP is generally higher than in 4.2.2. We believe that this is due to a generally lower TP. Therefore, the percentage of correctly assigned track ids is higher.
Generally we believe that this experiment also suffers from a lack of temporal connections. This limits the probability that a node $j \in \mathbf{N}_i$ is also part of the same tracklet composed of the nodes $\{i, j, ...\}$. In figure 4.3 we depict a close-up of the graph with $N = 12$. As we can see there are situations where there plenty of objects close to each other. Therefore, the amount of temporal edges k_{temp} does not allow that each nodes establishes a temporal connection to a node which is part of the same GT-trajectory. We depict the whole graph in the appendix on the figures B.1 and B.2.

N	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
3	**0.8581**	**0.2244**	**0.9933**	92,278	**5,024**	**867**	**8,752**
6	0.7935	0.2891	0.9840	89,411	8,103	1,645	10,841
12	0.6715	0.5494	0.8469	**77,070**	7,391	15,135	9,692

Table 4.6: MOT-results after varying N. Using GT annotations from Nuscenes validation split but GNN edge predictions

Influence of Temporal Edges

Similarly to the feasibility study we want to evaluate the influence of k_{temp} on the MOT performance of our learned tracking method. Therefore we vary k_{temp} while fixing the remaining parameters to:

- $k_{spatial} = 3$
- $N = 3$

- $\beta = 2$
- $a_{shift} = 20$

As we can see in table 4.7, with increasing k_{temp} we see an increase in AMOTA and decrease in AMOTP. The AMOTA and AMOTP behave similarly to our results in 4.2.3. However, the AMOTA and AMOTP from our learned tracker generally lie under,(respectively for AMOTP above) the results from the feasibility study in 4.2.3. We believe that this shows the gap between our GNN-based tracker and the optimal case tracker shown in 4.2.3. A potential source of error could come from the sparse amount of features that we assign our nodes, $h_i^{(0)}$, and edges,$h_{(i,j)}^{(0)}$, as seen in 3.3.2. As a result, k_{temp} improves AMOTA and AMOTP.

k_{temp}	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
3	0.8581	0.2244	0.9933	92,278	5,024	867	8,752
6	0.8972	0.1609	0.9933	94,814	4,074	848	6,235
12	**0.9060**	**0.1489**	**0.9934**	**96,454**	**3,456**	**820**	**4,623**

Table 4.7: MOT-results after varying k_{temp}. Using GT annotations from Nuscenes validation split but GNN edge predictions

Influence of Temporal Edges and Spatial Edges

We observe a increase in AMOTA and a respective decrease in AMOTP by increasing k_{temp} and $k_{spatial}$ individually. Now, we want to evaluate the tracking performance of our learned tracker by increasing k_{temp} and $k_{spatial}$ jointly. For comparison purposes we also included the base configuration to our experiment. We vary k_{temp} and $k_{spatial}$ jointly while fixing to the remaining parameters to:

- $N = 3$
- $\beta = 2$
- $a_{shift} = 20$

As expected the AMOTA increases when increasing k_{temp} and $k_{spatial}$. Moreover, AMOTP decreases with increasing k_{temp} and $k_{spatial}$. In table 4.8 we compare the results from table 4.7 with the identical configuration but with higher $k_{spatial}$. As a result, we can determine the average increase of AMOTA ∇, based on configurations with $k_{spatial} = 6$.

$$\nabla = \frac{100}{|B|} \cdot \sum_{k_{temp} \in \mathcal{B}} \left(\frac{AMOTA_{k_{temp},k_{spatial}=6} - AMOTA_{k_{temp},k_{spatial}=3}}{AMOTA_{k_{temp},k_{spatial}=3}} \right) \quad \text{with} \quad \mathcal{B} = \{6, 12\} \tag{4.7}$$

We receive a $\nabla = 1.398\%$, which means that the increase in $k_{spatial}$ from $k_{spatial} = 3$ to $k_{spatial} = 6$ leads on average to an increase of 1.398% compared to the previous configuration. This increase is reflected in the drop of FP, FN and IDS.
In addition, the improvement in AMOTA is joined with a decrease in AMOTP.
Furthermore, from table 4.8 we can determine that regarding all past experiments, the configuration $k_{temp} = 12$ and $k_{spatial} = 6$ yields the best performance in terms of AMOTA.

k_{temp}	$k_{spatial}$	AMOTA↑	AMOTP↓	RECALL↑	TP↑	FP↓	FN↓	IDS↓
3	3	0.8581	0.2244	0.9933	92,278	5,024	867	8,752
6	3	0.8972	0.1609	0.9933	94,814	4,074	848	6,235
12	3	0.9060	0.1489	0.9934	96,454	3,456	820	4,623
6	6	0.9090	0.1418	**0.9952**	96,157	3,673	**811**	4,929
12	6	**0.9195**	**0.1244**	0.9950	**96,464**	**3,222**	845	**4,588**

Table 4.8: MOT-results after varying jointly k_{temp} and $k_{spatial}$. Using GT annotations from Nuscenes validation split but GNN edge predictions.

4.3.2 Inference on Centerpoint Detections

To compare our method to other state-of-the-art (SOTA) methods, we load public 3D detections provided by CenterPoint [YZK21]. We ran the evaluation of CBMOT [BSZ21] and EagerMOT [KOL21] on our workstation. Both use CenterPoint detections for their evaluation on the validation split. However, EagerMOT [KOL21] uses a deprecated sample of detections called "centerpoint_voxel_1440_dcn(flip)". While, CBMOT [BSZ21] uses the updated version "centerpoint_voxel_1440".
To make the evaluation comparable, we once again run the evaluation first on the base configuration and then we adopted best performing configuration parameters from the evaluation on GT annotations seen in 4.3.1.
We vary k_{temp} and $k_{spatial}$ jointly while fixing to the remaining parameters to:

- $N = 3$
- $\beta = 2$
- $a_{shift} = 20$

Comparison to CBMOT

CBMOT [BSZ21] is a tracking-score-based 3D MOT method, as mentioned in 2.3.6. It refines the tracking of other SOTA 3D MOT methods. In our case CBMOT uses CenterPoint detections [YZK21] and 2D Trajectories from PointTracker [ZKK20]. As a result it uses camera and Lidar sensors. In addition, it introduces a novel tracking score update module, which updated for each tracklet a designated tracking score after each time frame. This tracking score is used for tracklet termination. This tracking-score update module can use different functions for its updates, maximum, addition, multiplication, learnable functions. For our experiment we regard the two best perfoming configurations of CBMOT.
CBMOT* uses 2D and 3D tracklets and uses a multiplication for its score update. CBMOT^ also uses 2D and 3D tracklets. However, it uses a neural network for its score update.
Our results are depicted in tables 4.9 and 4.10. Our method yields AMOTA scores that are significantly lower than the AMOTA scores from CBMOT. There is a gap of 50.942% percentage points between the best performing configuration of our method and the worst performing method of CBMOT, in terms of AMOTA. As we can see in table 4.10 our method yields more than double the FN as opposed to CBMOT's FN score. In addition, our method has a significant increase in IDS. IDS of ourr method is at least 36 times larger than the lowest IDS score of CBMOT. We believe that the main reason for the significant gap of AMOTA between our methods is the existence of false positive object detections provided by the CenterPoint detecions. Due to our assumption in the flow formulation, explained in 3.1.2, that all detections are true positives, our method cannot handle false positive object detecions. Thus, our method assings tracking ids to each available detection. As a result, our method is prone to

associate false positive detections to true positive ones. This would explain the significant increase in IDS, which lowers the AMOTA significantly. In figure 4.4 we depict the associations between our method and CBMOT. It shows that there is a significant amount of false positive detections, which our method cannot filter out. In addition, we can see examples of our method, associating false positive detections from the previous frame to true positive detections.
Moreover, we can see that our method's FP are almost half of CBMOT's FP, when configured with $k_{temp} = 12$ and $k_{spatial} = 6$

Method	k_{temp}	$k_{spatial}$	AMOTA↑	AMOTP↓	RECALL↑
CBMOT* [BSZ21]	-	-	**0.7219**	0.5337	**0.73784300**
CBMOT^ [BSZ21]	-	-	0.7196	**0.4869**	0.73473442
ours	3	3	0.2101	1.1948	0.42755481
ours	12	6	0.2021	1.3185	0.43722993

Table 4.9: MOT-results comparing our method with CBMOT[BSZ21] (Part 1). Using CenterPoint detections (centerpoint_voxel_1440) [YZK21] from Nuscenes validation split. CBMOT* uses 2D and 3D tracklets and uses a multiplication for its score update. CBMOT^ also uses 2D and 3D tracklets. However, it uses a neural network for its score update.

Method	k_{temp}	$k_{spatial}$	TP↑	FP↓	FN↓	IDS↓
CBMOT* [BSZ21]	-	-	**81,785**	17,446	**19,614**	498
CBMOT^ [BSZ21]	-	-	81,544	17,426	19,874	**479**
ours	3	3	41,989	12,464	42,303	17,605
ours	12	6	39,188	**9,634**	40,837	21,872

Table 4.10: MOT-results comparing our method with CBMOT[BSZ21] (Part 2). Using CenterPoint detections (centerpoint_voxel_1440) [YZK21] from Nuscenes validation split. CBMOT* uses 2D and 3D tracklets and uses a multiplication for its score update. CBMOT^ also uses 2D and 3D tracklets. However, it uses a neural network for its score update.

Comparison to EagerMOT

EagerMot [KOL21] is a SOTA 3D MOT method which fuses 2D detections with 3D detections to leverage the MOT-performance using advantages of the different domains, explained in 2.3.5. In our configuration EagerMOT uses CenterPoint detections, centerpoint_voxel_1440_dcn(flip), [YZK21] as 3D detections and 2D detections from a 2D object detector called Cascade R-CNN [CV18]. For comparison purposes we also perform tracking on the same CenterPoint detections "centerpoint_voxel_1440_dcn(flip)". Our results are depicted in tables 4.11 and 4.12.
We ran our own evaluation of the EagerMot method. However, the result slightly differ from the ones presented in their paper [KOL21]. Therefore, we take both into account for our comparison. Similarly to our comparison to CBMOT, our method performs worse than EagerMOR in terms of AMOTA. We believe that this is again due to our methods inability to filter out false positive object detections.
Nevertheless, our method presents a lower number FP than EagerMOT's FP.

Methods	k_{temp}	$k_{spatial}$	AMOTA↑	AMOTP↓	RECALL↑
EagerMot*	-	-	**0.7145**	**0.5664**	**0.76163287**
EagerMot ^	-	-	0.7120	0.5690	0.75200000
Ours	3	3	0.2328	1.1631	0.45707547
Ours	12	6	0.2145	1.2831	0.44252695

Table 4.11: MOT-results comparing our method with EagerMOT[KOL21] (Part 1). Using CenterPoint detections (centerpoint_voxel_1440_dcn(flip)) [YZK21] from Nuscenes validation split. EagerMOT* represents the results evaluated by our own computation. EagerMOT^ represents the results from the paper

Methods	k_{temp}	$k_{spatial}$	TP↑	FP↓	FN↓	IDS↓
EagerMot*	-	-	**81,853**	14,467	**19,165**	**879**
EagerMot ^	-	-	-	-	-	899
Ours	3	3	44,640	13,477	39,154	18,103
Ours	12	6	39,646	**9,018**	41,034	21,217

Table 4.12: MOT-results comparing our method with EagerMOT[KOL21] (Part 2). Using CenterPoint detections (centerpoint_voxel_1440_dcn(flip)) [YZK21] from Nuscenes validation split. EagerMOT* represents the results evaluated by our own computation. EagerMOT^ represents the results from the paper

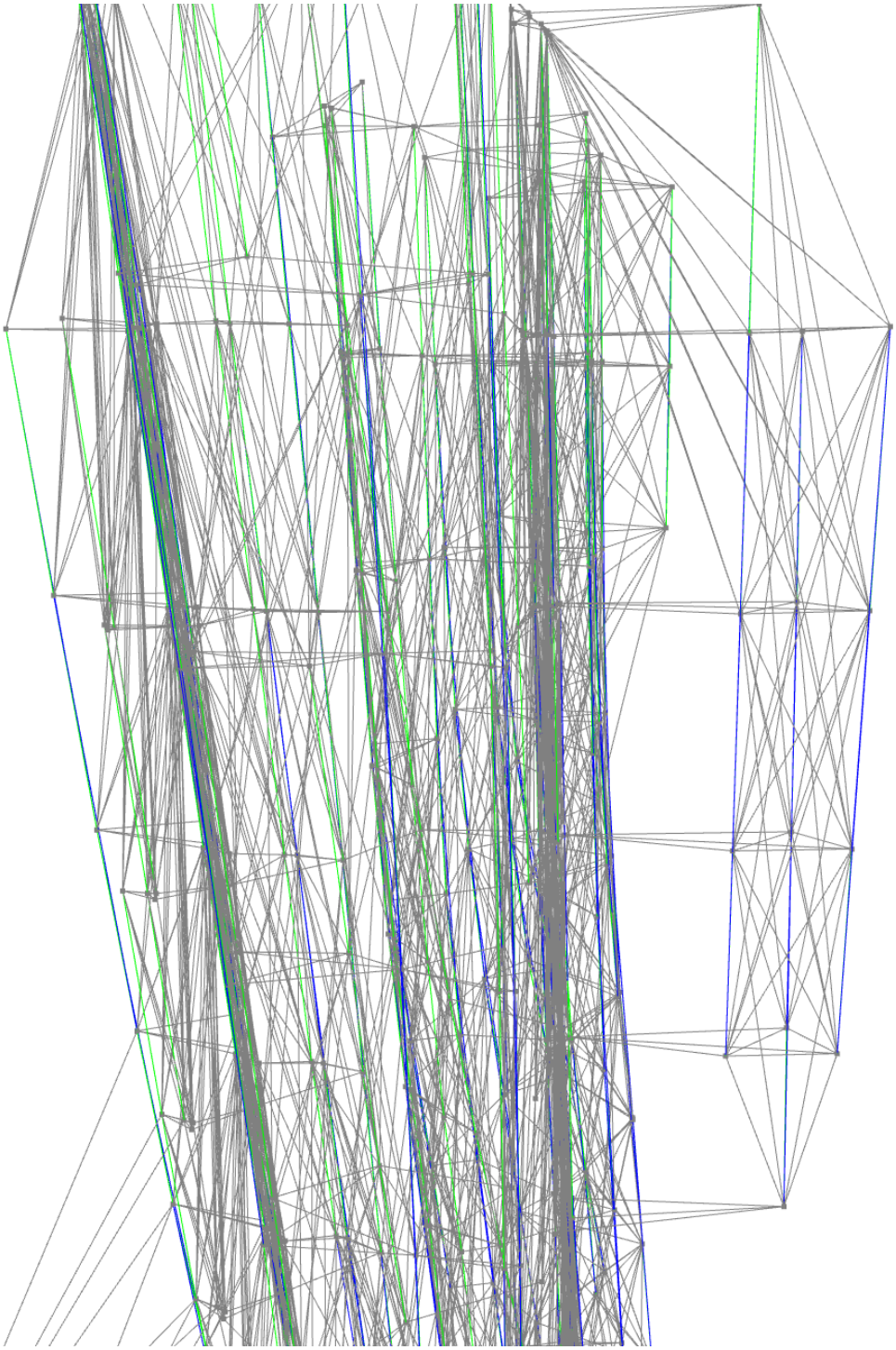

Figure 4.2: Visualization of of spatio-temporal graph after greedy rounding. Shows the high amount of False Positive active edges, which can lead to FN or IDS matches. Green edges are true positive. Red edges are false positives. Blue edges represent false negatives. Grey edges represent either true negatives or spatial edges. Computed by $N = 12$, $k_{temp} = 3$, $k_{spatial} = 3$,$\beta = 2$, $a_{shift} = 20$. Using GT-edge labels on GT-annotations .

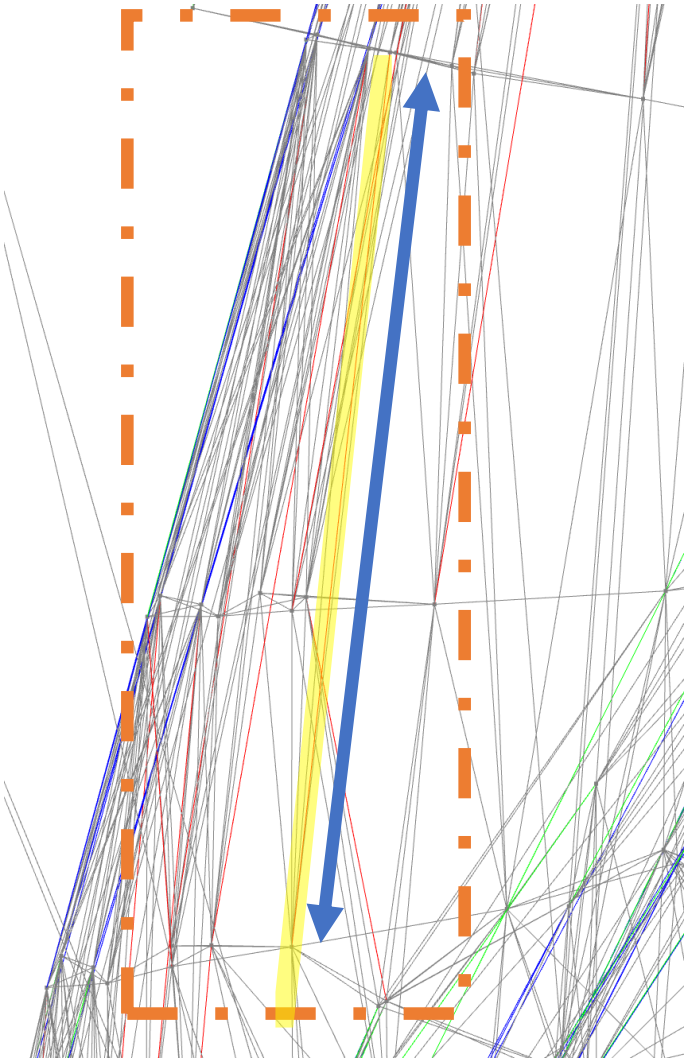

Figure 4.3: Visualization of close up of spatio-temporal graph after greedy rounding. Shows the high amount of False Positive active edges, which can lead to FN or IDS matches. Green edges are true positive. Red edges are false positives. Blue edges represent false negatives. Grey edges represent either true negatives or spatial edges. Computed by $N = 12$, $k_{temp} = 3$, $k_{spatial} = 3$,$\beta = 2$, $a_{shift} = 20$. Using GNN on GT-annotations .

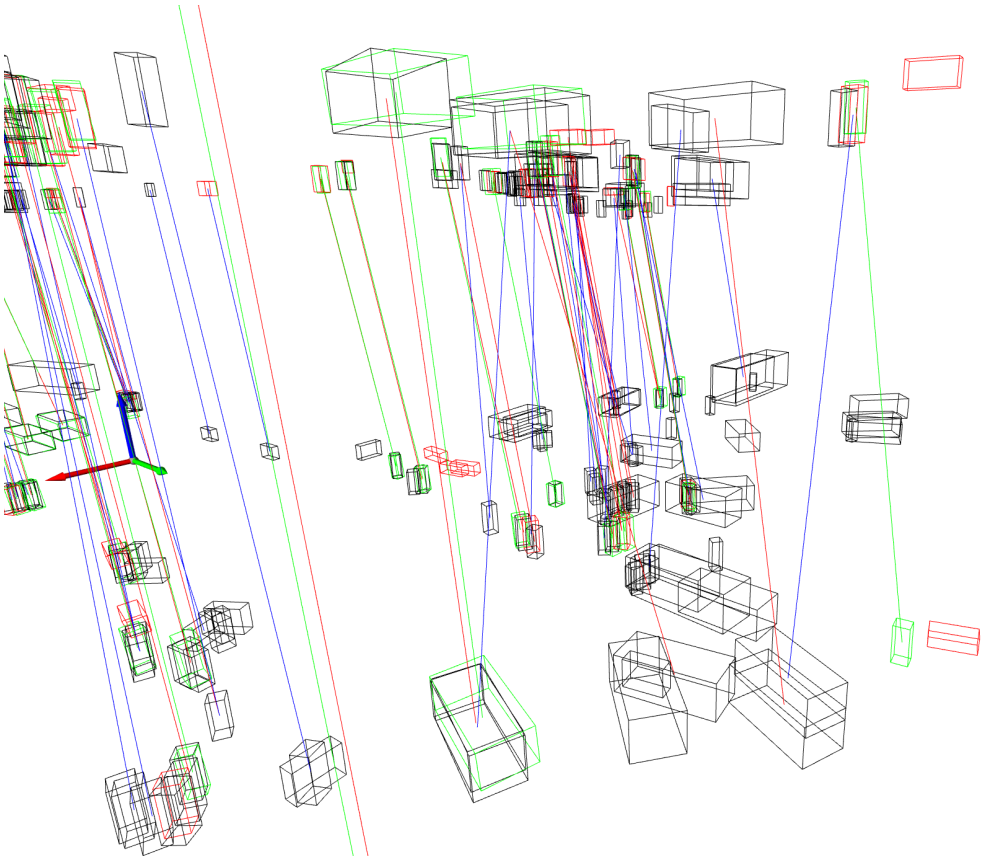

Figure 4.4: Visualization of comparison between our method and CBMOT[BSZ21]. Green represents GT annotations and associations. Red represents CBMOT detections used for association and the associations. Black Boxes represent the CenterPoint detections used by our method. Blue edges represent our method's associations. The bottom bounding boxes represent detections from the previous time frame, while the upper bounding boxes represent the current detections. It shows that there are a lot more black CenterPoint detections than green GT annotations.

Chapter 5

Conclusion and Outlook

In this chapter we would like to summarize our ideas regarding our feasibility experiment and the experiments with our learned MOT method. Lastly we would like to give an outlook to possible improvements and extensions of our method.

5.1 Feasibility

In our feasibility study in 4.2, we show that our method is able to perform MOT on a SOTA level ($AMOTA > 70\%$) under ideal conditions. Certain configurations, $k_{temp} = 12$ and $k_{spatial} = 6$, reach a AMOTA of up to 95.58%. Therefore, our MOT method shows the empirical potential to solve 3D MOT tasks. We show that the increase in the amount of temporal edges leads to an overall increase in connectivity between timeframes. Our method benefits from this higher connectivity because it increases the likelihood of connecting nodes which belong to the same GT-trajectory.
We also see that our method perform worse when increasing the number of frames per graph N. We believe that this is due to the large relative change in position between detections over time. Since we represent all objects relative to the LIDAR frame, it is possible that certain detections move faster in between the 2Hz annotations. As a result, a low number of temporal edges is unlikely to connect nodes that belong to the same GT-trajectory. We also observe that a increase of β has a minimal effect on our presented metrics. We believe that our method can only profit from a large β if we simulatneously increase the temporal connectivity between timeframes. This would mean a increase in k_{temp}. However, we also would like to note that a high β and a high k_{temp} could lead to a exponential growth of the number of edges in total $|E|$. Even though graph with total connections would be preferable for the MOT tracking problem, the computational cost also increases exponentially.

5.2 Learned Tracker

In our initial experiments based on the GT annotations, we observe that our GNN-based tracker still performs on SOTA levels. Our method reaches in certain configurations, $k_{temp} = 12$ and $k_{spatial} = 6$, an AMOTA of up to 91.95%. In addition, it yields a AMOTP as low as 0.1244. However, the low AMOTP is probably due to the high number of correct matches TP, because our detections are identical to the GT-annotations. Therefore, the $d_{i,t}$ should be constant.
Moreover, we could show that a increase in the number of spatial edges improve both AMOTA

and AMOTP, even though only on a small scale.
Nevertheless, the improvements due to higher spatial connectivity is reinforced when increasing both the number of temporal edges k_{temp} and the number of spatial edges per node $k_{spatial}$. Increasing both parameters allows our tracking method to have a higher likelihood of connecting nodes from the same GT-trajectory, while beeing able to encode and propagate spatial information over graph. This leads to a better tracking performance. Therefore, our experiment with the highest k_{temp} and $k_{spatial}$ presents the highest AMOTA and the lowest AMOTP among the experiments on GT-annotations. Furthermore, we observe a average increase in AMOTA by $\nabla = 1.398\%$ percentage points.
Nevertheless, when comparing our method to SOTA methods CBMOT[BSZ21] and EagerMOT[KOL21], we observe a significant drop in AMOTA by more than 75% relative to the same configurations running on the GT-annotations. Furthermore, the AMOTP also increases from 0.1244, (for configuration $k_{temp} = 12$ and $k_{spatial} = 6$, on GT-annotations), to 1.3185 (for CBMOT comparison) and to 1.2831 (for EagerMOT comparison). This increase is ten times as much as the same configuration running on the GT-annotations.
Concluding we can say that our method performs badly when using SOTA object detections, such as CenterPoint[YZK21] detections. The current object detections still detect a significant amount of false positives. For our method these false positive detections increase virtually the number of tracks and increase the likelihood of false positive matches. Therefore, our method is unable to perform as well as current SOTA 3D MOT methods, as long as there is not a way to filter out false positive detections.

5.3 Outlook

As mentioned above, a major issue of our method is the handling of false positive detections. Therefore, a future improvement could lie in the detection of false positives, similarly to [Zae+22]. Future work could add the task of node classification, to learn to identify false positive detections. A similar idea has been presented by [Zae+22] which also classifies nodes to identify false positive object detections. For training such a task a new augmentation method could be implemented, which randomly adds false positive detections to the spatio-temporal graph.
Future work could also lie in the increase of the association robustness. Similar in spirit to a dropout layer, future work could add an augmentation method which randomly remove nodes from the graph. As a result, the GNN would need to learn how to handle short-time occlusions.
Moreover, the graph construction only depends on the shifted translation x_i' until now. Hence, future work could include 3D appearance features or the orientation as additional features for the KNN-method, similarly to [BL20]. However, the orientation can also be harmful if the object detector estimates a significantly wrong orientation.
In addition, future work should include 3D appearance features, such as point features from PointPillars [Lan+19] or similar 3D encoding backbones, and the class identifier, to each node as initial node feature $h_i^{(0)}$. Furthermore, the initial edge features $h_{(i,j)}^{(0)}$ could contain the difference between bounding box dimensions between the connected nodes i and j.
Another possible extension of our method, would be the combination of our GNN-based tracker with a detection method. By performing detection and tracking simultaneously it would be possible to minimize the number of false positive detections. This would improve both the tracking and detection performance.

Appendix A

Model Parameters

The following tables show the Layer dimensions of each MLP used in our GNN

Edge Encoding Model	Dimensions
edge_in_dim	2
linear layer	18
linear layer	18
edge_out_dim	2

Table A.1: Dimensions for Encoding the initial edges features

Node Encoding Model	
node_in_dim	4
node_fc_dims	128
node_out_dim	4

Table A.2: Dimensions for Encoding the initial node features

Edge Feature Update Model	
edge_in_dim	2
linear	80
edge_out_dim	2

Table A.3: Dimensions of $\mathcal{N}_e$

Node Feature Update Model	
node_in_dim	4
linear	56
node_out_dim	4

Table A.4: Dimensions of $\mathcal{N}_v$

Classifier Model	
edge_in_dim	2
linear	8
edge_out_dim	1

Table A.5: Dimensions of $\mathcal{N}_e^{class}$

Appendix B

Visualizations of Spatio-Temporal Graph

Both of the figures B.1 and B.2 are related to figure 4.3.

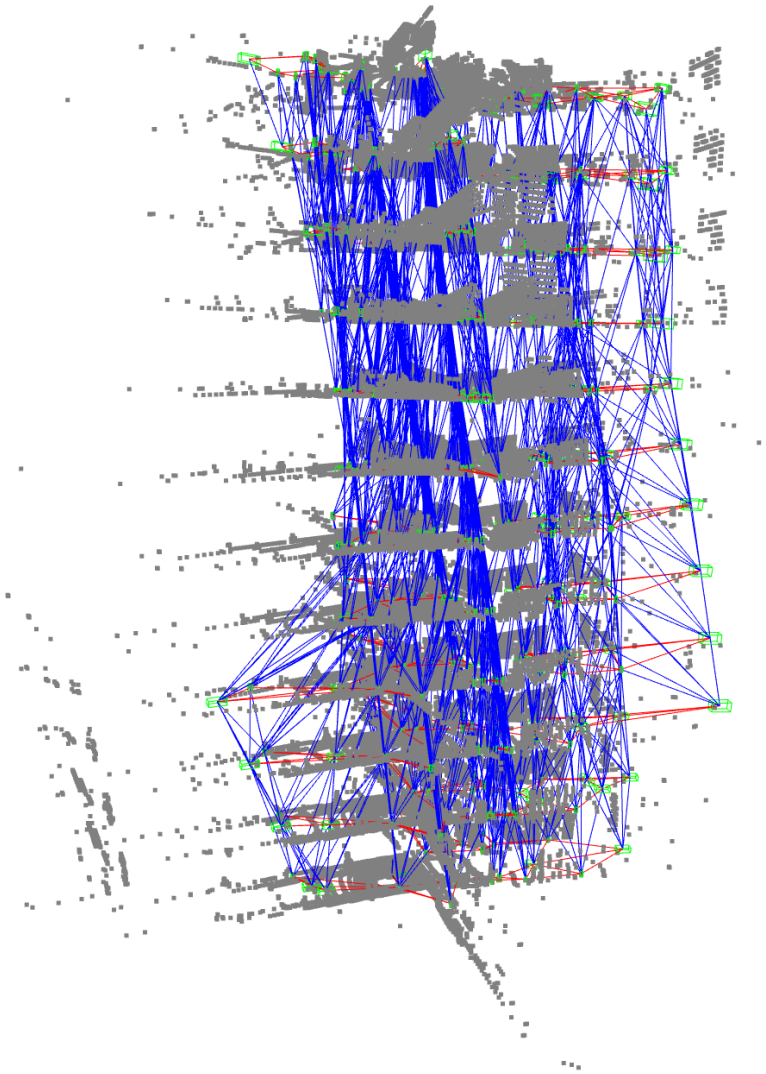

Figure B.1: Visualization of spatio-temporal graph after being build by the STGB module. $N = 12$, $k_{temp} = 3$, $k_{spatial} = 3$,$\beta = 2$, $a_{shift} = 20$. Blue edges represent temporal connections and red represent spatial connections. The green boxes are the corresponding bounding boxes of each detections o_i. The grey dots represent the point cloud, resulting from LIDAR-measurements.

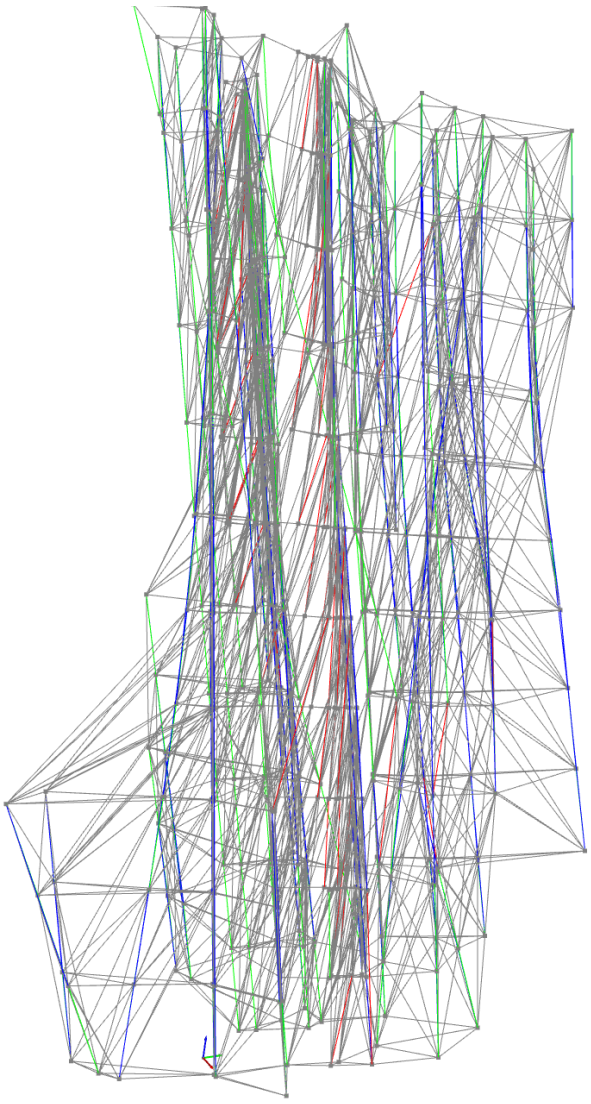

Figure B.2: Visualization of of spatio-temporal graph after greedy rounding. Shows the high amount of False Positive active edges, which can lead to FN or IDS matches. Green edges are true positive. Red edges are false positives. Blue edges represent false negatives. Grey edges represent either true negatives or spatial edges. Computed by $N = 12$, $k_{temp} = 3$, $k_{spatial} = 3$,$\beta = 2$, $a_{shift} = 20$. Using GNN on GT-annotations .

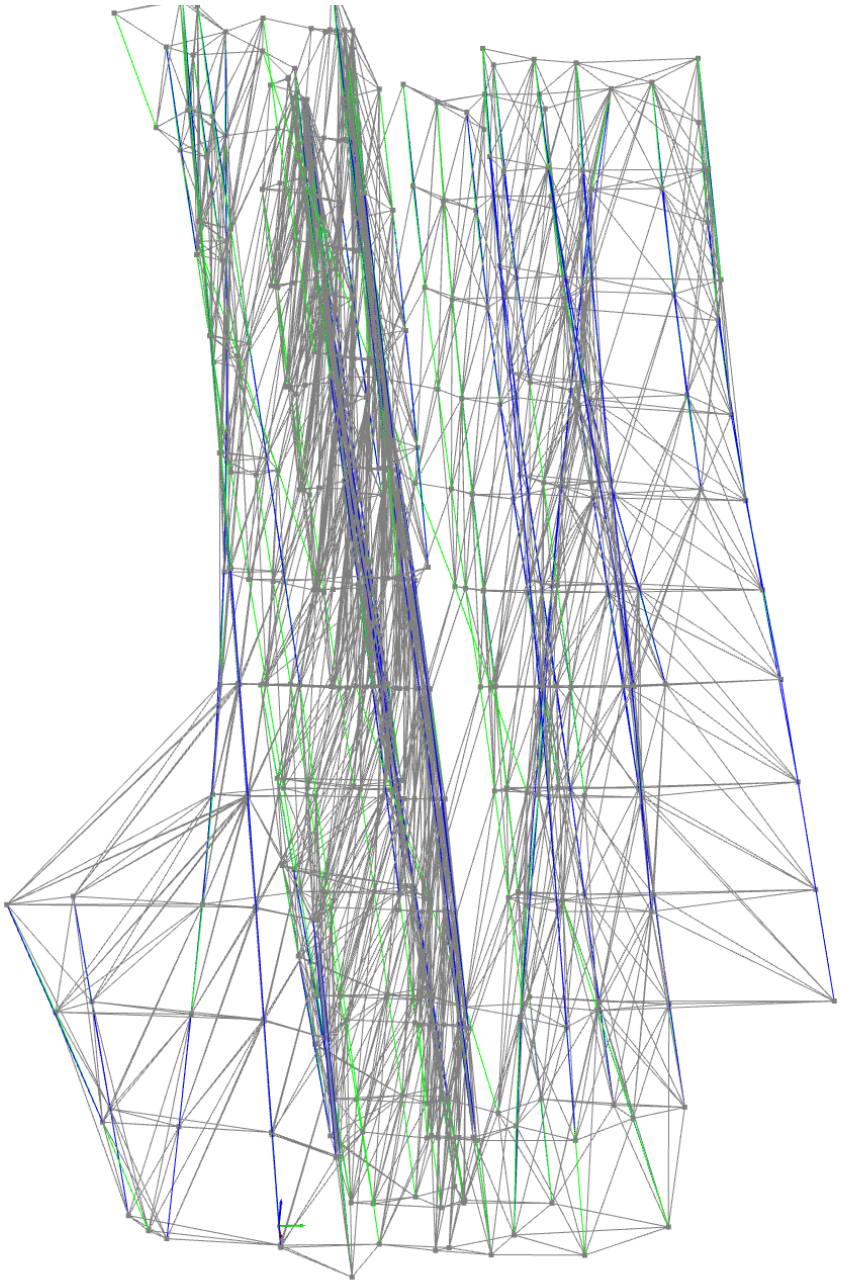

Figure B.3: Visualization of of spatio-temporal graph after greedy rounding. Shows the high amount of False Positive active edges, which can lead to FN or IDS matches. Green edges are true positive. Red edges are false positives. Blue edges represent false negatives. Grey edges represent either true negatives or spatial edges. Computed by $N = 12$, $k_{temp} = 3$, $k_{spatial} = 3$,$\beta = 2$, $a_{shift} = 20$. Using GT-edge labels on GT-annotations .

Bibliography

[AMO93] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network flows: Theory, algorithms, and applications*. 3. print. Upper Saddle River, N.J.: Prentice-Hall, 1993. ISBN: 978-0136175490.

[Bai+22] Bai, X., Hu, Z., Zhu, X., Huang, Q., Chen, Y., Fu, H., and Tai, C.-L. "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)*. 2022. URL: http://arxiv.org/pdf/2203.11496v1.

[Bat+] 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., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li Yujia, and Pascanu, R. *Relational inductive biases, deep learning, and graph networks*. URL: http://arxiv.org/pdf/1806.01261v3.

[BSZ21] Benbarka, N., Schroder, J., and Zell, A. "Score refinement for confidence-based 3D multi-object tracking". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 27.09.2021 - 01.10.2021, pp. 8083–8090. ISBN: 978-1-6654-1714-3. DOI: 10.1109/IROS51168.2021.9636032.

[Ber+11] Berclaz, J., Fleuret, F., Türetken, E., and Fua, P. "Multiple Object Tracking Using K-Shortest Paths Optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819. DOI: 10.1109/TPAMI.2011.21.

[BML19] Bergmann, P., Meinhardt, T., and Leal-Taixe, L. "Tracking without bells and whistles". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 941–951. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00103. URL: http://arxiv.org/pdf/1903.05625v3.

[BS08] Bernardin, K. and Stiefelhagen, R. "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10. ISSN: 1687-5176. DOI: 10.1155/2008/246309.

[BL20] Braso, G. and Leal-Taixe, L. "Learning a Neural Solver for Multiple Object Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[Bro+17] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42. ISSN: 1053-5888. DOI: 10.1109/MSP.2017.2693418.

[Bru+] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. *Spectral Networks and Locally Connected Networks on Graphs*. URL: http://arxiv.org/pdf/1312.6203v3.

[BCY22] Bui, K.-H. N., Cho, J., and Yi, H. “Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues”. In: *Applied Intelligence* 52.3 (2022), pp. 2763–2774. ISSN: 0924-669X. DOI: 10.1007/s10489-021-02587-w.

[Cae+20] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nuScenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. URL: http://arxiv.org/pdf/1903.11027v5.

[Cai+19] Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T.-J., Yuan, J., and Thalmann, N. M. “Exploiting Spatial-Temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 2272–2281. DOI: 10.1109/ICCV.2019.00236.

[CV18] Cai, Z. and Vasconcelos, N. “Cascade R-CNN: Delving into High Quality Object Detection”. In: *CVPR*. 2018. URL: http://arxiv.org/pdf/1712.00726v1.

[Che+15] Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., and Urtasun, R. “3D Object Proposals for Accurate Object Class Detection”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc, 2015.

[Chi+20] Chiu, H.-k., Prioletti, A., Li Jie, and Bohg, J. *Probabilistic 3D Multi-Object Tracking for Autonomous Driving*. 2020. URL: http://arxiv.org/pdf/2001.05673v1.

[Dai+21] Dai, P., Weng, R., Choi, W., Zhang, C., He, Z., and Ding, W. “Learning a Proposal Classifier for Multiple Object Tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021. URL: http://arxiv.org/pdf/2103.07889v3.

[Dan63] Dantzig, G. *Linear Programming and Extensions*. Princeton University Press, 1963. ISBN: 9781400884179. DOI: 10.1515/9781400884179.

[DBV16] Defferrard, M., Bresson, X., and Vandergheynst, P. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS’16. Red Hook, NY, USA: Curran Associates Inc, 2016, pp. 3844–3852. ISBN: 9781510838819.

[GM10] Gallicchio, C. and Micheli, A. “Graph Echo State Networks”. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 18.07.2010 - 23.07.2010, pp. 1–8. ISBN: 978-1-4244-6916-1. DOI: 10.1109/IJCNN.2010.5596796.

[Gao+21] Gao, J., Sharma, R., Qian, C., Glass, L. M., Spaeder, J., Romberg, J., Sun, J., and Xiao, C. “STAN: spatio-temporal attention network for pandemic prediction using real-world evidence”. In: *Journal of the American Medical Informatics Association : JAMIA* 28.4 (2021), pp. 733–743. DOI: 10.1093/jamia/ocaa322.

[Gil+17] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. JMLR.org, 2017. DOI: 10.5555/3305381.3305512. URL: http://arxiv.org/pdf/1704.01212v2.

[GMS05] Gori, M., Monfardini, G., and Scarselli, F. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. IEEE, 31 July-4 Aug. 2005, pp. 729–734. ISBN: 0-7803-9048-2. DOI: 10.1109/IJCNN.2005.1555942.

[He+21] He, J., Huang, Z., Wang, N., and Zhang, Z. "Learnable Graph Matching: Incorporating Graph Partitioning With Deep Feature Learning for Multiple Object Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 5299–5309.

[HS97] Hochreiter, S. and Schmidhuber, J. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[Jia+] Jiang, X., Li Peizhao, Li Yanjing, and Zhen, X. *Graph Neural Based End-to-end Data Association Framework for Online Multiple-Object Tracking*. URL: http://arxiv.org/pdf/1907.05315v1.

[KOL21] Kim, A., Ošep, A., and Leal-Taixé, L. "EagerMOT: 3D Multi-Object Tracking via Sensor Fusion". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. DOI: 10.1109/ICRA48506.2021.9562072. URL: http://arxiv.org/pdf/2104.14682v1.

[KSH12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc, 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Kuh55] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. ISSN: 00281441. DOI: 10.1002/nav.3800020109.

[Lan+19] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. "PointPillars: Fast Encoders for Object Detection From Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[LFS16] Leal-Taixé, L., Ferrer, C. C., and Schindler, K. "Learning by tracking: Siamese CNN for robust target association". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2016. DOI: 10.1109/CVPRW.2016.59. URL: http://arxiv.org/pdf/1604.07866v3.

[LGJ20] Li, J., Gao, X., and Jiang, T. "Graph Networks for Multiple Object Tracking". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 708–717. DOI: 10.1109/WACV45572.2020.9093347.

[LYS18] Li, Y., Yu, R., and Shahabi, C. "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SJiHXGWAZ.

[Mon+] Monti, F., Frasca, F., Eynard, D., Mannion, D., and Bronstein, M. M. *Fake News Detection on Social Media using Geometric Deep Learning*. URL: http://arxiv.org/pdf/1902.06673v1.

[MPS09] Moosmann, F., Pink, O., and Stiller, C. "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion". In: *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 3.06.2009 - 05.06.2009, pp. 215–220. ISBN: 978-1-4244-3503-6. DOI: 10.1109/IVS.2009.5164280.

[MS13] Moosmann, F. and Stiller, C. “Joint self-localization and tracking of generic objects in 3D range data”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, 6.05.2013 - 10.05.2013, pp. 1146–1152. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6630716.

[Pan+] Pan, X., Xia, Z., Song, S., Li, L. E., and Huang, G. *3D Object Detection with Pointformer*. URL: http://arxiv.org/pdf/2012.11409v3.

[PLW] Pang, Z., Li Zhichao, and Wang, N. *SimpleTrack: Understanding and Rethinking 3D Multi-object Tracking*. URL: http://arxiv.org/pdf/2111.09621v1.

[Ped+11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Qi+17a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. URL: http://arxiv.org/pdf/1612.00593v2.

[Qi+17b] Qi, C. R., Li Yi, Su, H., and Guibas, L. J. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc, 2017. URL: https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[RF18] Redmon, J. and Farhadi, A. *YOLOv3: An Incremental Improvement*. 2018. URL: http://arxiv.org/pdf/1804.02767v1.

[Ren+15] Ren, S., He, K., Girshick, R., and Sun, J. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc, 2015. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[SAS17] Sadeghian, A., Alahi, A., and Savarese, S. “Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 22.10.2017 - 29.10.2017, pp. 300–311. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.41.

[Sca+09] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[SSW05] Schwartz, J., Steger, A., and Weißl, A. “Fast Algorithms for Weighted Bipartite Matching”. In: *Experimental and Efficient Algorithms*. Ed. by Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., and Nikoletseas, S. E. Vol. 3503. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 476–487. ISBN: 978-3-540-25920-6.

[SWL19] Shi, S., Wang, X., and Li Hongsheng. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. URL: http://arxiv.org/pdf/1812.04244v2.

[SR20] Shi, W. and Rajkumar, R. "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. URL: http://arxiv.org/pdf/2003.01251v1.

[Sin+17] Singh, S. P., Kumar, A., Darbari, H., Singh, L., Rastogi, A., and Jain, S. "Machine translation using deep learning: An overview". In: *2017 International Conference on Computer, Communications and Electronics (Comptelix)*. IEEE, 1.07.2017 - 02.07.2017, pp. 162–167. ISBN: 978-1-5090-4708-6. DOI: 10.1109/COMPTELIX.2017.8003957.

[Sof+] Sofianos, T., Sampieri, A., Franco, L., and Galasso, F. *Space-Time-Separable Graph Convolutional Network for Pose Forecasting*. URL: http://arxiv.org/pdf/2110.04573v1.

[SS97] Sperduti, A. and Starita, A. "Supervised neural networks for the classification of structures". In: *IEEE transactions on neural networks* 8.3 (1997), pp. 714–735. DOI: 10.1109/72.572108.

[Ste89] Stephen M. Omohundro. *Five Balltree Construction Algorithms*. 1989. URL: https://www1.icsi.berkeley.edu/ftp/pub/techreports/1989/tr-89-063.pdf.

[TLT11] Teichman, A., Levinson, J., and Thrun, S. "Towards 3D object recognition via classification of arbitrary object tracks". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, 9.05.2011 - 13.05.2011, pp. 4034–4041. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5979636.

[TM17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *ICLR 2017*. Vol. abs/1609.02907. 2017.

[Voi+19] Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. "MOTS: Multi-Object Tracking and Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7942–7951.

[Wan+21a] Wang, Q., Chen, Y., Pang, Z., Wang, N., and Zhang, Z. *Immortal Tracker: Tracklet Never Dies*. 2021. URL: http://arxiv.org/pdf/2111.13672v1.

[Wan+21b] Wang, T., Xu, N., Chen, K., and Lin, W. "End-to-End Video Instance Segmentation via Spatial-Temporal Graph Neural Networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10797–10806.

[Wan+] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. *Dynamic Graph CNN for Learning on Point Clouds*. URL: http://arxiv.org/pdf/1801.07829v2.

[Wen+20a] Weng, X., Wang, J., Held, D., and Kitani, K. "AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics". In: *Computer vision – ECCV 2020 Workshops*. Ed. by Bartoli, A. and Fusiello, A. LNCS sublibrary, SL 6, Image processing, computer vision, pattern recognition, and graphics. Cham, Switzerland: Springer, 2020. ISBN: 978-3-030-66414-5. URL: http://arxiv.org/pdf/2008.08063v1.

[Wen+20b] Weng, X., Wang, Y., Man, Y., and Kitani, K. "GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with Multi-Feature Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020. ISBN: 978-1-7281-7168-5. URL: http://arxiv.org/pdf/2006.07327v1.

[Wu+21] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/tnnls.2020.2978386).

[YZK21] Yin, T., Zhou, X., and Krahenbuhl, P. "Center-based 3D Object Detection and Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11784–11793.

[Zae+22] Zaech, J.-N., Liniger, A., Dai, D., Danelljan, M., and van Gool, L. "Learnable Online Graph Representations for 3D Multi-Object Tracking". In: *IEEE Robotics and Automation Letters* (2022), p. 1. DOI: 10.1109/LRA.2022.3145952.

[Zen+] Zeng, A., Sun, X., Yang, L., Zhao, N., Liu, M., and Xu, Q. *Learning Skeletal Graph Neural Networks for Hard 3D Pose Estimation*. URL: http://arxiv.org/pdf/2108.07181v2.

[Zha+19] Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., and Loy, C. C. "Robust Multi-Modality Multi-Object Tracking". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 2365–2374. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00245.

[ZKK20] Zhou, X., Koltun, V., and Krähenbühl, P. "Tracking Objects as Points". In: *Computer Vision – ECCV 2020*. Ed. by Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. Vol. 12349. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 474–490. ISBN: 978-3-030-58547-1. DOI: 10.1007/978-3-030-58548-8{\textunderscore}28.

[ZWK] Zhou, X., Wang, D., and Krähenbühl, P. *Objects as Points*. URL: http://arxiv.org/pdf/1904.07850v2.

[Zhu+] Zhu, B., Jiang, Z., Zhou, X., Li Zeming, and Yu, G. *Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection*. URL: http://arxiv.org/pdf/1908.09492v1.