Real-time Generalized Sensor Fusion with Transformers

Aayush Ahuja, Jen Li, Shili Xu, Vivek Rane, Ashesh Jain, Balazs Kovacs
Level 5, Woven Planet Holdings Inc.
{aayush.ahuja, jen.li, shili.xu, vivek.rane,
ashesh.jain, balazs.kovacs}@woven-planet.global

Abstract

3D Multi Object Tracking (MOT) is essential for the safe deployment of self-driving vehicles. While major progress has been made in 3D object detection and machine-learned tracking approaches, real time 3D MOT remains a challenging problem in dense urban scenes. Commercial deployment of self-driving cars requires high recall and redundancy often achieved by using multiple sensor modalities. While existing approaches have been shown to work well with a fixed input modality setting, it is generally hard to reconfigure the tracking pipeline for optimal performance with changes in the input sources. In this paper, we propose a generalized learnable framework for multi-modal data association leveraging Transformers (25). Our method encodes tracks and observations as embeddings using joint attention to capture spatio-temporal context. From these embeddings, pairwise similarity scores can be computed between tracks and observations, which are then used to classify track-observation association proposals. We experimentally demonstrate that our data-driven approach achieves better performance than heuristics-based solutions on our in-house large-scale dataset and show that it is generalizable to different combinations of input modalities without any specific hand-tuning. Our approach also has real-time performance even with a large number of inputs.

1 Introduction

3D Multi Object Tracking (MOT) is a critical problem that needs to be solved to safely deploy autonomous driving systems (16; 15; 11). It is largely modeled as a tracking by detection problem, where object proposals are generated by upstream object detectors for each time step, while the tracking system links the detections across time steps and updates the states of each track (27). Although there have been significant advancements in object detection (9; 14) and machine learned tracking (17; 15; 32) in recent years, the MOT problem remains challenging for several reasons. Objects in dense urban environments often occlude each other which causes missing detections that require tracking to fill in these gaps. Moreover, object appearance changes over time making object re-identification harder. Finally, machine learned detectors don't have perfect recall, missing objects from time to time, while autonomous driving systems must have near perfect recall for safe operation. This goal can be achieved by multi-modal fusion during the tracking step to take advantage of the needed redundancy from different observation sources while maintaining a consensus representation of the agents around the ego vehicle.

The multi object tracking problem can be divided into two main stages (see (b) and (c) in Fig. 1). The first is Data Association, where we need to decide for each observation in the current time step which track it belongs to, or if it belongs to a new object that was not tracked before. The second problem is State Estimation, where we take the observations associated with each track in the current time step to update the state representation of that track.

Machine Learning for Autonomous Driving Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

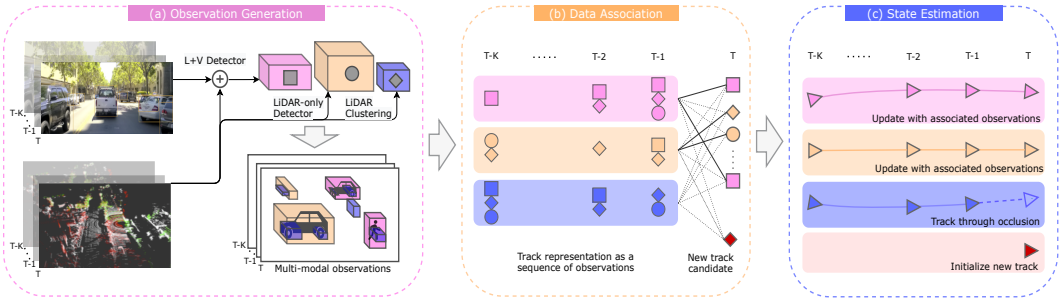

Figure 1: **Overview of our approach to the 3D MOT problem.** (a) Observation Generation process from multi-modal sensor inputs. Observations from different modalities or combination of modalities are labeled with different colors. (b) Data Association Stage. Each track (labeled with a unique color) is represented by a sequence of multi-modal observations (shown as unique shapes ■ ● ◆). (c) State Estimation Stage. ▶ represents the updated track state, while ▷ represents the track state from prediction only.

This paper focuses on solving the Data Association problem in a multi-modal setting. Our approach is inspired by recent advances in natural language processing (25) and machine learned tracking (11). We use transformers to encode tracks, represented by a collection of previously associated observations, and the observations in the current time step. To demonstrate how our method works with multiple modalities, in our experiments the observations come from multiple upstream object detectors based on a single or a combination of sensor modalities, as well as a geometry-based LiDAR pipeline. We train our model using supervised contrastive loss (12) to generate an embedding for each track and observation in a metric space where the cosine similarity of the embeddings of track - observation pairs indicates how likely they should be associated together. The final data association assignments can be formulated as a track-observation proposal classification problem based on these scores, which can be solved efficiently despite the large amount of observations from multiple modalities.

A key advantage of our approach is that it is a general framework for data association that can be easily extended to multiple modalities without changing the architecture of the model, making it simple to add new modalities and achieve the redundancy and recall needed for fully autonomous driving. Another benefit of this approach is the ability to test different sensor architectures and sensor sets, without changes to the model or methodology. Adapting to a new sensor suite requires only the data be re-annotated in order to train a new model.

In summary, our contributions are the following:

- We formulate a general framework for the data association problem using Transformers and supervised contrastive loss.
- Our method is agnostic to input modality combination and class of the tracked objects. It's designed to have real-time inference performance and runs in ~8msec for typical input sizes seen in production.
- We analyze the performance of our method through ablation studies and show that it outperforms heuristics based data association baseline which is generally used in production systems in key metrics on our large-scale in-house dataset.

2 Related Work

Online Multi-Object Tracking. Recently significant progress has been made in the multi-object tracking (MOT) area, especially in visual tracking (30; 26; 1; 29). Online 3D MOT (28; 32; 10), however, remains to be a challenging problem, especially in the context of multi-modal sensor fusion. Since most 3D MOT problems are modeled as tracking by detection, the performance is heavily affected by detection quality. At the same time, discriminative appearance modeling as well as the utilization of the spatial-temporal information are also crucial factors for the tracking quality.

Liang *et al.* (15) and Yin *et al.* (32) tackle the 3D detection and tracking problem as a whole and achieve improved tracking results by propagating observation information from raw sensor data. Both

systems, however, take the raw LiDAR spins as a single source of sensor input and would need to be heavily reconfigured to accommodate multi-model sensor inputs in the detection module.

There is a stream of work focusing on improving the discriminative power of appearance features (13; 28; 34). While good performance has been achieved, the dominance of the appearance features leaves the system error prone in situations when the input signals are sparse, e.g. under occlusion and at long distances, and makes it harder to scale and generalize to multiple sensor inputs.

Various works have been done to take advantage of the spatial context and temporal motion cues for data association (31; 20; 6; 15; 17). (15; 17) model the track temporal states using LSTMs, which are updated every cycle with associated new observations. This is error-prone when noisy associations are made, polluting the latent track representations. To capture the spatial context of observations and tracks, Hu *et al.* (10) and Poschmann *et al.* (20) model the association problem as a learned linear programming problem and a factor graph respectively. However, both use filtering based on the state estimation for track representation, which makes it infeasible to rectify incorrect associations from past time steps, and also makes their method less robust to observation noise. In contrast, we represent the track as a sequence of raw observations and then use a PointNet (22) to robustly encode the sequence and capture the motion information in latent space.

Transformers in Tracking. Transformer architecture has become dominant in many natural language processing (NLP) tasks. Due to the self-attention module that aggregates information from the entire sequence, it has achieved a clear performance edge over RNNs in processing long sequences and also gained popularity in many vision tasks, such as image generation (18), image recognition (7), object detection (2) and visual tracking (11; 26).

Hung *et al.* (11) successfully incorporated the transformer based attention mechanism in the visual MOT formulation. In contrast to their work, which operates only in the image space, we apply the transformer to multi-modal data and utilize supervised contrastive loss (12) for similarity score learning, which makes it possible for each track to be robustly associated with multiple observations from current time step. Also thanks to our PointNet-based track encoding mechanism, the final data association model is still light-weight and efficient, and achieves real-time performance even with significantly increased amount of observations from multiple modalities.

Early vs Late Sensor Fusion. Early fusion based 3D object detectors have achieved tremendous success in advancing detection quality by directly fusing image and LiDAR data together (14; 21; 3). They, however, still face difficulty in guaranteeing the recall level required by self-driving cars if used as the single source of observation. For example, a detector might miss objects unseen in the training set but a clustering based geometric LiDAR pipeline could help pick them up. On the contrary, late fusion approaches (33; 4; 5; 23) take as inputs all observations from upstream systems and exploit complementary information from different observation sources while maintaining consensus agent representations. Traditional late fusion methods (4; 5) rely on carefully hand-tuned motion models to update track states, and as a result, heavy parameter tuning and heuristics are required to adapt to different observation properties. Instead, our transformer-based data association model implicitly captures motion cues and adjusts to the modality properties directly from data.

3 Data Association with Transformers

The input to our system is a set of 3D object proposals that could come from different sources over multiple time steps, hereby called as observations $\{O_i \in \mathbb{R}^{d_o} | \forall i \in [1, N_o]\}$ where N_o is the total number of observations in the current time step. The tracker also keeps a list of live tracks over time $\{T_j | j \in [1, N_t]\}$, where N_t is the total number of tracks in the current time step. Our task is to associate new observations that arrive in the current time step to existing tracks. We refer to this problem as the data association task. Our overall approach is to learn feature embedding networks to embed each track and each observation in a common vector space with dimension d_c: $\mathbb{R}^{d_c}$. We train the embedding networks in such a way that pairwise association scores can be obtained by taking cosine similarity between the track and observation embeddings, which gives us a similarity matrix $\mathbb{R}^{N_t \times N_o}$. The next sections go into details on how we learn the embeddings for tracks and observations.

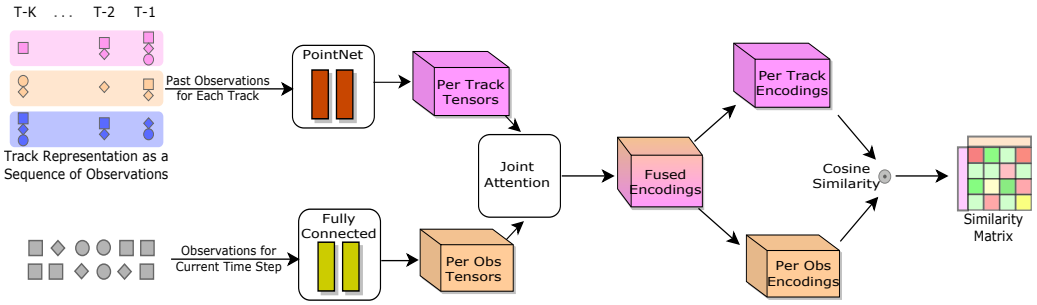

Figure 2: **Overview of our model architecture** For each track, we collate its past associated observations, and pass them through a PointNet encoder to produce a track embedding. In parallel, we collate all observations in the current time step and encode each of them with a MLP. We pass the track and observation embeddings through the joint attention module (based on self-attention layers). Finally, we extract the track and observation embeddings from the output of the joint attention module and compute cosine similarity between each track-observation pair to obtain a similarity matrix.

3.1 Observation Embeddings

As mentioned earlier, the observations could come from multiple sources. Our approach is designed to be agnostic to the properties of the input source. We could have observations from a geometric pipeline (e.g. clustered LiDAR points as obstacles) or from machine learned object detectors (e.g. (21; 3; 24)) that are trained to represent the true extent of objects. Also, the approaches generating these observations could have different degrees of precision and recall. Our embedding approach needs to cater to all of these diverse properties. We discuss the observations we use in Sec. 4.2 and show our method can work on any combination of them with high accuracy.

We represent each observation by its box and motion attributes: $\{c_x, c_y, c_z, e_x, e_y, e_z, \theta, v_x, v_y, v_z, L, \Delta t\}$ where c_i: center coordinates, e_i: box extent, θ: yaw, v_i: velocity, L: class label of the input observation and Δt is relative timestamp of the observation from the current time step. Not all input sources would generate all these attributes and we zero out entries from respective sources not providing the attribute. We learn a neural network (R_o: $\mathbb{R}^{d_o} \rightarrow \mathbb{R}^{d_c}$) which maps the input observation vectors with dimension d_o to the embedding vector space $\mathbb{R}^{d_c}$. We've experimented with different observation embedding networks as discussed in the experiments section 4.1. Our optimal architecture (in terms of accuracy vs memory/latency tradeoff) is a multilayer perceptron (MLP) style architecture.

3.2 Track Embeddings

A track is a representation of a particular object over time. Each track $\{T_t | \forall t \in [1, N_t]\}$ is represented by all the observations that are associated with it in the past K time steps: $\{O_{k,i} | \forall k \in [T-K, T-1], i \in [1, N_o^k]\}$, where N_o^k is the number of observations associated with the track at the kth time step. Using these, we want to learn a feature embedding network and obtain a vector representation of the track ($f_t \in \mathbb{R}^{d_c}$). Our embedding approach uses all the observations associated with the track in the past time steps such that it's able to learn track properties like motion and extent which require multiple time steps to obtain. We leverage a PointNet style architecture (22) to determine the track embeddings. More specifically, the input to our embedding network is a list of observation vectors $\{O_{k,i} \in \mathbb{R}^{d_o}\}$. We pass each observation independently through the same multilayer perceptron (MLP) followed by a pooling layer to aggregate the features to generate the embedding for each track.

3.3 Joint attention mechanism

To produce meaningful similarity scores, we would like objects (tracks and observations) which have similar attributes (e.g. spatial location, velocity, extent) to have similar feature embedding. Towards this goal, we adopt the self-attention mechanism from the Transformer architecture, which has been shown to capture the dependencies between input entities effectively (11; 25). In addition, since the feature embeddings for all tracks $\{f_t\}_{t=1}^{N_t}$ and all observations $\{f_o\}_{o=1}^{N_o}$ (both in $\mathbb{R}^{d_c}$) are generated independently, we stack the feature embeddings for each track and observation together before passing them through a set of self-attention layers to generate output embedding vectors f_e^{out} as follows:

$$f_e^{out} = \sum_k softmax\left(\frac{Q_e \cdot K_k^T}{\sqrt{d_k}}\right) V_k \tag{1}$$

where $Q_e = \boldsymbol{W_q} f_e$, $K_k = \boldsymbol{W_k} f_e$ and $V_k = \boldsymbol{W_v} f_e$ represent the query, key and value tensors in the self-attention formulation ((25)) and f_e is the stacked feature embeddings from the track and observation embeddings.

Intuitively, this would make the feature embedding of each observation and track dependent on all other observations and tracks. As shown in Sec. 4.6.1, the joint attention mechanism leads to significant improvement in model performance. We extract the first N_t and the last N_o vectors after the joint attention layers as the final embedding vectors for each track and observation respectively. Our final network architecture is shown in Fig 2.

Computational complexity. Our approach can handle hundreds of observations with history from multiple past time steps at $5Hz$ due to our novel encoding scheme using a combination of PointNet (22) and Transformers (25). Given the number of past observations are N_{po}, number of observations at the current time step is N_o, and number of tracks in the current time step is N_t, the computational complexity of our track embedding module is $O(N_{po})$, while that of the joint attention module is $O((N_t + N_o)^2)$. In contrast, the complexity for the approach discussed in (11) would be $O((N_{po} + N_o)^2)$, since all observations in past and current timestamp are fed to the transformer at the same time to enable soft attention. Since N_t is usually much smaller than N_{po}, our approach is more efficient and achieves real-time performance.

3.4 Data association with occlusion reasoning

Our final goal is to determine which of the observations in the new time step are associated with which existing track. Having generated the feature embeddings for the tracks and observations, we take cosine similarity between them to generate pairwise association scores leading to a similarity matrix ($\mathbb{R}^{N_t \times N_o}$).

As part of ground truth generation (details in 4.1), we have information of which observations are associated to the same track for each time step. We get on average hundreds of observations in each time step and roughly 2-10 observations associate to each track. We leverage the supervised contrastive loss (12) which has been shown to work significantly better than the conventional cross entropy loss especially in cases of large number of negative examples as in our case. The formulation for the supervised contrastive loss is the following:

$$\mathcal{L}^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(f_i \cdot f_p / \tau)}{\sum_{a \in A(i)} \exp(f_i \cdot f_a / \tau)} \tag{2}$$

where f_i denotes the embedding for the ith track, $P(i)$ are all observations associated to ith track, $A(i)$ denotes all observations (associated or not) in the vicinity of the ith track and $\tau \in \mathbb{R}^+$ is a scalar temperature parameter. We apply the above supervised contrastive loss between track-observation ($\mathcal{L}_t$) and observation-observation pairs ($\mathcal{L}_o$). $\mathcal{L}_o$ helps with creating new tracks in the current time step. For $\mathcal{L}_o$, f_i denote the ith observation in the current time step.

$$\mathcal{L}_{tot} = \mathcal{L}_t + \lambda \cdot \mathcal{L}_o \tag{3}$$

where λ is the loss balancing factor between the observation-observation and track-observation loss terms.

In each time step, it is not necessary for each observation to get associated with an existing track, for example when a new track is observed for the first time, or when there are false positive observations present. Similarly, an existing track could get occluded in the current time step and as such won't have any new observation (as shown in Fig 1). To cater to both these cases, we learn a fixed occlusion state (f_{occ}) similar to (11). The occlusion state is denoted by a all -1 vector ($\{\vec{-1}\} \in \mathbb{R}^{d_o}$) and passed through the same observation embedding network as described in 3.1 to obtain f_{occ}. For tracks occluded in the current time step, we assign its GT association to be true to the occlusion state and false with every other observation. This helps having supervision for training embeddings of occluded tracks. Similarly, observations which aren't supposed to be associated with any existing track have false GT association with all tracks. Combined with the contrastive loss, the model learns to robustly assign no new associations during occlusion, instead of enforcing wrong associations.

4 Experiments

4.1 Dataset

Ground truth generation: We used a temporal in-house dataset for our experiments. The source dataset consists of high quality human annotated 3D cuboids and is recorded at $5Hz$ on Autonomous Vehicles equipped with LiDARs, Cameras and Radars. As explained earlier, we get input observations from multiple sources in the tracking pipeline. The required ground truth for the data association task is a list of associations between these input observations and the human annotated 3D cuboids. The source of these observations could also change with time, for example release of a new version of upstream object detector or addition of a completely new input source. Hence it would be expensive to get the annotations of which observations belong to the same track every time there is a change. We instead auto-label the GT association of observations using the human annotated 3D cuboids. More specifically, we run the upstream pipelines that generate the observations and decide their association based on 3D IOU overlap with the human annotated 3D cuboids as shown in Figure 3. This also allows us to re-curate datasets easily whenever there is change in the input observations.

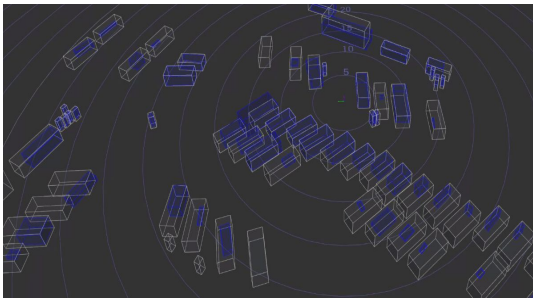

Figure 3: **Overview of our approach to GT generation**: Observations from multiple sources are generated, in this figure we show the LiDAR obstacles as an example with blue boxes. The human annotated boxes are shown in gray. We compute overlap of input observations and GT boxes to determine GT association of observations.

Our dataset consists of 52,839 scenes (with 80/10/10% train, val, test splits) and each scene has 5 time steps.

4.2 Observation Generation

In this paper, we experiment with observations from the following sources:

- **LiDAR Obstacles (LO)**: Boxes output from a LiDAR processing pipeline which clusters the LiDAR pointcloud and generates obstacles for an object (sometimes with the help of Vision segmentation results). While this pipeline has high recall and ensures almost all LiDAR points are included, it tends to over-segment and can generate multiple boxes for the same object.
- **Object Detections**: Boxes output from in-house machine learned detector using LiDAR and/or Vision inputs similar to those studied in literature (21; 3; 24). In our experiment, we included the following two detectors: one using LiDAR-only inputs (**LD**) and another using both LiDAR and Vision inputs (**L+V**). These detectors are generally trained to represent the full extent of the object. We don't assume any level of precision/recall from the upstream ML detectors.

4.3 Implementation details

We implemented the proposed architecture in PyTorch (19). All models were trained on AWS using Tesla V100-SXM2-16GB GPUs (all runs were trained on 4 instances).

We use observations from $K = 3$ past time steps to generate the track embeddings unless specified otherwise in the section. To account for the class imbalance in our dataset, we weigh each class using the following values in the loss function: $w_{car} = 1.0, w_{ped} = 5.0, w_{cyclist} = 10.0$. We use $\tau = 0.07$ in Eq 2 and $\lambda = 1.5$ in Eq 3 for the supervised contrastive loss setting. We use a neighborhood of 20m around the track's center for the $A(i)$ term in Eq 2.

4.4 Evaluation metrics

Once we have the pairwise similarity scores between tracks and observations, we compute mAP (as defined in (8)) for the per class and per modality precision-recall (PR) curves. As we threshold the similarity scores with the swept values during the PR curve generation, we additionally ensure that each observation is matched only to the highest scoring track. In Fig. 4, we visualize a handful of interesting examples and some failure cases.

4.5 Baseline method

Our approach generates a representation for each track (from past associated observations) and for all the observations in the current time step using which we perform data association. We compare this to a baseline method that is traditionally used in tracking systems where a Kalman Filter is used for track state estimation and IOU based scores between track and current frame observations are used for data association. For each track, we use the Kalman Filter (KF) framework to estimate the state from the past associated observations. We use all observation types (see Sec. 4.2) as input to the KF setup. For the track process, we use a constant velocity model and our state space is a 10-dimensional vector consisting of $\{\mathbf{c}, \mathbf{e}, \theta, \mathbf{v}\}$ where $\mathbf{c}$ is the 3D vector for the center coordinates, $\mathbf{e}$ is extent vector, $\mathbf{v}$ is velocity vector and θ is yaw. We predict the state of the KF at the timestamp of the current time step.

Next, we compute 2D Bird-eye view IOU scores between all tracks and observations in the current time step to determine a similarity matrix ($\mathbb{R}^{N_t \times N_o}$) for each track-observation pair. We use the same mAP metrics to compare to other model variations.

4.6 Results

4.6.1 Effect of Self-attention Layers

In this section, we evaluate the effect of having self-attention layers (25) in our model architecture. We have separate embedding networks for tracks and observations. Self-attention layers would allow for the embeddings to attend to all entities (track and observations) in the current time step. Table 1a shows the results of having attention layers at multiple places in the model architecture:

- **No attention layers**: This is the baseline model where we just have the MLP architecture for observation embedding network and the PointNet for track embedding.
- **Attention layers only in observation network**: We add attention layers after the MLP layers in the observation embedding network. This allows observations to attend to each other.
- **Joint Attention layers**: This is the same architecture as shown in Fig. 2.
- **Observation attention + Joint Attention layers**: We add self-attention layers in the observation network as well as after the observation and track embedding and attending to both simultaneously.

We can see that having joint layers gives a significant boost compared to having no attention layers in the network highlighting their importance. Having joint attention over track and observations is also quite important compared to just having attention layers in the observation network. Our Joint + Obs attention network roughly doubles the number of params (52M vs. 27M) and it starts to overfit leading to some dip in APs.

We also see that our models significantly outperform the baseline Kalman Filter (KF) based approach that has been traditionally used in tracking systems. KF based tracking pipelines require considerable hand tuning and customized logic for different cases to achieve optimal performance. In addition, KF

Model type	AP_{all}	AP_{car}	AP_{ped}	AP_{cyc}
Kalman Filter	75.32	72.55	93.69	80.25
No attention	94.48	97.37	87.96	77.07
Obs attention	96.02	98.69	87.90	77.22
Joint attn	**99.05**	**99.54**	**97.83**	**93.78**
Joint & obs attn	98.87	99.51	97.74	92.51

(a) **Effect of joint attention.** We compare different ways of using attention. Our Joint attention model clearly outperforms the model without attention and significantly outperforms the Kalman Filter baseline. We use all modalities (see Sec. 4.2) as inputs to the above models.

Modality	AP_{L+V}	AP_{LD}	AP_{LO}
L+V	**99.71**	N/A	N/A
L+V & LD	98.95	99.89	N/A
LD & LO	N/A	99.66	96.25
All	99.68	**99.90**	**98.96**

(b) **Evaluation on different input sources**. L+V: LiDAR+Vision based detector, LD: LiDAR only detector, and LO: LiDAR obstacles. Each row signifies the observation sources we trained on. Results are shown aggregated for all classes and we use the joint attention model (Sec. 4.6.1).

Table 1: Effect of joint attention and using different input sources

K	AP_{all}	AP_{car}	AP_{ped}	AP_{cyc}
1	99.02	99.52	97.83	93.94
2	98.92	99.52	97.73	94.66
3	**99.08**	**99.54**	**98.01**	**95.00**
4	98.85	99.49	97.49	94.85
5	98.94	99.50	97.49	94.26

(a) **AP for number of past time steps**. We don't see significant differences in aggregate AP metric.

Training set %	AP_{all}	AP_{car}	AP_{ped}	AP_{cyc}
25	98.74	99.40	96.51	89.51
50	98.56	99.34	96.76	90.59
75	98.85	99.45	97.45	91.70
100	**99.08**	**99.54**	**98.01**	**94.61**

(b) **Effect of training dataset size.** As expected, our model performs better when trained on more data, especially for under-represented object classes like cyclists.

Table 2: AP as a function of number of past time steps and training dataset size

based systems take significant effort to adapt to changes in input sources, while our framework only requires retraining a model.

4.6.2 Evaluation on different input modalities

One of the main contributions of this paper is to show the ability of our model to work with multiple input sources without much hand tuning of the parameters. In this section we explore the effect of having different combinations of input observation types. For working with different input sources, we simply need to ensure they are present in our dataset curation (see Sec. 4.1) and observation vectors are created as input into the model. No other tuning was required for optimizing performance.

We show the APs across all class labels in Table 1b (see modality details in Sec. 4.2). We see that our model performs effectively for different modality combinations w/o changes to the model architecture. We also show that training on additional modality in some cases helps boost performance on the original modality (see all modalities vs. LD + LO).

N_o	Inference (ms)
50	7.0
100	6.9
250	7.0
500	8.0
1000	12.4

(a)

K	Inference (ms)
1	8.1
2	8.2
3	8.3
4	7.9
5	8.1

(b)

Table 3: **Inference time (data association part)** by varying #observations in current time step (3a) and #past time steps used in the track embedding network (3b) keeping N_o fixed. All are measured on a single Tesla V100-SXM2-16GB GPU. The inference time increases marginally when doubling the number of observations. We don't observe notable differences varying K.

4.6.3 Effect of number of past time steps

We hypothesize that incorporating a greater number of past time steps of associated observations will give a more accurate representation of tracks. Table 2a shows a sweep across K from 1 to 5. We see similar performance across these different values for K and observe that the inference time does not significantly change as we increase K (Table 3b), giving room to increase the number of past steps if necessary. We hypothesize that a higher K will help in cases of fast moving tracks or tracks that are occluded for some time steps. We leave this exploration for future work.

4.6.4 Evaluation of training dataset size

One of the drawbacks of a heuristics based system is that scaling up the dataset size wouldn't improve its performance. It's desirable that a system improves by feeding in more training data. In this section,

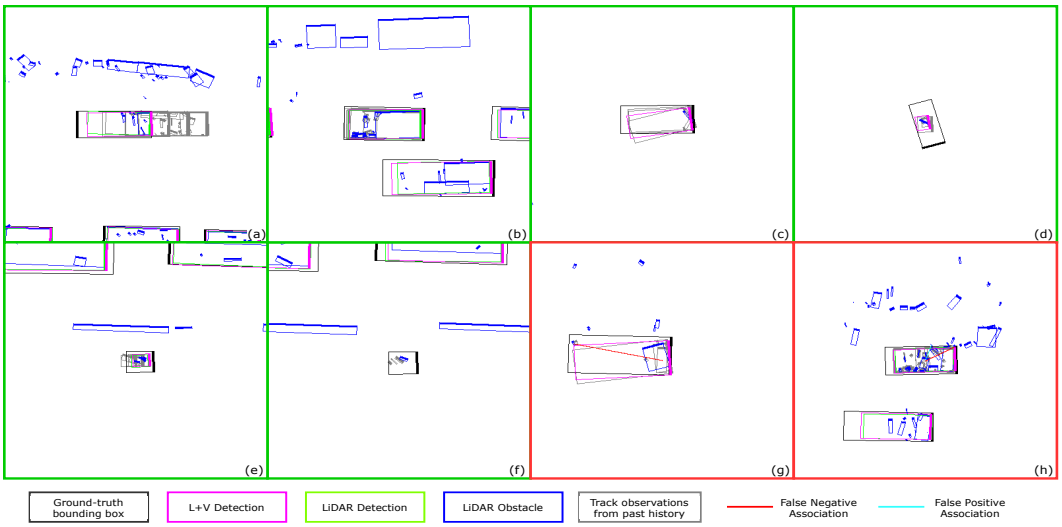

Figure 4: **Qualitative results:** Each figure shows the Bird-eye view of the track information and its surrounding obstacles. (a)-(f) show some success cases, while (g) and (h) show the typical association errors. Specifically, (a) demonstrates a scenario for moving car, (b) shows a car in dense environment, and (c) shows that correct associations can be made robustly when the track observations are noisy. In (d) and (e,f) association results for a cyclist track and pedestrian tracks are shown respectively.

we explore this question and report the model performance (Table 2b) with increasing the size of the training set. We vary the percentage of training data used while keeping the same entire test set for reporting results. We see roughly 5 AP points improvement for Cyclists increasing the data from 25% to 100% and improved results for other classes as well.

4.6.5 Real-time inference

We designed the network to have efficient inference w/o compromising on performance in diverse settings. The number of observations in the current time step could vary a lot depending on the scene and upstream observation sources. For example, a busy street could generate hundreds of observations from different road agents. In Table 3a, we analyze the inference time varying the number of observations in the current time step. We see that the latency of our approach doesn't increase significantly by doubling the number of observations and remains within reasonable bounds for real-time inference.

4.6.6 Qualitative look at success and failure cases

Figure 4 showcases a handful of example cases for our joint attention model. We visualize a variety of success cases with no errors in (a-f) and a few error cases (g-h). We can see in (f) that our model is able to reason about the track state from just the LiDAR obstacles which don't represent the extent of the Pedestrian well enough and able to associate LiDAR obstacles in current time step w/o having clear overlap with past associated observations.

We can see from the error cases that false positive associations and the missing associations are all caused by the small LiDAR obstacles at the track box boundary. This could result from ambiguity in the ownership of such small obstacles during the GT generation stage as described in Sec. 4.1.

5 Conclusion and Future Work

We presented a novel approach leveraging self-attention mechanism and supervised contrastive loss for the data association task within multi-object tracking (MOT). We showed that our generic approach is able to handle multiple input modalities without modality specific changes to the architecture. Our framework has real-time performance even with a large number of observation inputs. Finally, we demonstrated that the performance of our data-driven method scales well with increasing the amount of training data. Our model architecture can easily be extended to other problems within tracking such as state estimation which remains an area for future work.

References

[1] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6182–6191, 2019.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[4] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv preprint arXiv:2001.05673*, 2020.

[5] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[6] Peng Dai, Renliang Weng, Wongun Choi, Changshui Zhang, Zhangping He, and Wei Ding. Learning a proposal classifier for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2443–2452, 2021.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020.

[10] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399, 2019.

[11] Wei-Chih Hung, Henrik Kretzschmar, Tsung-Yi Lin, Yuning Chai, Ruichi Yu, Ming-Hsuan Yang, and Dragomir Anguelov. Soda: Multi-object tracking with soft data association. *CoRR*, abs/2008.07725, 2020.

[12] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[13] Chanho Kim, Li Fuxin, Mazen Alotaibi, and James M Rehg. Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9553–9562, 2021.

[14] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[15] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[20] Johannes Pöschmann, Tim Pfeifer, and Peter Protzel. Factor graph based 3d multi-object tracking in point clouds. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10343–10350. IEEE, 2020.

[21] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[23] Abhijeet Shenoi, Mihir Patel, JunYoung Gwak, Patrick Goebel, Amir Sadeghian, Hamid Rezatofighi, Roberto Martín-Martín, and Silvio Savarese. Jrmot: A real-time 3d multi-object tracker and a new large-scale dataset. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10335–10342. IEEE, 2020.

[24] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[26] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1571–1580, 2021.

[27] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics. *arXiv preprint arXiv:2008.08063*, 2020.

[28] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6499–6508, 2020.

[29] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12352–12361, 2021.

[30] Qiangqiang Wu, Jia Wan, and Antoni B Chan. Progressive unsupervised learning for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2993–3002, 2021.

[31] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[32] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11784–11793, 2021.

[33] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374, 2019.

[34] Linyu Zheng, Ming Tang, Yingying Chen, Guibo Zhu, Jinqiao Wang, and Hanqing Lu. Improving multiple object tracking with single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2453–2462, 2021.