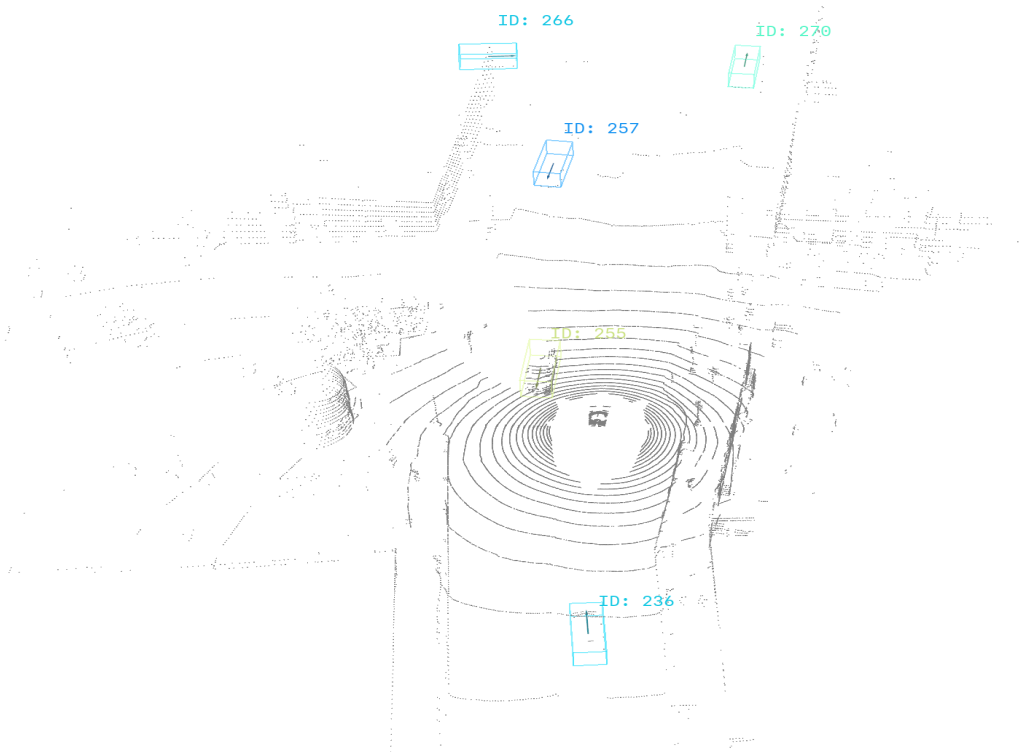


MOTPT: Multi-Object Tracking with 3D Point-based Transformer

Master's thesis in System, Control and Mechatronics

Zihao Liu & Yucong Feng

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

MOTPT: Multi-Object Tracking with 3D Point-based Transformer

Zihao Liu
Yucong Feng

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical engineering
Chalmers University of Technology
Gothenburg, Sweden 2022

MOTPT: Multi-Object Tracking with 3D Point-based Transformer
Zihao Liu
Yucong Feng

Supervisor: Georg Hess & Christoffer Petersson, Zenseact
Examiner & supervisor: Lennart Svensson, Electrical engineering

Master's Thesis 2022
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

MOTPT: Multi-Object Tracking with 3D Point-based Transformer
Zihao Liu
Yucong Feng Master's Thesis 2022
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology

Abstract

3D multi-object tracking (MOT) is dominated by the track-by-detect paradigm, and few jointly train a tracker and a detector end-to-end. For 2D, there have been multiple end-to-end tracking methods enabled by the Transformer-based detector DETR. Furthermore, recently, the effectiveness of DETR-based detectors was shown for 3D as well, such as 3DETR, a DETR-based 3D detector for indoor point clouds. Inspired by this, we create a DETR-based end-to-end multi-object tracker in 3D. We propose our end-to-end MOT method, MOTPT, by combining IA-SSD, 3DETR, and MOTR. IA-SSD is an efficient single-stage point-based 3D detector that introduces instance-aware downsampling and contextual centroid perception and can distinguish foreground and background points. MOTR is a DETR-based 2D end-to-end tracker that can capture the long-range temporal relationships among objects. We combine IA-SSD and the transformer of 3DETR as the point-based backbone of our model and then modify the MOTR to make it suitable for 3D. Finally, we combine these to obtain our 3D tracker, MOTPT. We pre-train the backbone and evaluate its detection performance on the NuScenes dataset. Based on the detection, we train the end-to-end MOTPT tracker and make inferences using the score-filter and post-matching methods. Experimental results show that MOTPT using the score-filter method leads to terrible tracking performance. In contrast, the post-matching method can simultaneously detect vehicles, predict the tracks and get better and more reasonable tracking performance.

Keywords: 3D, Multi-object tracking, End-to-end, Transformer, NuScenes.

Acknowledgements

We have gotten a great lot of help and support while writing our thesis. We appreciate Lennart, our examiner, as well as Georg and Christoffer, our supervisors, whose guidance was crucial in developing the study topics and technique. Your astute criticism encouraged us to improve our ideas, which raised the caliber of our work. Without their assistance, this thesis could not have been finished.

Zihao Liu & Yucong Feng, Gothenburg, 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

MOT	Multi-object tracking
TBD	Tracking-by-detection
JDT	Joint detection and tracking
IoU	Intersection-over-union
FPS	Frames per second
MLP	Multi-layer perceptron
SA	Set abstraction
FP	Feature propagation
CNN	Convolution neural network
DLA	Deep layer aggregation
ReID	Re-identification
CVA	Cost volume based association
MFW	Motion-guided feature warper
QIM	Query interaction module
TAN	Temporal aggregation network
RNN	Recurrent neural network
STM	Spatial-temporal module
AP	Average precision
ATE	Average translation error
ASE	Average scale error
AOE	Average orientation error
AVE	Average velocity error
AAE	Average attribute error
AMOTA	Average multi object tracking accuracy
AMOTP	Average multi object tracking precision
GT	Ground truth
MOTA	Multi-object tracking accuracy
MOTP	Multi-object tracking precision
MT	Number of mostly tracked trajectories
ML	Number of mostly lost trajectories

FAF	The average number of false alarms per frame
TP	Number of true positive
FP	Number of false positives
FN	Number of false negatives
IDS	Number of identity switches
FRAG	Number of track fragmentations
LGD	Average longest gap duration in seconds

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

t	Indices for point cloud frames

Parameters

τ_n	The ground truth filter threshold
λ_{sam}	The weight for the down-sample strategy loss
λ_{cent}	The weight for the centroid prediction loss
λ_{cls}	The weight for centroid prediction loss
λ_t	The weight for track score loss
λ_{loc}	The weight for location loss in the box prediction loss
λ_{size}	The weight for size loss in the box prediction loss
λ_{bin}	The weight for angle-bin loss in the box prediction loss
λ_{res}	The weight for angle-residual loss in the box prediction loss
τ_{active}	Active track threshold
τ_{new}	New-born object score threshold
τ_{dis}	Disappear object score threshold
τ_{ex}	Exit object threshold
τ_{match}	Hungarian match distance threshold
N	The number of encoder layers
M	The number of decoder layers

Variables

pc^t	The tth frame of the point cloud sequence
q^t_{det}	The detect query of frame t
q^t_{otr}	The original track query of frame t
q^t_{tr}	The updated track query of frame t
C_{det}	The center of detection bounding box
C_{tr}	The center of tracked bounding box
d_{det}	Detect query offset
d_{tr}	Track query offset

Operations

PE	The Fourier positional embedding
Concat	Concatenate
MLP	Multi-layer perceptron
FC	Fully connected

Contents

List of Figures

List of Tables

1
Introduction

Recently, more and more research on applying the Transformer to 2D multi-object tracking (MOT) has been published, mostly achieving state-of-the-art (SOTA) performance. In contrast, using the Transformer in 3D LiDAR MOT is still rare. The Transformer has the advantage of capturing global and long-term relationships. Therefore, combining the Transformer with 3D LiDAR MOT is a promising topic for further research.

In this thesis, we aim to build a 3D Transformer-based multi-object tracker LiDAR point cloud, train it in an end-to-end fashion, and evaluate the tracking performance on the NuScenes dataset.

1.1 Background

Autonomous driving technology is a popular field currently. The popularization of autonomous driving technology has the potential to significantly improve traffic safety and efficiency, reduce energy consumption and emissions, and profoundly change our future traffic patterns. In order to achieve assisted autonomous driving or fully autonomous driving, the perception system is an important part, especially the LiDAR sensor, which is one of the most necessary and widely used sensors nowadays. 3D object tracking for LiDAR point cloud has seen rapid development in autonomous vehicles in recent years.

MOT importance. The popularization of autonomous driving technology has the potential to significantly improve traffic safety, increase transport efficiency, reduce energy consumption and emissions, and profoundly change our future transport patterns. In autonomous driving, MOT is a critical computer vision task, and a long-standing topic [23]. The MOT tasks require detecting objects in the continuous image or point cloud frames and assigning each object a class label and tracking ID without prior knowledge (shape or quantity). MOT plays an essential role in tasks with a temporal aspect since it correlates detections over time to compensate for the instability of detection and improve the accuracy of perception.

MOT difficulties. However, MOT tasks remain genuinely challenging, and several difficulties need to be addressed

- Datasets are rare and labelling is difficult.
- Detection is not accurate enough.

- Various occlusions and interactions between objects.
- The motion model of the object is uncertain
- The real-time inference speed is not fast enough.

First, datasets are rare, and labeling is complex. The labeling difficulty comes from the enormous quantity of information. In a scene, there are many kinds of objects such as pedestrians, cyclists, vehicles, animals, facilities, etc. Second, detections are not stable and accurate enough. This leads to missing and false detections, leading to poor tracking results. The third and fourth difficulties are the reasons for the unstable and inaccurate detection. Every object has different motion patterns (direction, speed) in a video or point cloud sequence. Besides, there are always situations where the object is occluded by other objects or occludes other objects. This makes it difficult for the system to learn the movement patterns of all objects and to detect occluded objects. Last, due to the complexity of the network and the massive volume of data, it is difficult to increase the speed of MOT in practice.

1.2 Purpose and method

2D MOT has been extensively explored and achieved good performance. Especially after the advent of DETR [6], more and more methods are being developed to use the concept of DETR to complete 2D MOT with transformers in an end-to-end fashion, such as TransTrack [36], Trackformer [26] and MOTR [51]. There is also another extension of DETR, 3DETR [27], an indoor object detection model for the 3D point cloud. However, it is uncommon to use Transformer-based end-to-end learning for 3D MOT. So the purpose of our thesis is to extend the DETR-based 2D MOT to its 3D counterpart and to achieve a reasonable tracking result.

To achieve this purpose, we implemented DETR and these DETR-based methods. In addition, we did research on object detection methods like PointNet [30], PointNet++ [31], 3D-SSD [22], and IA-SSD [52] for the 3D LiDAR point cloud to find a backbone for our own model.

Pointnet [30] works similarly to the classifier and processes the raw point cloud directly. It has two main operations: multi-layer perceptron (MLP) to extract point features and max-pooling to obtain global features. The object classification network uses global features as input, while the segmentation network uses both global features and point features. Since its network directly pools all points maximally into a global feature, local point-to-point connections are not learned by the network. Therefore, PointNet [30] has limited ability in both detail processing and generalization to complex scenes. To address this problem, PointNet++ [31] uses a hierarchical point set feature learning framework consisting of multiple set abstraction (SA) layers to perform multi-level feature learning on local areas, much like a pyramid feature aggregation scheme. Each SA layer consists of a sampling layer, a grouping layer, and a PointNet layer (the details will be shown in Sec. 2.3.1). Though PointNet++ can capture the local features of the point cloud, it can not learn the spatial distribution of the point cloud.

3D-SSD [22] finds that the feature propagation (FP) layer and refinement layer are the bottlenecks of the point-based detector. It proposes a new grouping method that considers the similarity between points in the geometric domain and feature domain. Through this improved grouping method, the output of the SA layer can be directly used to generate object proposals, avoiding the large amount of calculation brought by the FP layer. At the same time, to avoid region pooling in the refinement, 3D-SSD directly adopts the representative points output by SA, uses the improved grouping algorithm mentioned above to find its neighborhood points, and uses a simple MLP to predict the category and object frame 3D bounding box. IA-SSD [52] is one of the latest single-stage point-based detectors. It uses class-aware and centroid-aware sampling strategies to preserve the foreground points. In addition, it provides a contextual instance centroid perception, which allows instance center regression to fully leverage the contextual information around bounding boxes.

Among DETR-based trackers, MOTR has the best ability to identify new-born and exit objects, and to associate long-term spatial-temporal relationships between objects. Besides, considering the sparsity of foreground points in the outdoor point cloud, the downsampling strategies of IA-SSD are effective in helping our model distinguish between foreground and background points. Therefore, we decided to combine IA-SSD and the transformer of 3DETR as the backbone of our model. We modified MOTR and combined the backbone with MOTR to build our own end-to-end 3D MOT model, MOTPT, for the 3D LiDAR point cloud. For inference, we used score-filter and post-matching 2 methods, the outcome of the experiment demonstrates that MOTPT using post-matching can detect the vehicles and predict the track simultaneously, and it achieved a reasonable tracking performance.

1.3 Limitations

For our model MOTPT, there are two limitations:

- Only track the vehicles.
- Track a maximum number of objects.

For the first one, the method mainly focuses on tracking the vehicles and ignores the pedestrian and bicyclist trajectory because the point cloud of the vehicle is much more obvious and that is much more important in the traffic. On the contrary, the point cloud of pedestrians and bicyclists is sparse and random in their movement, so it is difficult to predict and track their trajectories.

The other one comes from the object query of DETR, since the number of the object queries which are represented by learnable positional embedding is fixed, and one object query only corresponds to one object. In other words, if the number of objects is larger than that of the object queries in a scene, then there would be some wrong tracks.

2

Theory

This chapter presents the theoretical knowledge needed to understand this work, including the theory of MOT, some components in deep machine learning, detection, and tracking, and several latest methods proposed by other authors used in our model, NuScenes dataset and its tracking evaluation.

2.1 Multi-object tracking

MOT aims to find moving objects in an image (2D) or a point cloud (3D) sequence, recognize them in different frames, and give them accurate class labels and IDs. These objects can be anything: pedestrians, cycles, vehicles, various animals, etc. Shown as Fig. 2.1, MOT methods can be divided into 2 paradigms: Tracking-by-detection (TBD) [2, 17, 25, 44, 45, 46, 49, 53] and joint detection and tracking (JDT) [16, 29, 39, 43, 47, 48, 51, 54, 55].

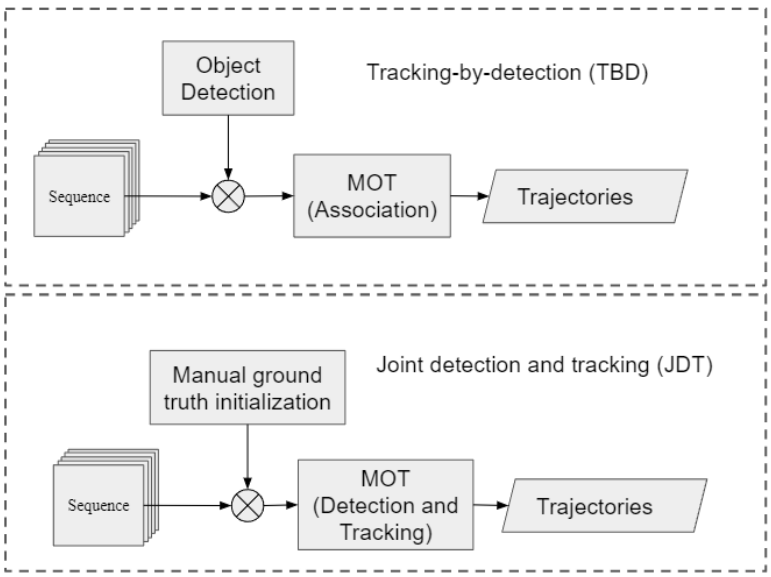


Figure 2.1: Two MOT paradigms: Tracking-by-detection (TBD and joint detection and tracking (JDT).

2.1.1 Tracking-by-detection (TBD)

TBD is the most commonly used paradigm in MOT. Generally, the TBD paradigm decouples tracking into two steps: detection and association. First, detect the objects in every frame, then associate object detections across frames to form the trajectories of all objects and judge which objects are the new-born and exit objects. The association step is solved by two mainstream methods: model-based [17, 44, 45, 53] and model-free [2, 49, 25]. For the former, motion is the association cue. The Kalman filter is used to predict the next frame location of the objects by leveraging the motion law of objects, and the Hungarian algorithm or Greedy algorithm based on different matching criteria (intersection-over-union (IoU), Euclidean distance or Mahalanobis distance) is used to match the detections of objects and the predictions. For the other kind of method, the neural network is used to extract the appearance features of the objects and then use them to match objects. Association is typically expressed as a bipartite matching problem. As a result, most of the present research focuses on improving the affinity matrix between tracked objects and new detections.

SORT [45] and DeepSORT [46] are two representative classical methods. The core of SORT is the Kalman filter and Hungarian matching algorithm. The Kalman filter predicts the bounding boxes according to the previous tracks. The IoU of prediction and detection is input into the Hungarian algorithm for linear allocation to associate the object ID across frames. The problem is that once the object is occluded or not detected for some other reason, the state information predicted by the Kalman filter will not match the detection results, and the tracking segment will end in advance. After occlusion, the object detection may be continued, so SORT can only assign a new ID number to the object, representing the beginning of a new tracking segment. Thus, the disadvantage of SORT is that it is greatly affected by occlusion, and there will be a lot of ID switching. The main idea of DeepSORT is the same as SORT. To solve the above problem, a simple convolution neural network (CNN) is used to extract the appearance features of the detected object, and these figures are saved. In each subsequent step, the similarity calculation between the appearance features of the detected object in the current frame and the previously stored appearance features shall be performed. This similarity is used as an important criterion. Compared with SORT, the ID switching of DeepSORT decreases by 45%.

The TBD paradigm has been extensively explored and achieves good performance thanks to improved and advanced detection methods. However, the heuristic matching step in the association module of the methods following the TBD paradigm inevitably requires many hand-crafted rules. In addition, since TBD has two steps and the performance highly relies on the detection quality, it leads to sub-optimal problems in terms of efficiency and accuracy.

2.1.2 Joint detection and tracking (JDT)

JDT paradigm is becoming increasingly popular because of the development of deep machine learning. As a single-stage paradigm, the methods following JDT use a neural network or transformer [40] to detect objects and assign them class labels or

IDs in one step.

CenterTrack [55] is a method following JDT paradigm. It uses a CenterNet [8] detector to locate object centers, setting the detector on two adjacent frames and a heatmap of the prior tracks represented by points. The detector also outputs an offset vector from the center of the objects in the current frame to that in the previous frame. A greedy algorithm is leveraged to complete the association by relying on the distance between the predicted offset and the detected center point in the previous frame. However, CenterTrack only associates the detection between two adjacent frames, so forming long-term associations and dependence is difficult. As a result, it has a high level of ID switch. In addition to extracting features layer by layer, FairMOT [54] adds a layer of deep layer aggregation (DLA) to merge the features of each layer. Then it carries out the detection and Re-identification (ReID) branch two tasks in parallel and combines the loss of the two branches for balanced optimization to achieve the optimal accuracy and speed results. However, ReID tracking loss is incompatible with the detection loss in the same network, and even the tracking loss will damage the performance of the detection task to a certain extent. This happens because the ReID task focuses on intra-class differences, while detection aims to amplify inter-class differences and minimize intra-class differences. For example, for pedestrian tracking, the ReID task expects to maximize the difference between each person, while the detection task expects to maximize the difference in goals between different categories. It expects the pedestrian category to be as close as possible in the high-level semantic space.

TraDeS [47] designed the Cost Volume based Association (CVA) module and the Motion-guided Feature Warper (MFW) module. The CVA module extracts ReID embedding features point by point through the backbone to construct a cost volume, which stores the matching similarity between the embedding pairs between two frames. Then, the model can infer the tracking offset according to the cost volume, the spatial-temporal offset of all points, and obtain the potential object center between two frames. Tracking offsets and embeddings are used together to construct a simple long-range data association. After that, the MFW module uses the tracking offset as a motion cue to transfer the features of the object from the previous frame to the current frame. Finally, the features from the previous frame and the features of the current frame are aggregated to perform the detection and segmentation tasks of the current frame. In the CVA module, the cost volume is used to supervise the ReID embedding, and different object categories and background regions are implicitly considered. That is, the ReID branch considers the inter-class differences. Therefore, it can learn not only effective embeddings but also be compatible with detection losses without compromising detection performance. In addition, since the tracking offset is obtained based on appearance feature similarity, it can match a large moving target or work at a low frame rate. Therefore, this predicted tracking offset can serve as a robust motion cue to guide feature propagation in the MFW module. Objects occluded or blurred in the current frame may be clear in the previous frame, so through the MFW module, the previous frame feature propagation can support the current frame features to recover potential invisible objects.

Unlike above JDE methods, TransTrack [36], Trackformer [26] and MOTR [51] are three transformer-based MOT using attention mechanism. They all use the object query concept originally proposed in DETR [6] (this method will be introduced in Sec. 2.4.1), which provides a representation of new-born objects. In TransTrack [36], authors proposed another concept "track query" to maintain objects information and their trajectories in the previous frames. It has two decoder blocks. One is responsible for detection and generating detection bounding boxes, and the other is responsible for target propagation and generating tracking bounding boxes. Then the tracking boxes and detection boxes will be matched by using the Hungarian algorithm based on IoU. The matched boxes are the objects from the track to the previous frame, and the other boxes are the new objects in the current frame. TrackFormer [26] is similar to the TransTrack. It employs a Transformer encoder-decoder architecture, which uses the self-attention mechanism to encode image features from a CNN and the self- and cross-attention mechanisms to decode queries into bounding boxes associated with identities in every frame. Each query represents an object and follows it autoregressively in space and time. When new objects arrive, they are detected by static object queries, which are then turned into subsequent track queries. Compared to TrackFormer, MOTR [51] has an additional query interaction module (QIM) consisting object entrance and exit mechanism and a temporal aggregation network (TAN) that can provide contextual relation information to the tracked objects and enhance temporal relation. We used MOTR to design our own model, so more detail will be shown in Sec. 2.4.4.

Compared to the methods following the TBD paradigm, these methods do not need data association and post-processing operations such as NMS, so they are more efficient and have lower computation complexity.

2.2 Deep machine learning components

2.2.1 Multi-layer Perceptron (MLP)

MLP is one of the most basic deep learning models with a forwarding structure based on the biological neuron model, including an input layer, an output layer, and multiple hidden layers. The neural network diagram of a three-layer perceptron is shown in Fig. 2.2.

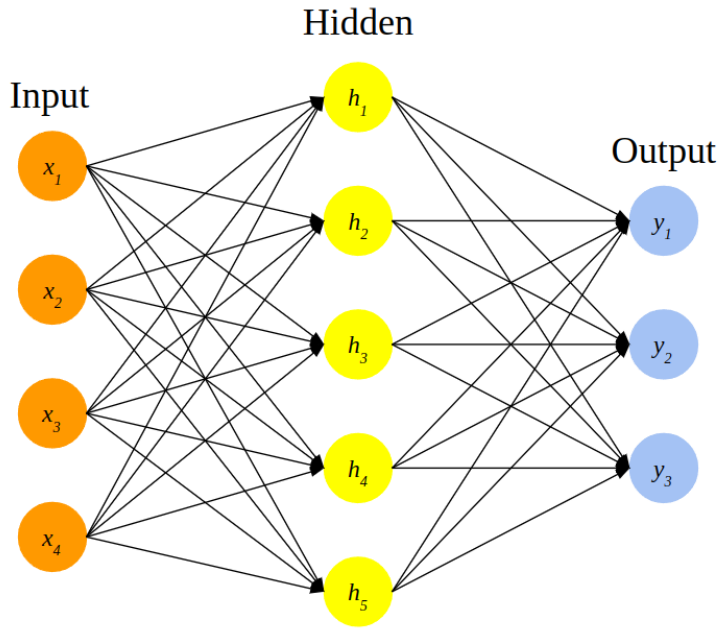


Figure 2.2: The network diagram of a three-layer perceptron.

The calculation of each neuron in hidden layers or output layers is shown in Fig. 2.3. An MLP consists of three basic elements: weight, bias, and activation function. The connection strength between neurons is represented by the weight, which represents the importance of the corresponding input. Bias is a learnable parameter set to minimize a loss function. Besides, it is an essential parameter in the model to ensure that the output value calculated through input cannot be activated arbitrarily. The activation function acts as a nonlinear mapping, limiting the output amplitude of neurons to a specific range, generally $(-1, 1)$ or $(0, 1)$.

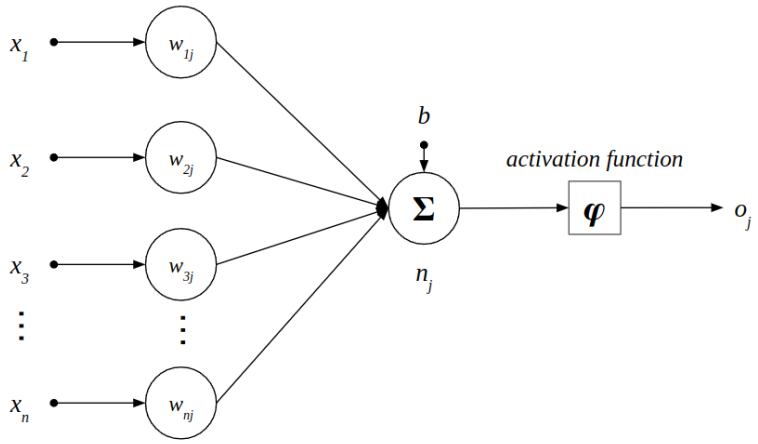


Figure 2.3: The calculation of each neuron in hidden layers or output layer, where x_{1-n} represent the input of the node n_j, w_{ij} is the weight between input x_i and node n_j, b represents the bias of n_j, and o_j is output from activation function.

Assuming that there are n input with m features $X \in \mathbb{R}^{n \times m}$ and the MLP has only

one hidden layer consisting h neurons, the weight and bias of the hidden layer can be expressed as $W_h \in \mathbb{R}^{m\times h}$ and $b_h \in \mathbb{R}^{1\times h}$. If there are q output, the weight, and bias of the output layer are $W_o \in \mathbb{R}^{h\times q}$ and $b_o \in \mathbb{R}^{1\times q}$ respectively. Then the output of the hidden layer and output layer can be calculated by the following formula

$$\begin{aligned} H &= f(XW_h + b_h) \\ Y &= f(HW_o + b_o) \end{aligned} \tag{2.1}$$

where $f(.)$ is activation function. The input of one neuron passes through the nonlinear activation function to the next neuron, and after the activation of the neuron in this layer, it continues to pass down until the output layer. It is precise because of the repeated superposition of these nonlinear functions that the neural network has enough capacity to grasp complex patterns. Sigmoid [11], Tanh [35], ReLU [4], Leaky ReLU [24], PReLU [12] and GeLU [13] are several common used activation functions.

2.2.2 Transformer

The Transformer [40], which abandons the idea of traditional CNN and RNN (Recurrent neural network), has become popular in multiple fields, especially natural language processing (NLP) and CV. The Transformer consists of self-attention and feed-forward neural network. The reason for adopting the attention mechanism is that the calculation of RNN (or LSTM [14], GRU [7], etc.) is limited to sequential, that is, RNN-related algorithms can only be calculated from left to right or from right to left. This mechanism brings three problems:

- The calculation of time t depends on the calculation result of $t-1$ time, which limits the parallel ability of the model
- Information will be lost in the process of sequential calculation. Although the structure of gate mechanisms such as LSTM alleviates the problem of long-term dependence to a certain extent, LSTM is still powerless for the phenomenon of special long-term dependence.
- Exploding and or vanishing gradients

The proposal for the Transformer solves the above three problems. The attention mechanisms of the Transformer can provide context information for any position in the input sequence, and it can process all tokens in the sequence at the same time. Therefore, the Transformer has better parallel computing ability than RNN, in line with existing GPU frameworks. For example, if use Transformer to do the language translation, it considers the input sentence to be a sequence of words and processes them at the same time to output the translation result. As shown in Fig. 2.4 and Fig. 2.5, the Transformer consists of encoder and decoder two components, and they are formed by identical encoder layers and decoder layers, respectively.

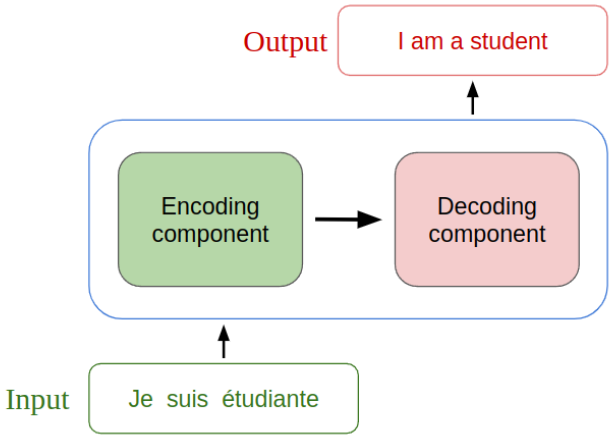


Figure 2.4: A simple example of using transformer to translate language.

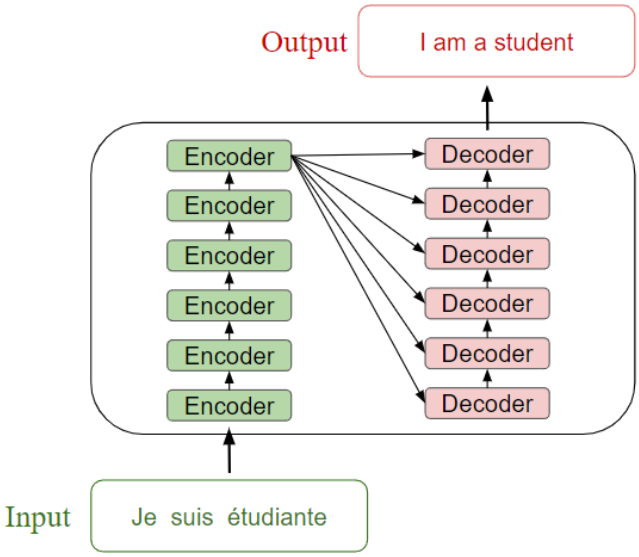


Figure 2.5: The encoding component and decoding component of Transformer consist of several identical encoder layers and decoder layers, respectively.

Encoder-decoder structure. The encoder contains several identical encoder layers that process the input data sequence layer by layer, while the decoder contains several identical decoder layers that do the same thing to the output of the encoder. The encoder layer aims to encode the input to the encoding that contains the information about relationship between all tokens in the input. Then the encoding is fed into next encoder layer. Contrary to encoder layer, each decoder layer converts encoding to the output sequence.

The structure of the encoder layer and decoder layer is shown in Fig. 2.6. Each encoder layer contains two main elements: self-attention mechanism and feed forward neural network. The self-attention mechanism takes encoding from the previous

encoder layer as input and evaluates their interrelationships to create output encoding. Each output encoding is then further processed by the feed-forward neural network and sent to the next encoder layer and decoder. The first encoder layer takes embedding of the input sequence and positional embedding as input instead of encoding. Compared to the encoder layer, the decoder layer has an additional attention mechanism over the encoding to draw the relevant information from the encoder.

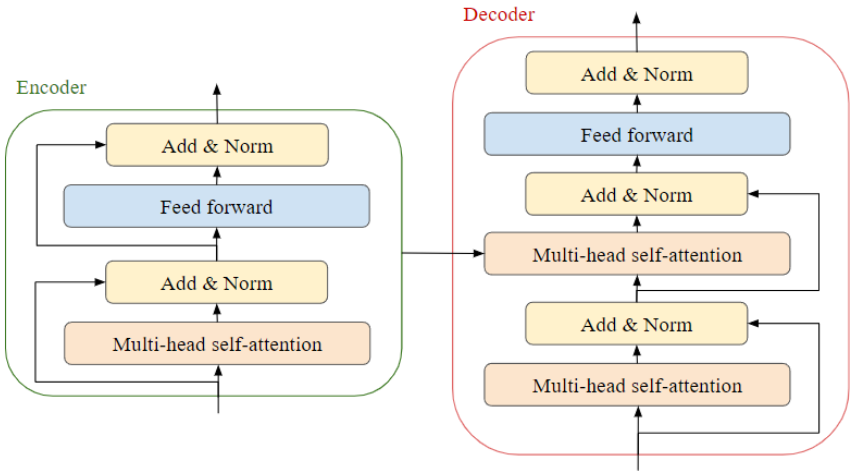


Figure 2.6: The structure of encoder and decoder.

Position embedding. In the RNN, the sentence is divided into words and sent to the network in turn, so that the input of each sentence has its own text order. But the transformer structure feeds the words in all positions together at the same time as a sequence into the network, which loses the order of the text. So position embedding is needed to solve this problem.

The most commonly used position encoding method is sinusoidal position embedding

$$\begin{aligned} PE(pos, 2i) &= sin(\frac{pos}{10000^{2i/d}}) \\ PE(pos, 2i+1) &= cos(\frac{pos}{10000^{2i/d}}) \end{aligned} \tag{2.2}$$

where pos represents the position of the word in the sequence, d represents the size or the dimension of word at the position pos, and i refers to the each individual dimension of the word embedding. Fig. 2.7 shows an simple example, for the first word position embedding P_0, $pos = 0$, $d = 5$, which means it have five individual dimensions of the embedding $(i = 0, 1, 2, 3, 4)$.

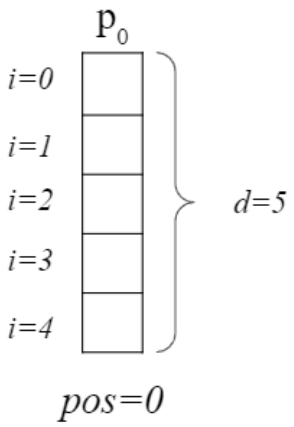


Figure 2.7: An simple example to illustrate pos, i and d.

However, inspired by [51, 6, 27], we use the Fourier positional embedding [37] in our project, which generates the best performance between the underfitting (low scale, recovery function frequency too low) to overfitting (high scale, recovery function frequency too high). For a point $v \in \mathbb{R}^d$, the Fourier positional embedding $\gamma(v)$ is defined as

$$\gamma(v) = [\cos(2\pi Bv), \sin(2\pi Bv)]^\top \tag{2.3}$$

where $B \in \mathbb{R}^{m \times d}$ is a random Gaussian matrix, mapping the input point to the high dimension m, and all entries are independent of each other by a normal distribution $N(0, \sigma^2)$. It should be noted here that the sine and cosine are calculated for each element in the matrix.

Self-attention. Attention mechanism is the core of the Transformer. In Fig. 2.8, it shows the structure of the self-attention mechanism.

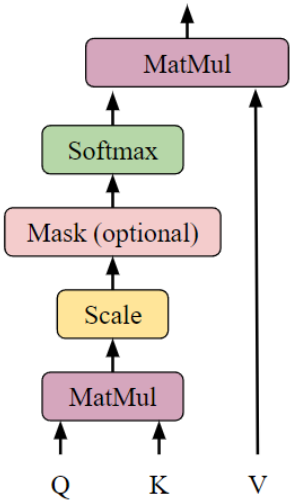


Figure 2.8: The structure of self-attention.

First, three vectors q, k, and v, representing the query, key, and value, respectively are obtained by multiplying three matrices W^Q, W^K, and W^V respectively with

input embedding X, and it uses the equation

$$\begin{aligned} q_i &= X_i W^Q \\ k_i &= X_i W^K \\ v_i &= X_i W^V,\ i = 1, 2, 3 \dots T \end{aligned} \tag{2.4}$$

where T is the number of the words.

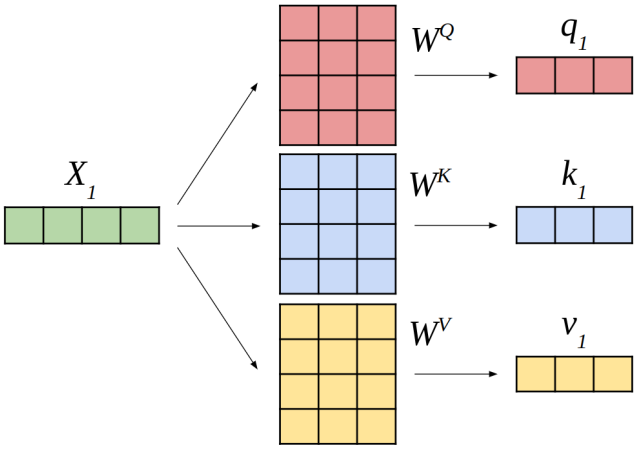
Then the self-attention score for each word can be obtained by taking the dot product of the query of the target word with the key of itself and other words

$$s_{ij} = q_i \cdot k_j,\ i, j = 1, 2, 3 \dots T. \tag{2.5}$$

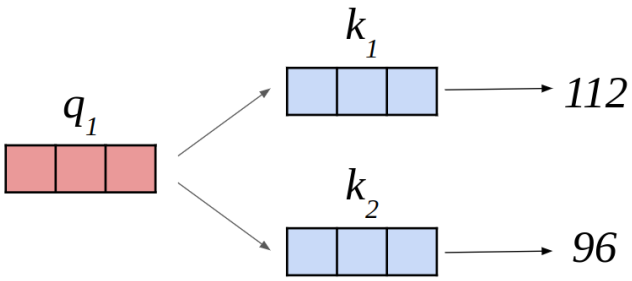
To have more stable gradients, the scores obtained by Eq. 2.5 divided by $\sqrt{d_k}$, the square root of the dimension of the key, and then a softmax operation along key dimension is applied on the result to get the possible values (weights) w ranged in the $[0, 1]$. The softmax scores reflect how closely other words relate to target position. Obviously, the word at the target position has the greatest softmax score, but it's occasionally helpful to focus on another word with high score that is related to the target word. Lastly, w multiplies with each value vector respectively and then sums the weighted values up to get self-attention output z

$$z_i = \sum_j w_{ij} v_j\ i, j = 1, 2, 3 \dots T. \tag{2.6}$$

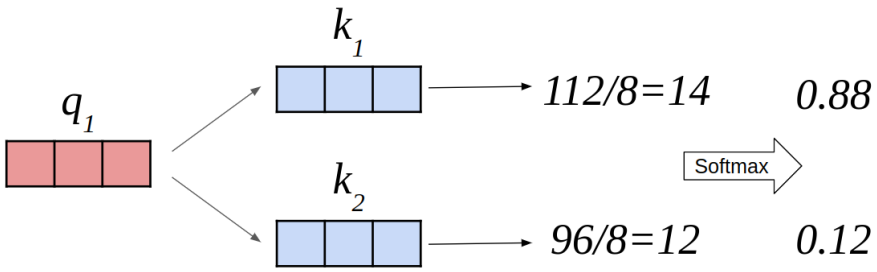
Here is a simple example to make the whole process of self-attention more intuitive. Assuming that the input consists of 2 words and their input embeddings are X_1 and X_2. Take X_1 as an example to calculate the self-attention score.



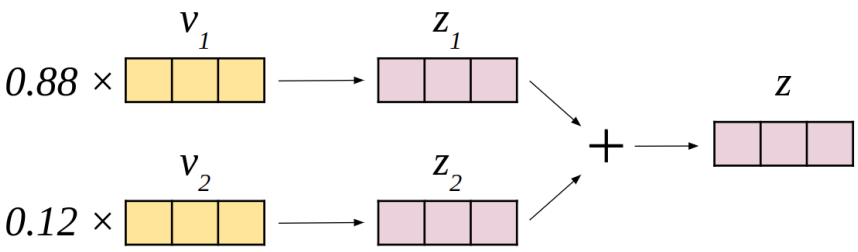
(a) First step, calculate q, k and v vectors of X_1.



(b) Second step, calculate the scores (assuming that the scores are 112 and 96).



(c) Third step, divide scores by $\sqrt{d_k}$ (normally, $d_k = 64$) and then apply a softmax operation to get the weights.



(d) In the fourth step, multiply weights with values and sum the weighted values up to get the Self-Attention output.

Figure 2.8: A simple example of calculating self-attention.

In conclusion, the expression of self-attention in matrix format is

$$SelfAttention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{2.7}$$

where Q, K and V are the matrix format of the query, key and value, respectively.

Multi-head self attention. Multi-head self-attention is used so that the network can focus on different aspects of how the data relates to itself, and to make use of the parallel computation power of GPUs. As shown in Fig. 2.9, $d-dimensional$ queries, keys and values are linearly projected h times to the d_q, d_k and d_v dimensions by different projection matrices W^Q, W^K and W^V, respectively. Then self-attention are applied on each of these projected query, key and value in parallel and then get the output values of dimension d_v. Finally, all of them will be concatenated and linearly projected again by the projection matrix W^O to the final output values, we get that

$$\begin{aligned} MultiHead(Q, K, V) &= \text{concat}(head_1, head_2, \cdots, head_h)W^O \\ \text{where } head_i &= Self\ Attention(XW_i^Q, XW_i^K, XW_i^V),\ i = 1, 2, 3, \ldots, h \end{aligned} \tag{2.8}$$

where h is the number of the heads, $W_i^Q \in \mathbb{R}^{d\times d_q}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$ and $W^O \in \mathbb{R}^{hd_v\times d}$ are the linear projection matrices.

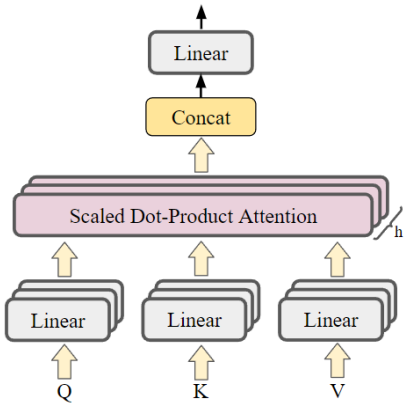


Figure 2.9: The structure of Multi-Head Self-Attention.

2.3 Point-based detection and tracking componets

Directly learning geometric features from unstructured point clouds, point-based algorithms generate explicit proposals for objects of interest. This section will explicate several important elements in point-based detection and tracking which are used in our method.

2.3.1 Set abstraction (SA)

Set abstraction (SA) layer consists of three parts: Sampling layer, grouping layer, and PointNet layer. In general, set abstraction is equivalent to sampling a sparse

point cloud from the input point cloud according to XYZ spatial coordinates and then calculating a new feature of each sampling point through grouping and PointNet networks.

Sampling layer. The sampling layer uses the farthest point sampling (FPS) sampling algorithm to to reduce the input point set from scale N to a smaller scale N'. The FPS randomly selects a point as the initial point as the selected sample point, calculates the distance between each point in the unselected sample point set and the selected sample point set, adds the point with the greatest distance to the selected sample point set, and then updates the distance, iterating until the target number of sample points is obtained.

Grouping layer. The purpose of grouping is to find a fixed size of neighbours of each sampling point to form a neighbourhood patch. This means that N' neighbourhood patches are generated.

There are two ways to find a neighbourhood here: k-nearest neighbor (KNN) and query ball point. KNN is to find the K nearest points in coordinate space, while query ball point is to set a certain radius and find points within the ball of that radius as neighbouring points.

PointNet layer. Here PointNet is that point-based detector PointNet [30]. Each group obtained by grouping layer can be regarded as a 'local point cloud', and the feature of this local point cloud is extracted by the PointNet network.

2.3.2 Hungarian algorithm

The Hungarian algotithm [18] is a combinatorial optimization algorithm that solves the assignment problem and it can be used to do the bipartite matching in the tracking tasks. Assuming there are two sets of point of same size N, the Euclidean distance between points in two sets are used as matching cost, which is expressed as

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \tag{2.9}$$

The Hungarian algorithm is to find a minimum total cost of matching

$$\begin{aligned} &\min \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} p_{ij} \\ &s.t. \sum_{i=1}^{N} p_{ij} = 1, \sum_{j=1}^{N} p_{ij} = 1, p_{ij} \in \{0, 1\} \end{aligned} \tag{2.10}$$

where p_{ij} is used to limit that every point can only be matched once.

In some tracking tasks, the Hungarian algorithm also take IoU or GIoU as matching cost to match the objects in two adjacent frames, it can be expressed as

$$\text{GIoU} = \text{IoU} - \frac{|C/(A \cup B)|}{|C|} \tag{2.11}$$

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{2.12}$$

(a) Overlap. (b) Union.

Figure 2.10: The area of overlap and union.

where C is the smallest enclosing convex for A and B.

2.3.3 Non-maximum suppression

Non maximum suppression (NMS) is an algorithm to remove non maximum values, which is often used object detection. In object detection, assuming there are N detection boxes $\mathcal{B} = \{b_1, b_2, \ldots, b_N\}$ with corresponding detection scores $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$. The NMS algorithm flow shows as

1. Create an initially empty set $\mathcal{D}$ to store boxes with highest score.
2. Put the box with highest score b_m into $\mathcal{D}$, and remove it from $\mathcal{B}$.
3. Traverse the boxes in $\mathcal{B}$, and calculate the IoU with the box b_m respectively. If it is higher than a certain threshold (generally 0 0.5), this box is considered to overlap with box b_m, and this box will be removed from $\mathcal{B}$.
4. Go back to step 1 and iterate until $\mathcal{B}$ is empty. Finally, the boxes in $\mathcal{D}$ are results.

2.4 The methods used in thesis

2.4.1 DEtection TRansformer (DETR)

The majority of traditional detectors are built on convolutional designs, which have advanced significantly in the recent decade [33, 32, 3, 10]. However, these methods highly rely on handcrafted components such as anchors and NMS (non-maximum suppression). DETR [6] creates the end-to-end fashion in this field. It is an encoder-decoder Transformer-based detector that eliminates the need for hand-designed components, such as anchors, and exhibits promising performance in comparison to recent anchor-based detectors like Faster RCNN [33]. Since Transformer is a fascinating architecture for sequence prediction, DETR approaches object detection as a set prediction issue, with learnable queries probing and pooling features from pictures to create predictions without the use of spatial anchors and non-maximum suppression.

Overall structure. The overall structure of DETR is shown in Fig. 2.11, consisting of three main modules: a CNN backbone to extract a set of image features ($3 \times H_0 \times W_0 \rightarrow 2048 \times \frac{H_0}{32} \times \frac{W_0}{32}$), a Transformer formed by an encoding component (several stacked identical encoder layers) and a decoding component (several stacked identical decoder layers), and a feed-forward network (FFN) to predict the bounding boxes and the labels of objects in the image. There are two key parts to the whole process. The first is to use the encoder-decoder architecture of Transformer to generate N box predictions at once, where N is an integer far larger than the number of objects in the image. The second is the design of bipartite matching loss, which calculates the loss based on the bipartite graph matching of predicted boxes and ground truth boxes, so as to make the location and category of predicted boxes closer to ground truth.

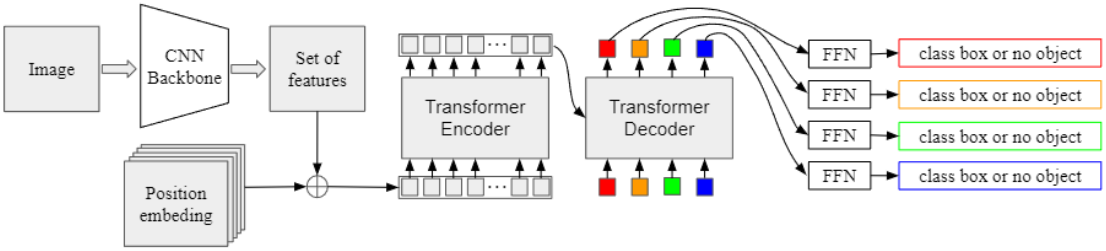


Figure 2.11: The overall structure of DETR.

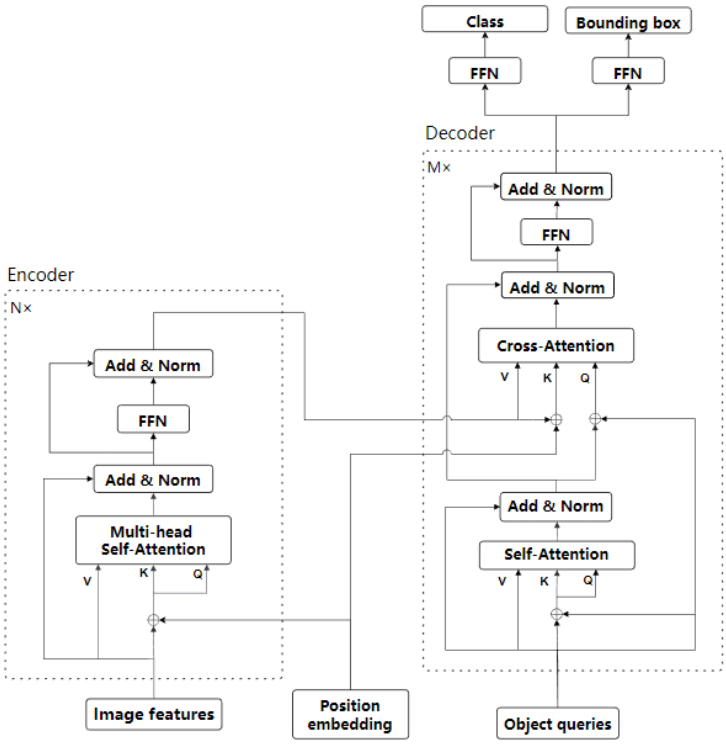


Figure 2.12: The structure of DETR's Transformer.

Transformer. The Transformer structure of DETR is shown in Fig. 2.12. The encoder takes the feature map from the CNN backbone and spatial positional encoding as input and outputs the image embedding used in the decoder. Before encoding, the feature map needs to be converted to the sequential data that can be processed by the Transformer encoder. The main steps are as follows:

- Dimension compression: first process the $C \times W \times H$ feature map from CNN backbone with 1×1 convolution, and compress the number of channels from C to d, obtaining a new $d \times W \times H$ feature map.
- Convert to sequential data: compress the spatial dimension (height and width) into one dimension, that is, reshape the $d \times W \times H$ feature map obtained in the previous step into a $d \times HW$ feature map.
- Add position encoding: since the Transformer model is permutation equivalent, and the HW in the $d \times HW$ feature map is related to the location of the original image, position encoding needs to be added to reflect the location information.

The decoder takes the image embedding, spatial positional encoding and object queries (learned positional encoding) as input and outputs the prediction bounding box and class label of objects. Here, the number of object queries N is much larger than that of objects in the image. In addition, the object queries will be updated based on the loss between prediction and the ground truth.

Loss of DETR. Since the number of predictions is greater than that of ground truths, the ground truth is padded with "no object" class to be the same number of predictions, then a bipartite matching between two sets, prediction and ground truth, can be implemented by finding the lowest cost shown in Eq. 2.13, where y_i denotes the ith object in ground truth set, $\hat{y}$ denotes the prediction set and $\sigma(i)$ is the index of the prediction box paired with the i^{th} object

$$\hat{\sigma} = \arg\min_{\sigma} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}). \tag{2.13}$$

The cost $\mathcal{L}_{match}$ matching each pair of prediction and ground truth is defined as

$$\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}), \tag{2.14}$$

where c_i is the class label of the ith object, $\hat{p}_{\sigma(i)}(c_i)$ is the probability predicted by the Tranformer that an object has class c_i, b and $\hat{b}$ represent the box information including the normalized center coordinates (x, y) and normalized size (width, height) of ground truth and prediction respectively, the definition of $\mathbb{1}$ is shown in Eq. 2.15, and $\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)})$ is the distance between prediction box and ground truth box whose expression is shown in Eq. 2.16, where λ_{iou} and λ_{L1} are the loss weights, $B(b_{\sigma(i)}, \hat{b}_i)$ represents the largest box containg both $b_{\sigma(i)}$ and $\hat{b}_i$

$$\mathbb{1}_{\{c \neq \varnothing\}} = \begin{cases} 1, \text{if } c \neq \varnothing, \\ 0, \text{otherwise} \end{cases} \tag{2.15}$$

$$\begin{aligned} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) &= \lambda_{iou} \mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1} ||b_i - \hat{b}_{\sigma(i)}||_1, \\ \text{where } \mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) &= 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) / b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|}\right). \end{aligned} \tag{2.16}$$

After obtaining the optimal match between prediction boxes and image objects, then get the set prediction loss by

$$\mathcal{L}_{Hungarian}(y, \hat{y}) = \sum_{i}^{N} [-\log \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})]. \tag{2.17}$$

Drawbacks and improvements of DETR. In DETR, each object query will extract the corresponding object features from the image through attention layers in the Transformer decoder according to their respective regions of interest. This feature extraction process includes two steps: one is the similarity matching of key (image features) to query (object queries), and the other is the weighted average of value (image features) according to the matching results. The problem lies in the first step: since query embedding is initialized randomly, object queries and image features cannot be matched correctly in the early stage of model training. Intuitively, a key (object query) opens a lock (an object in a specific area on the picture). However, due to the random initialization of the key, it can not actually open any lock (the similarity and matching degree of the features at any position on the image to object query are very low). The result is that the cross attention layer actually extracts the features of the whole picture almost evenly, rather than extracting the object features in a specific area. It is conceivable that the object query after this not only collects the features of the target object, but also the features of the picture background and other irrelevant objects. Therefore, it is still very difficult to classify and locate the target object with it. From another perspective, cross attention can be regarded as a kind of soft RoIAlign: divide the region of interest from the picture according to KQ correlation map and collect the corresponding features. The reason why DETR is difficult to converge is that due to the misalignment between object query and image feature, it is unable to collect object features in specific areas, but other very messy features in the image.

Since the original cross attention layer has too high a degree of freedom to focus on a specific area, Deformable DETR [57] proposes to set a reference point for each object query, and the proposed deformable attention only samples n picture features near the center point, which not only accelerates convergence but also reduces the amount of calculation. Moreover, since the amount of computation of deformable attention is independent of the size of the feature map, a multi-scale feature map can also be used to optimize the detection of small objects. In SMCA DETR [9], before calculating cross attention, each query first predicts the position, length, and width of the object to be detected, and then generates a corresponding Gaussian-like weight map according to the predicted position, length, and width of the object, which is integrated into KQ similarity matching calculation in cross attention. In this way, the features extracted by each query are limited to the center of the object, and the convergence speed is improved. Anchor DETR [42] directly encodes the location of the anchor point as an object query, and the encoder and decoder share a set of location coding methods (sine encoding + MLP). In this way, the position embedding of encoder and decoder is aligned, and the features extracted by each query are concentrated near the anchor point. DAB-DETR [20] is optimized in two aspects compared with anchor DETR. One is to encode the anchor box (including

four dimensions of location length and width) as an object query, rather than just encode the location information of the anchor point. Second, each layer continuously refines the anchor box (the output of the previous layer is used as the input of the next layer). It is worth noting that the authors use the predicted box length and width to further limit the KQ similarity matching calculation in cross-attention so that the generated attention map can fit objects with different positions, scales, and aspect ratios. These papers all limit the sampling area of object query in different ways so that the network can focus on the object area faster and speed up the training.

2.4.2 3D DEtection TRansformer (3DETR)

3DETR [27] is an extension of DETR and applies the same design philosophy to 3D point clouds for indoor object detection. It employs the Transformer with non-parametric queries and Fourier positional embeddings to cast detection as set-to-set problem and make it more competitive than the models which employ lots of 3D operators with hand-crafted designs, such as VoteNet and PointNet++.

The process of 3DETR. 3DETR takes the point clouds as input and predicts object positions using a 3D bounding box. A point cloud is an unorganized collection of N points, each with its own 3D XYZ coordinates. Due to the enormous number of input points, a set aggregation downsampling operation is utilized to downsampling and project the points into N' features using MLP with two hidden layers. The encoder receives the features and produces the same size features by applying multiple layers of self-attention and non-linear projections. The decoder takes these features and a set of non-parametric query embeddings of size B $\{q_1^e, \cdots q_B^e\}$ as input and then generate a set of B features which is later leveraged to predicts B bounding boxes in the parallel decoding way.

For non-parametric query embeddings, from the N input points, a set of B query points $\{q_i\}_{i=1}^{B}$ is randomly selected from the N' input points. The random samples are drawn using Farthest Point Sampling, which can guarantee to cover all points in the original set of points. By transforming the coordinates of q_i into Fourier positional embeddings and then projecting with an MLP, each query point q_i is associated with a query embedding q_i^e.

Set matching and loss function. 3DETR uses the method in DETR to do bipartite graph matching and then calculates a loss for each predicted boxes $\hat{b}$ and its corresponding ground truth boxes b. Using a geometric and a semantic term, a matching cost for a pair of boxes, a predicted bounding box $\hat{b}$ and a ground truth bounding box b is calculated as follows

$$C_{match}(\hat{b}, b) = \underbrace{-\lambda_1 GIoU(\hat{b}, b) + \lambda_2 ||\hat{c} - c||_1}_{\text{geometric}} \underbrace{- \lambda_3 \hat{s}[s_{gt}] + \lambda_4 (1 - \hat{s}[s_{bg}])}_{\text{semantic}}. \tag{2.18}$$

The geometric cost uses GIoU to calculate the box overlap and measures the distance between the box centers. The semantic cost is a metric that evaluates the

likelihood of the ground truth class s_{gt} under the anticipated distribution $\hat{s}$, as well as the likelihood of box features belonging to a foreground class. Boxes that are not matched to the background class will be regarded as matched to the background class s_{bg}.

For the center locations and the box dimensions, l_1 regression loss is used, and an additional normalization is used to limit them both in the range [0, 1]. Huber regression loss [15] is used for the angular residuals and cross-entropy loss is used for both the angular classification and semantic classification. The whole training loss for 3DETR is shown as follows

$$\begin{aligned}\mathcal{L}_{3DETR} = \lambda_c||\hat{c} - c||_1 + \lambda_d||\hat{d} - d||_1 + \lambda_{ar}||\hat{a}_r - a_r||_{huber} \\ - \lambda_{ac} a_c^\top \log \hat{a}_c - \lambda_s s_c^\top \log \hat{s}_c.\end{aligned} \tag{2.19}$$

2.4.3 IA-SSD

IA-SSD [52] is an efficient single-stage detector with good performance in terms of both running speed and accuracy. Unlike common point-based pipelines using random sampling or farthest point sampling which will take background useless points into the account to downsampling the input point cloud, IA-SSD proposed to use instance-aware downsampling strategies (class-aware and centroid-aware) to select foreground points that correspond to the objects of interest. Fig. 2.13 is the point cloud of an example scene in the NuScenes dataset. It is obvious that the useless background points are more numerous than the foreground points of interest. Therefore, IA-SSD can exploit more useful and important information in the point cloud and can be much more efficient than other point-based methods.

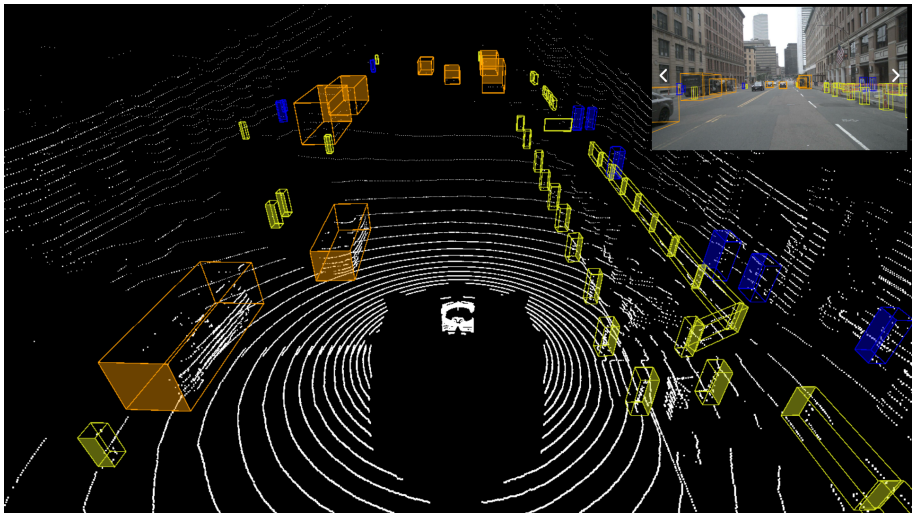


Figure 2.13: The point cloud of an example scene in the NuScenes dataset. It shows that the useless background points are more numerous than the foreground points of interest.

IA-SSD takes the point cloud as input, and then a 3D backbone is leveraged to extract point-wise features, which is then followed by the proposed instance-aware

downsampling to lower the computation complexity while still maintaining the important foreground points. The contextual centroid perception module (consisting of contextual centroid prediction, centroid-based instance aggregation, and proposal generation head) receives the learned features and uses them to build instance proposals. Then 3D NMS post-processing is used to filter the proposals based on a specific IoU threshold to obtain the ultimate bounding boxes.

Class-aware sampling. To accomplish selective downsampling, this sampling approach seeks to learn semantics for each point. To do this, MLP layers are added to make use of the rich semantics of latent features and predict the semantic categories of each point

$$\mathcal{L}_{cls-aware} = -\sum_{c=1}^{C}(s_i \log(\hat{s_i}) + (1 - s_i)\log(1 - \hat{s_i})) \tag{2.20}$$

where C is the number of categories, s_i is the one-hot labels generated from the annotations of the ground truth bounding box, while $\hat{s}_i$ is the predicted counterpart of s_i. The foreground points with top k scores will be retained and fed into the next layer. This way, more background points can be removed while more foreground points can be preserved compared to the common random sampling and farthest point sampling.

Centroid-aware sampling. Since the centroid of the object is essential to locate the position of the object, this strategy gives higher mask scores to the points which are closer to the centroid. The definition of the mask score of instance i notes as follows

$$Mask_i = \sqrt[3]{\frac{\min(f^*, b^*)}{\max(f^*, b^*)} \times \frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(u^*, d^*)}{\max(u^*, d^*)}} \tag{2.21}$$

where f^*, b^*, l^*, r^*, u^*, d^* denote the distance between the point and six surfaces of the bounding box (front, back, left, right, up, and down). The highest mask score is 1 if the point is the centroid of the object, while the lowest is 0 if the point lies on the surface. This mask score is used to assign the weight to the points in the bounding box in the following way

$$\mathcal{L}_{ctr-aware} = -\sum_{c=1}^{C}(Mask_i \cdot s_i \log(\hat{s_i}) + (1 - s_i)\log(1 - \hat{s_i})). \tag{2.22}$$

Contextual instance centroid perception. To predict instance centroid, contextual information around the bounding box is used. An offset $\Delta\hat{c_{ij}}$ from the jth point of ith instance to the centroid of ith instance is predicted and a regularization ingredient is also included to reduce the uncertainty in the centroid prediction. The centroid prediction loss is defined as

$$\begin{aligned} \mathcal{L}_{cent} =& \frac{1}{\mathcal{F}_+}\frac{1}{\mathcal{S}_+}\sum_i\sum_j(|\Delta\hat{c_{ij}} - \Delta c_{ij}| + |\hat{c_{ij}} - \bar{c_i}|) \cdot \mathbf{I}_{\mathcal{S}}(p_{ij}), \\ & \text{where } \bar{c_i} = \frac{1}{\mathcal{S}_+}\sum_j \hat{c_{ij}}, \quad \mathbf{I}_{\mathcal{S}} : \mathcal{P} \to \{0, 1\}. \end{aligned} \tag{2.23}$$

Here $\mathcal{F}_+$ is the number of instances, $\mathcal{S}_+$ is the number of the points used in prediction, Δc_{ij} is the ground truth of the offset, $\bar{c}_i$ is the mean location coordinates of ith instance, and $\mathbf{I}_\mathcal{S}$ is a function that determines whether or not this point will be utilized to estimate the instance centroid. If point p_{ij} is used, then $\mathbf{I}_\mathcal{S}(p_{ij})$ is 1, otherwise 0.

End-to-end loss. In the end-to-end training procedure, the total loss includes the loss of two downsampling strategies, centroid prediction, classification, and box generation

$$\mathcal{L}_{total} = \underbrace{\mathcal{L}_{cls-aware} + \mathcal{L}_{ctr-aware}}_{\text{downsampling}} + \mathcal{L}_{cent} + \mathcal{L}_{cls} + \mathcal{L}_{box} \tag{2.24}$$

where the box generation loss consists of position, size, angle classification, angle division regression and corner point 5 components

$$\mathcal{L}_{box} = \mathcal{L}_{loc} + \mathcal{L}_{size} + \mathcal{L}_{angle-bin} + \mathcal{L}_{angle-res} + \mathcal{L}_{corner}. \tag{2.25}$$

2.4.4 MOTR

MOTR [51] is a true full end-to-end tracking framework. MOTR is able to learn to model long-range temporal variation of the object, which implicitly makes temporal associations and avoids previous explicit heuristic strategies. Based on Transformer and DETR [6], MOTR introduces the concept of "track query". A track query is responsible for modeling the entire trajectory of an object. It can be transmitted and updated between frames to seamlessly complete object detection and tracking tasks. An additional temporal aggregation network (TAN) in Query Interaction Module (QIM) with multi-frame training is used to model long-range temporal relationships.

Detect query and track query. The fixed-length detect query q_{det} represents the new-born objects Obj_{new} at every frame. while the track query q_{tr} represents tracked objects Obj_{tr} in the previous frames. As shown in Fig. 2.14, new-born objects (such as "1" and "3" at T_1) are detected using fixed-length detect queries. At the next frame, the detect queries given to Obj_{new} are updated and merged into the track query set. The empty initially track query set is dynamically updated and has an adjustable length. Disappear objects Obj_{dis} (such as "3" at T_3) counts its disappear time for next several frames. If disappear time of Obj_{dis} larger than a threshold τ_{dis}, then remove the q_{tr} of Obj_{dis} from the track query set and set Obj_{dis} as exit object Obj_{exit}.

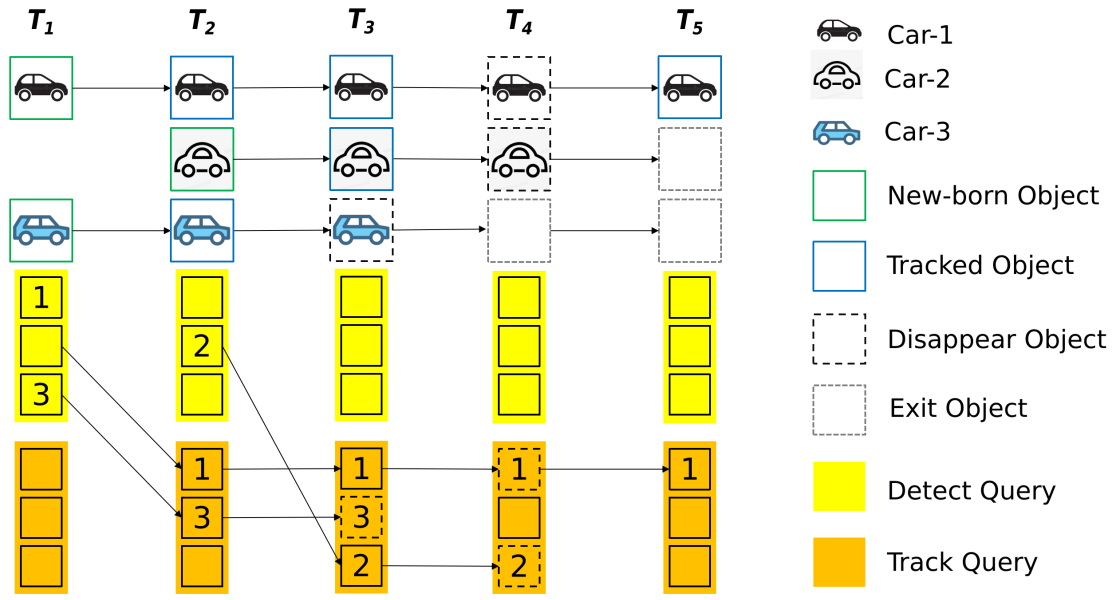


Figure 2.14: The detect query and track query in MOTR.

MOTR structure. The overall structure of MOTR is shown in Fig. 2.15. The video sequence will first be sent to CNN, and then enter the encoder of Deformable DETR to extract the basic features $f = \{f_1, \cdots f_N\}$, where f_i is the feature of frame T_i. For first frame T_1, the feature f_1 and empty query set q_d are input into the decoder network to locate all initialization targets and generate the original track query set q_{ot}^1, and then q_{ot}^1 is sent to next frame via QIM. Generally, the whole model processes each frame F_i ($i \in [1, N]$) iteratively. The q_t^i generated by QIM according to the output of the previous frame will be concatenated with the empty query set q_d. The concatenated query set will be sent to the decoder together with feature f_i to directly generate the prediction result in Y_i of the current frame, and update the query set q_t^{i+1} to send it to the next frame. The prediction and ground truth are used to calculate the collective average loss.

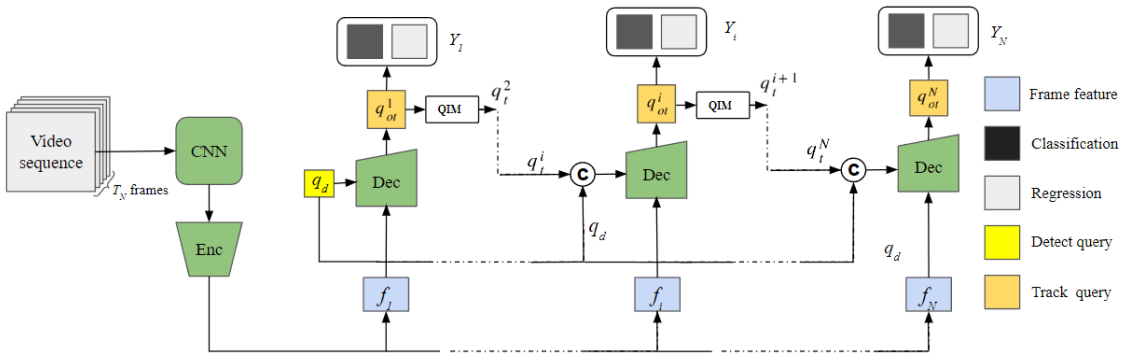


Figure 2.15: The overall structure of MOTR.

Query interaction module (QIM). QIM consists of the object entrance-exit mechanism and temporal aggregation network (TAN). It aims to detect new-born

objects and exit objects and model long-range temporal relationships.

For frame T_i, track query set q_t^i is generated from T_{i-1} frame through QIM, and then concatenated with the empty query set q_d. Then the concatenated result is input into the decoder to generate the original track query set q_{ot}^i containing tracking score. Then q_{ot}^i is divided into two set q_{tr}^i consisting of tracked and dead objects and q_{det}^i consisting of new-born objects. For the entry of the object, if the score of query in q_{det}^i is greater than the entry threshold τ_{en}, it will be retained and the rest will be removed, the equation can be expressed as

$$\bar{q}_{det}^i = \{q_k \in q_{det}^i | s_k > \tau_{en}\} \tag{2.26}$$

where s_k represents the score of the kth query q_k of q_{det}^i.

For the exit of the target, if the tracking score of q_{tr}^i query is lower than the exit threshold τ_{ex} for M consecutive frames, it will be removed, and the remaining query will be retained. The formula is as follows

$$\bar{q}_{tr}^i = \{q_k \in q_{det}^i | \max(s_k^i, \cdots, s_k^{i-M}) > \tau_{ex}\} \tag{2.27}$$

where s_k^i represents the score of the kth query q_k of q_{tr} at frame T_i.

The purpose of TAN is to enhance time correlation and provide context prior information for tracking objects. The filtered track query $\bar{q}_{tr}^i$ and $\bar{q}_{tr}^{i-1}$ above are concatenated to go through the multi-head attention layer as query and key and $\bar{q}_{tr}^i$ is regarded as a value. The whole procedure is shown below

$$tgt = softmax[\frac{(\bar{q}_{tr}^i + \bar{q}_{tr}^{i-1}) \cdot (\bar{q}_{tr}^i + \bar{q}_{tr}^{i-1})^\top}{\sqrt{d}}] \cdot \bar{q}_{tr}^i. \tag{2.28}$$

After that, tgt is further adjusted by a FFN network

$$tgt = LN(tgt + \bar{q}_{tr}^i) \tag{2.29}$$

Here, FC represents linear projection layer, LN represents layer normalization, and σ_r represents ReLU activation function.

The output tgt is concatenated with $\bar{q}_{det}^i$ to get the tracking query q_t^{i+1} at the frame T_{i+1}

$$tgt = LN(FC(\sigma_r(FC(tgt))) + tgt). \tag{2.30}$$

Collective average loss. MOTR takes the video sequence as the input and calculates the tracking loss frame by frame. The overall tracking loss is the sum of the losses of each frame and the result is normalized according to the number of ground truth objects, as shown in Eq. 2.31, where N is the length of the input sequence, $Y = \{Y_i\}_{i=1}^N$ is the multiple predictions of N frames, $\hat{Y} = \{\hat{Y}_i\}_{i=1}^N$ is the ground truth of N frames, and V_i represents the total number of ground truth objects at T_i frame

$$\mathcal{L}_o(Y, \hat{Y}) = \frac{\sum_{n=1}^{N} (\mathcal{L}(Y_{tr}^i, \hat{Y}_{tr}^i) + \mathcal{L}(Y_{det}^i, \hat{Y}_{det}^i))}{\sum_{n=1}^{N} (V_i)}. \tag{2.31}$$

$\mathcal{L}$ is the tracking loss of a single frame, which can be expressed by the Eq. 2.32, where $\mathcal{L}_{cls}$ is focal loss, $\mathcal{L}_{l1}$ denotes L1 loss, and $\mathcal{L}_{giou}$ represents generalized IoU. λ_{cls},λ_{l1} and λ_{giou} are the weights corresponding to the each loss

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{l1}\mathcal{L}_{l1} + \lambda_{giou}\mathcal{L}_{giou}. \tag{2.32}$$

2.5 NuScenes dataset and evaluation

2.5.1 NuScenes

NuScenes [5] is the first large-scale dataset to provide a complete set of sensor data for automatic vehicles, including 6 cameras, 1 lidar, 5-millimeter wave radars, GPS, and IMU. Compared with the Kitti dataset, it contains more than 7 times more object annotations. The dataset consists of 1000 driving scenes in both Singapore and Boston, each of which is 20 seconds long and contains a variety of traffic situations. In each scene, there are 40 keyframes, that is, there are 2 key frames per second, and the other frames are sweeps. The keyframes are manually labeled for 23 object classes, and there are several annotations in each frame. The form of annotation is a bounding box. Not only the size, range, and class but also the visibility, activity, and pose are marked.

2.5.2 Evaluation

To judge whether our MOT model can accurately locate the position of each object, as so to achieve continuous tracking in the continuous frame, we use MOT metrics [1] to evaluate our model, including the following criteria which are same as for the NuScenes tracking tasks

1. Multiple Object Tracking Accuracy (MOTA)
2. Multiple Object Tracking Precision (MOTP)
3. Mostly Tracked(MT): The proportion of tracks that meet the requirements of ground truth that match successfully at least 80% of the time in all tracking targets. Note that MT and ML here have nothing to do with whether the ID of the current track changes, as long as the ground truth matches the target.
4. Mostly Lost(ML): The proportion of tracks that meet the ground truth matching success in less than 20% of the time in all tracking targets.
5. ID Switch (IDS): The number of times the ID assigned by ground truth has changed.
6. Fragmentation(FM): FM calculates how many times the track is interrupted (i.e. the track of ground truth is not matched). In other words, FM will be counted whenever the track changes its state from tracking state to untracked state and tracks the same track at a later time point. It should be noted here that the status of ground truth during FM counting needs to meet the requirements of tracked $\rightarrow$ untracked $\rightarrow$ tracked. It should be noted that FM has nothing to do with whether the ID has changed.
7. False Positive (FP): The track and detection predicted in the current frame do not match, and the incorrectly predicted track point is called FP. Whether the matching is successful or not is related to the threshold set when matching.

8. False Negative (FN): The predicted track and detection of the current frame are not matched, and the unmatched ground truth point is called FN (also called Miss).
9. Recall: Ratio between the number of correctly matched detections and the total number of all GT boxes.
10. Precision: Ratio between the total number of detected objects and FP.
11. FAF: The average number of false alarms each frame.
12. FRAG: The total number of fragmentation in a trajectory.

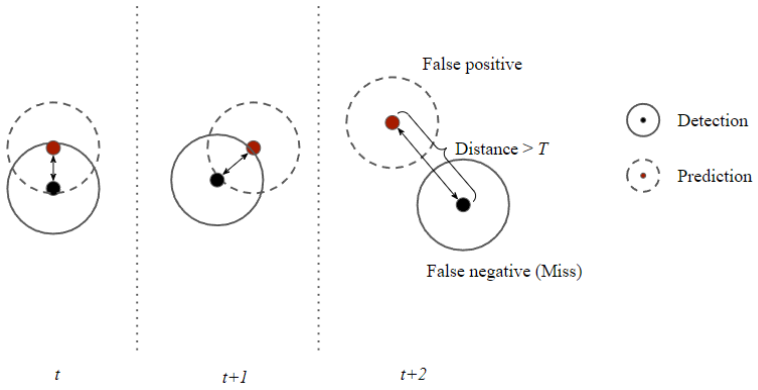


Figure 2.16: The illustration of FP and FN.

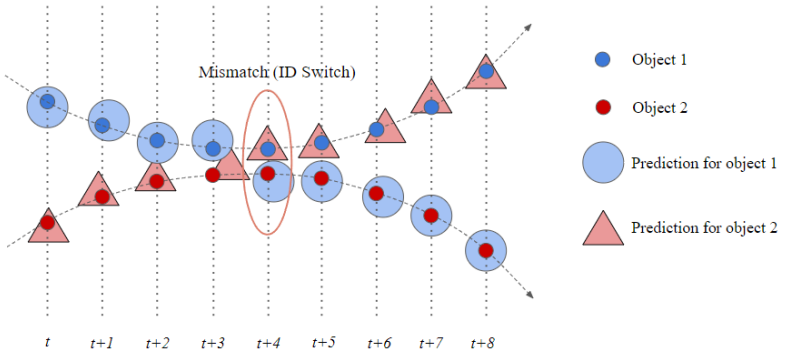


Figure 2.17: A simple example of mismatch (IDS).

MOTA reflects the accuracy of determining the number of targets and the accuracy of relevant attributes of targets, which is used to count the error accumulation in tracking, including FP, FN, IDS, which is expressed as

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t} \tag{2.33}$$

where FN_t, FP_t, IDS_t and GT_t represent the FN, FP, IDS and the number of ground truth objects at the frame t.

MOTP measures the accuracy of the detection results, that is, the average measurement distance between the detection frame and the GT assigned to it. Defined

as

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{2.34}$$

where c_t represents the number of detection frames successfully matched with GT in the tth frame and the matching criterion is to judge whether the distance between detection and ground truth is greater or less than a certain threshold T. $d_{t,i}$ represents the distance measurement between matching ith pair in the tth frame (IoU, GIoU or European distance, etc.).

3

Method

Inspired by point-based 3D detector IA-SSD[52], DETR-based 3D detector 3DETR [27], and DETR-based 2D tracker MOTR [51], we propose an efficient end-to-end method about multi-object tracking with 3D point-based Transformer (MOTPT).

It first downsamplings the point cloud, extracts the point cloud feature, and then inputs these downsampled points and point features into the Transformer to detect and predict the track for vehicles. At last, spatial and temporal attention help to capture the long-term relationship of vehicles in different frames.

3.1 MOTPT pipeline

Our MOTPT model combines an efficient single-stage point-based 3D detector IA-SSD as the backbone with an end-to-end 2D tracker MOTR. Both have some modifications to make our model able to track multiple objects based on a 3D point cloud in an end-to-end fashion. Compared with MOTR, we replace the convolutional neural network (CNN) backbone with the IA-SSD backbone because the IA-SSD backbone efficiently extracts point cloud features and distinguishes the foreground and background instances. Furthermore, we replace the deformable Transformer with learnable queries to the 3DETR Transformer with non-parametric queries because we find that learnable queries can not work well in the sparse point cloud.

The whole pipeline is shown in Fig. 3.1. A continuous point cloud sequence of length n is the input. For point cloud pc^{t-1} at frame $t-1$, first, it is fed to the IA-SSD backbone, which consists of several downsampling SA layers to extract the sampled points and point features as detection query q_{det}^{t-1}. Then Transformer predicts the detection objects as original track query q_{otr}^{t-1}. Furthermore, q_{otr}^{t-1} is used to calculate the long-term spatial and temporal relationship by the spatial-temporal module (STM) to get the track query q_{tr}^{t-1}. At frame t, the q_{tr}^{t-1} is concatenated with the new detect query at frame t as query and fed to the Transformer to create original track query q_{otr}^{t} at time t. In the training mode, all the output bounding boxes B_{tr} of the track queries $\{q_{tr}^{t-n}, q_{tr}^{t-n+1}, \ldots, q_{tr}^{t}\}$ from frame $t-n$ to t are used to calculate collective average loss with their corresponding ground truth of each frame.

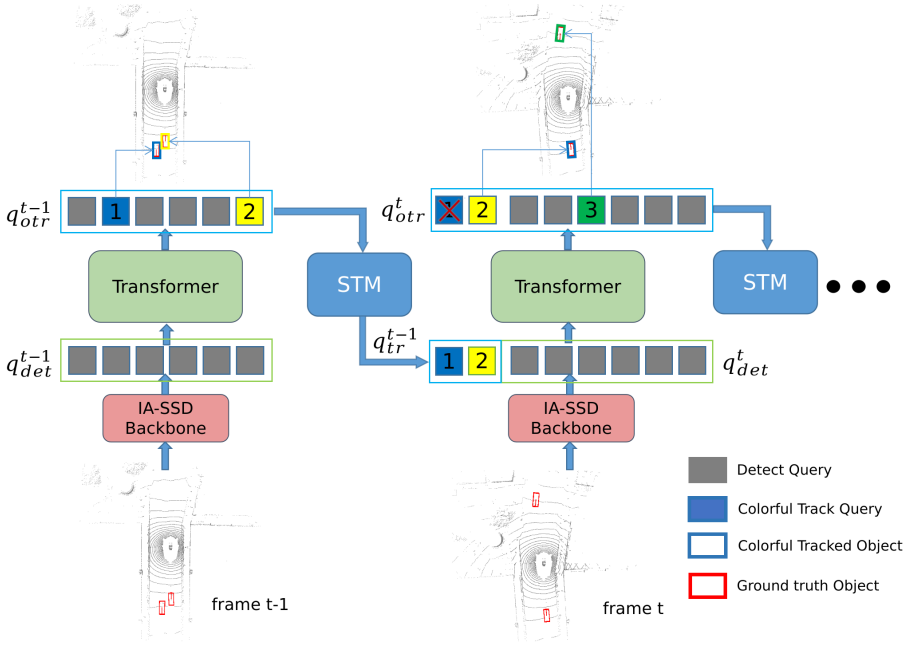


Figure 3.1: The overall structure of MOTPT.

3.2 Preprocessing

Preprocessing is needed to make the training of our model more stable and reliable. Several preprocessing methods are applied to input data, including point cloud clipping, data normalization, division of driving direction, coordinate transformation, and increasing the frame rate of the point cloud sequence.

Point cloud clipping. When the detection distance of the LiDAR is farther, the noise of the LiDAR is more apparent, and the reliability is low. Hence, we need to clip the range of lidar detection. We visualize the histograms of point clouds with different distances in Fig.3.2.

We note that the point cloud is sparse in the NuScenes dataset. Most vehicle bounding boxes only contain 2 or 3 points. The number of vehicles with a few points decreases dramatically by clipping a smaller range. When the clipping range is too small, like 40 meters in X and Y directions, it discards some of the valuable point clouds. Thus, we decide to choose 50 meters as the clipping range.

Data normalization. The extreme data increases the training frame or leads to divergence. In this case, it is necessary to apply normalization to the input data before the whole model training to reduce the impact of extreme data.

The normalized point cloud $PC_{normal}(x, y, z)$ is the input point cloud $PC(x, y, z)$ divide by its range R_{PC}. Additionally, the GT needs to normalize the center coordinates $C(x, y, z)$ and the size $S(l, w, h)$ including length, width and height. To be specific, R_{PC} is a three dimensional vector $[R_X, R_Y, R_Z]$ consisting of the range

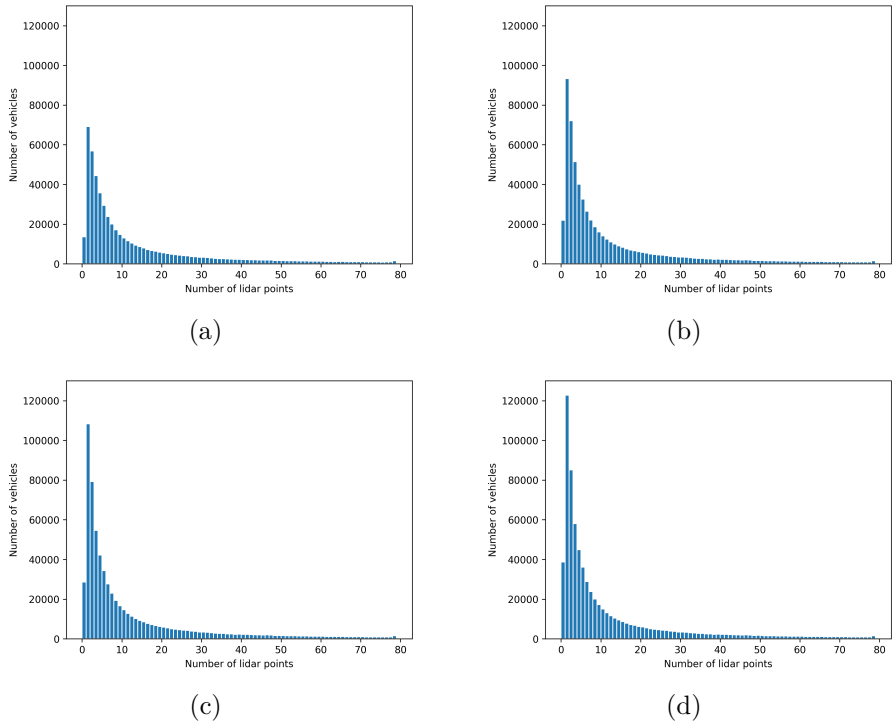


Figure 3.2: The histogram of the number of LiDAR point clouds in each vehicle bounding box with different distance clipping in X,Y direction. (a) 40 meter clipping, (b) 50 meter clipping, (c) 60 meter clipping, (d) no clipping.

on the X, Y and Z axis, and R is the difference between the maximum coordinate value and minimum coordinate value of the points in the point cloud. Point cloud normalization equations show as

$$\begin{aligned} R &= [R_X, R_Y, R_Z] = [X_{max} - X_{min}, Y_{max} - Y_{min}, Z_{max} - Z_{min}] \\ PC_{normal}(x, y, z) &= (\frac{PC_x}{R_X}, \frac{PC_y}{R_Y}, \frac{PC_z}{R_Z}) \in [-1, 1]^3. \end{aligned} \tag{3.1}$$

Moreover, ground truth bounding boxes normalization equations is given by

$$C(\bar{x}, \bar{y}, \bar{z}) = (\frac{C_x}{R_X}, \frac{C_y}{R_Y}, \frac{C_z}{R_Z}) \in [-1, 1]^3 \tag{3.2}$$

$$S(\bar{l}, \bar{w}, \bar{h}) = (\frac{S_l}{R_X}, \frac{S_w}{R_Y}, \frac{S_h}{R_Z}) \in [0, 1]^3. \tag{3.3}$$

Division of driving direction. In 3D MOT, predicting the orientation is a hard step compared to the 2D counterpart, where the bounding box is always a rectangle pointing in the same direction. To get the precise prediction of orientation, we divide the 360° angle into 12 bins, representing 30°. Then the orientation prediction problem becomes a classification problem to predict which bin the direction belongs to. In addition, a residual angle $\gamma \in [0, \pi/6)$ is needed to determine a specific direction in that bin.

Ego-Motion Compensation. It is necessary to apply for the ego-motion compensation in two LiDAR frames because the ego-motion transformation influences the tracker's accuracy. The ego-motion compensation (EMC) module transforms the tracker state in feature space [34]. In our method, we explored the importance of ego-motion compensation. To be simplified, the ego-motion will be removed in the tracking mode for the experiment, which transfers the point cloud and the ground truth bounding boxes from the LiDAR coordinate to the global coordinate

$$P_{global} = T_{lidar}^{global} \cdot P_{lidar} \tag{3.4}$$

where P_{global} and P_{lidar} are the point cloud in global coordinate and LiDAR coordinate respectively. T_{lidar}^{global} is the homogeneous transform from LiDAR coordinate to global coordinate.

Increasing frame rate. For the MOTR 2D tracker, the frame rate of the dataset can reach 20hz, and the pedestrian movements between two frames are small enough to have IoU. However, in the NuScenes dataset, the LiDAR scans at 2 FPS and 20 FPS, and the ground truth 3D box annotations are only provided in keyframes at 2 FPS. Moreover, the vehicle movements are much larger than pedestrians. Therefore, it is possible for the vehicles to have no IoU between frames at a low frame rate, which causes difficulty for the Transformer to learn the motion trend and spatial-temporal relation of objects. In this case, increasing the frame rate in the NuScenes dataset is necessary to simplify the tracking.

First, We assume that the fastest speed of cars is 30 m/s and the average length of cars is 4 m. According to the Tab. 3.1, we note that cars can move 15 m at 30 m/s between two frames at 2 FPS, the displacement much larger than the car's length, which means there is no IoU between two frames. While too high frame rate interpolation decreases the accuracy and too low frame rate interpolation is still hard to track the cars. Therefore, we decide to apply a linear interpolation to estimate the location of the boxes at the intermediate frames to 10 FPS, which ensures most of the cars have IoU between frames.

Displacement / FPS / Speed	2 FPS	10 FPS	20 FPS
10 m/s	5 m	1 m	0.5 m
20 m/s	10 m	2m	1 m
30 m/s	15 m	3m	1.5 m

Table 3.1: The comparison of displacement among three different frame rates, 2 FPS, 10 FPS and 20 FPS, depending on the speed 10 m/s, 20 m/s and 30 m/s.

3.3 Backbone of MOTPT

The point-based method provides the precise location information and 3D point relationship in the point cloud compared with the voxel-based method. However, the point-based method's random or farthest point sampling cannot efficiently extract instances from the point cloud. In contrast, IA-SSD can sample the foreground points better and faster than other PointNet-based methods due to the class-aware sampling and centroid-aware sampling, two downsampling strategies that can distinguish background and foreground points. Therefore, MOTPT uses IA-SSD as the backbone to downsampling and extract the feature of the point cloud. The structure of our backbone is shown in Fig. 3.3.

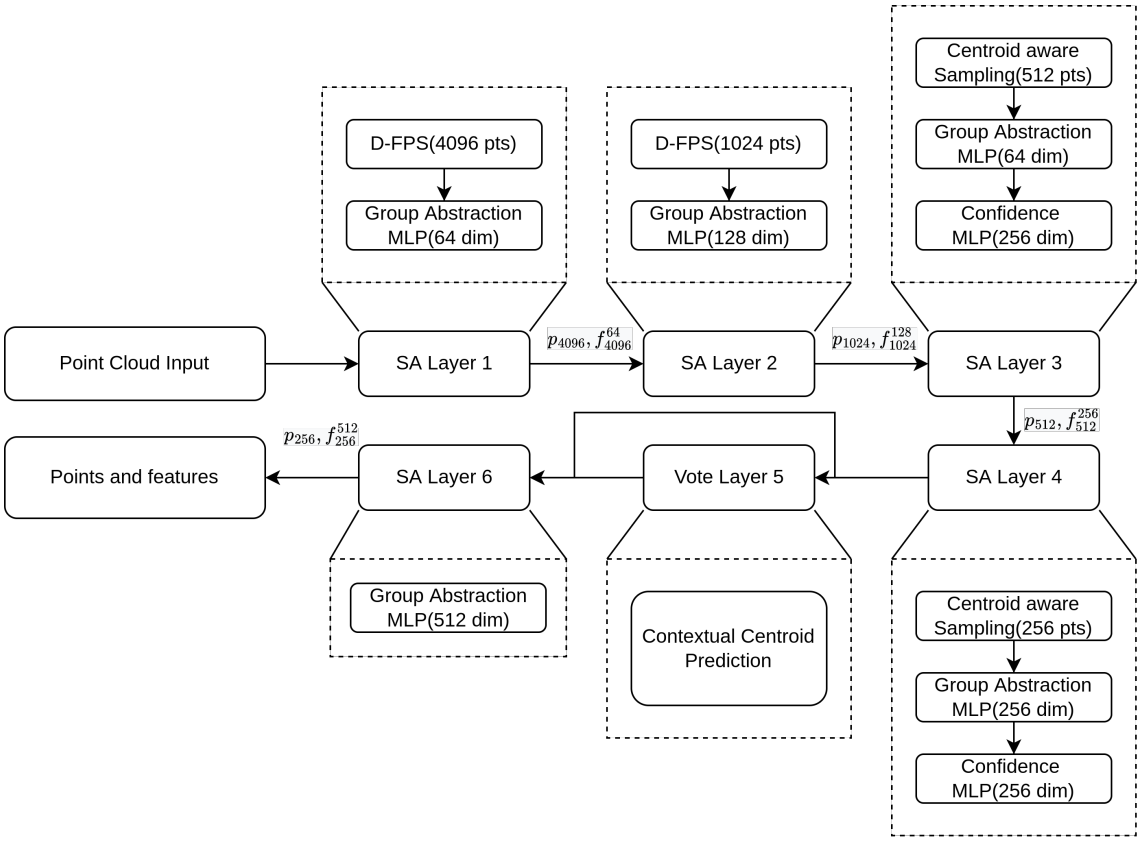


Figure 3.3: The backbone of MOTPT.

The point cloud is initially sent into 5 different set abstraction (SA) layers and a vote layer to extract features. First, we use distance-based furthest point down-sampling(D-FPS) and centroid-aware down-sampling to decrease memory and computation costs. The vote layer will vote on the center point of the target to which they belong so that many vote points close to the target can be obtained, and bounding box proposals are proposed on the vote point. The center information of 3D bounding boxes is then obtained by feeding the conserved representative points into the contextual centroid perception module. There are 4 down-sampling layers, we will get the information of 4096, 1024, 512 and 256 points respectively from each down-sampling layer including points coordinates information $\{p_{4096}, p_{1024}, p_{512},$

$p_{256}\}$ and the features of the $\{f^{64}_{4096}, f^{128}_{1024}, f^{256}_{512}, f^{512}_{256}\}$, where f^{dim}_{n} represents the number of point is n and its dimension of point feature is dim. The encoder takes p_{1024} and f^{128}_{1024} as the key of the Transformer because the Transformer requires much high memory and calculation cost and for the large scale of point. While the decoder takes p_{256} and f^{512}_{256} as query inputs. The subscripts here represent the number of points after each downsampling, and each point represents one object, but it is not the center of the bounding box, just a point within the box.

The performance of the IA-SSD backbone is shown in Fig. 3.4, the downsampling candidate points lie in most vehicles thanks to the centroid-aware method.

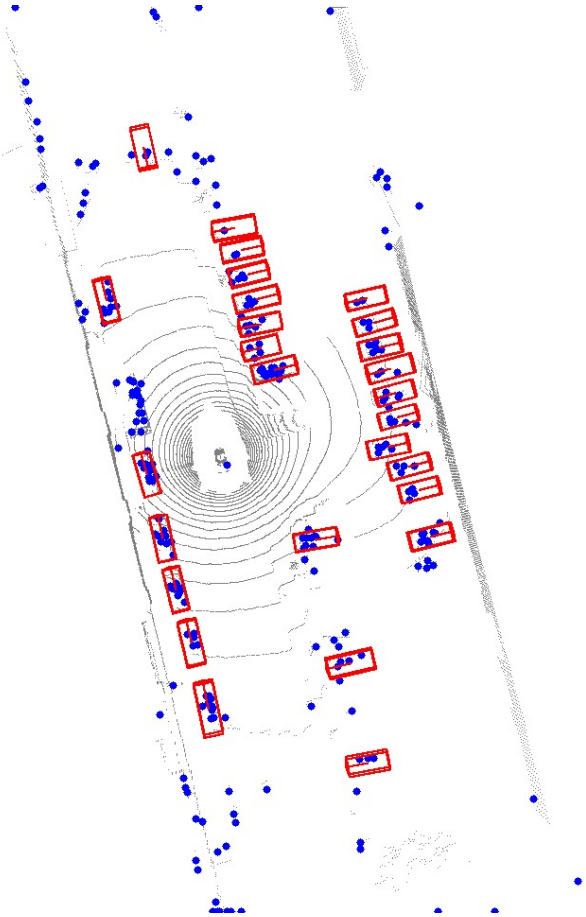


Figure 3.4: The down-sampling result used the IA-SSD backbone. The red bounding boxes are the vehicle ground truth, and the blue points are down-sampling points p_{256} from the IA-SSD backbone.

3.4 Detection of MOTPT

The Transformer of our model is modified from 3DETR. The encoder-decoder architecture allows our model to learn the long-term spatial-temporal relations among objects crossing multiple frames of point clouds in an end-to-end fashion. The structure of the MOTPT Transformer is shown in Fig. 3.5. The encoder of the

Transformer takes the points and the features obtained from the above backbone at frame t, p^t_{1024} and $f^{128,t}_{1024}$ as input to get the self-attention features.

To be specific, the features of 1024 points $f^{128,t}_{1024}$ are used as value V_{enc} of the encoder, while the features added with the Fourier positional embeddings of 1024 points p^t_{1024} is regarded as both key K_{enc} and query Q_{enc} of the encoder. The expressions of Q_{enc}, K_{enc} and V_{enc} are shown in Eq. 3.5, where $PE(.)$ is the Fourier positional embedding function

$$\begin{aligned} V_{enc} &= f^{128,t}_{1024} \\ K_{enc} &= f^{128,t}_{1024} + PE(p^t_{1024}) \\ Q_{enc} &= f^{128,t}_{1024} + PE(p^t_{1024}). \end{aligned} \tag{3.5}$$

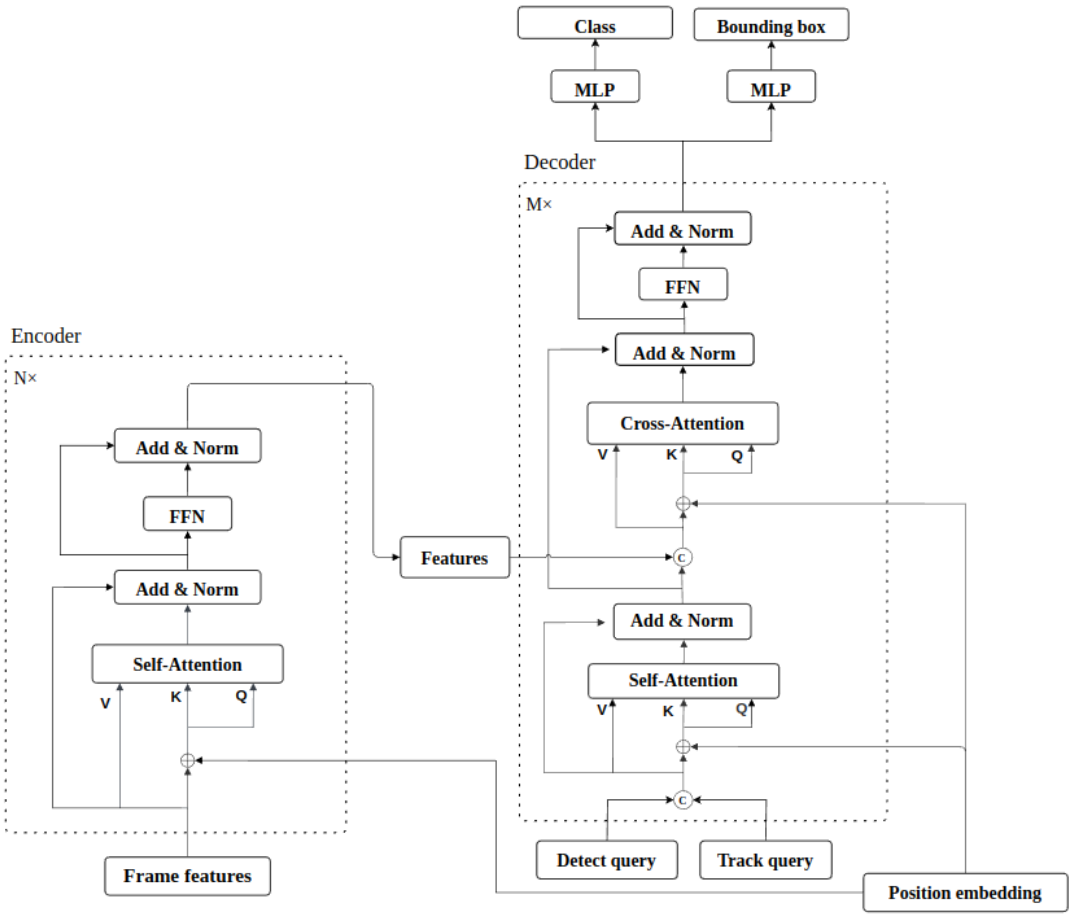


Figure 3.5: The structure of MOTPT Transformer.

The decoder contains two parts, one is the self-attention module, and the other is the cross-attention module. The input of the decoder module is a concatenated vector of detect query q_{det} and track query q_{tr}. Detect query q_{det} contains the 256 sampled points p^t_{256} and point feature $f^{512,t}_{256}$ in the frame t, and track query contains points p^{t-1}_{tr} and the feature $f^{512,t-1}_{tr}$, where tr is the number of tracked query from

last frame $t-1$. From the beginning frame, there is no tracked query, and therefore, tr is 0.

The value V_{dec} of decoder module is a concatenated vector of detect query q_{det} and track query q_{tr}, while the key K_{dec} and query Q_{dec} of dec module and are both this vector added with the Fourier positional embedding of queries, we get

$$\begin{aligned} V_{dec} &= \text{Concat}(f_{256}^{512,t}, f_{tr}^{512,t-1}) \\ K_{dec} &= \text{Concat}(f_{256}^{512,t} + PE(p_{256}^{t}), \quad f_{tr}^{512,t-1} + PE(p_{tr}^{t-1})) \\ Q_{dec} &= \text{Concat}(f_{256}^{512,t} + PE(p_{256}^{t}), \quad f_{tr}^{512,t-1} + PE(p_{tr}^{t-1})). \end{aligned} \tag{3.6}$$

Here, the detect query q_{det} aims to represent the detection of the objects in the current frame, which is $f_{256}^{512,t}$ the output from the backbone at frame t. While track query q_{tr} is the tracked object from previous frames. Note that q_{tr} is empty at the first frame, and this size of q_{tr} depends on the number of the tracked objects previously.

Finally, the decoder gets the output embedding $embed_{out}$, and MLP is leveraged to predict the class label, location, size, and orientation (bin class and residual angle) of objects from $embed_{out}$.

To make the training task easier and predicted location more precise, we predict the offset of the location from $embed_{out}$ and deal with the location of detect queries and track queries separately, shown in Fig. 3.6.

For the candidate points of detect query, p_{256} represents 256 candidate points of 256 proposals. The centers of the detection bounding boxes C_{det} are the p_{256} plus the detect offset d_{det} predicted by MLP of detect-offset MLP_{do} after the decoder

$$C_{det} = p_{256} + d_{det} = p_{256} + MLP_{\text{do}}(embed_{out}) * \tau_{do} \tag{3.7}$$

where τ_{do} is the upper boundary of d_{det}, which usually is set to half of the car length because the distance from p_{256} to C_{det} is not larger than half of the car length.

For the track query, track-offset MLP MLP_{to} is leveraged to predict the motion offset. The center of tracked bounding box C_{tr}^{t} in the current frame t is the center of tracked bounding box C_{tr}^{t-1} in the previous frame $t-1$ plus the track offset d_{tr}^{t-1}

$$C_{tr}^{t} = C_{tr}^{t-1} + d_{tr}^{t-1} = C_{tr}^{t-1} + MLP_{\text{to}}(embed_{out}) * \tau_{to}. \tag{3.8}$$

The d_{tr} is the normalized prediction of MLP_{to} multiply the τ_{to}. Moreover, the d_{tr} represents the vehicle motion displacement, which has upper boundary τ_{to}.

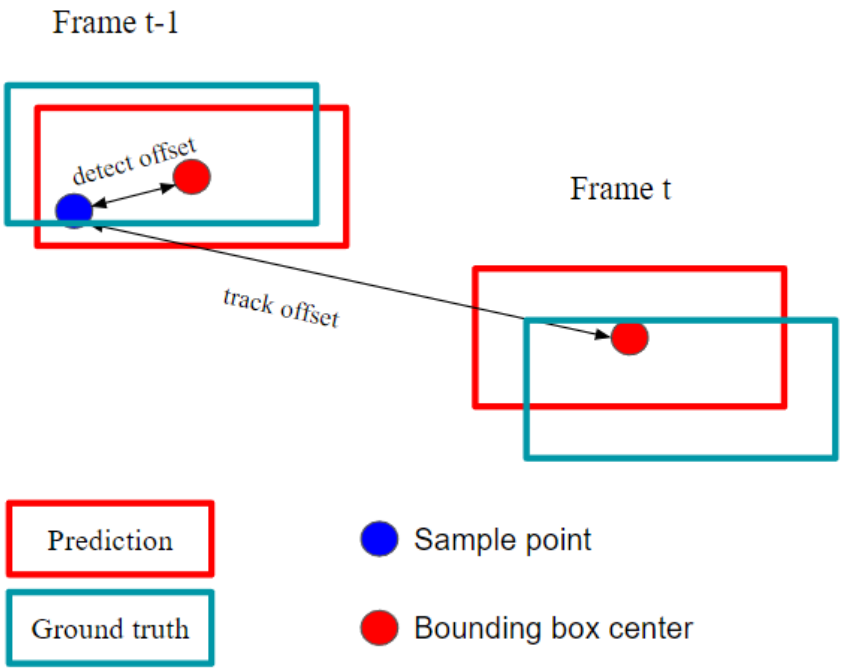


Figure 3.6: The MLP prediction of offset for detect query and track query. Detect-MLP is used to predict the detect offset from sampled point to the center of the bounding box in frame $t-1$. Another Track-MLP is used to predict the track offset from sampled point in frame $t-1$ to the center of the bounding box in frame t.

3.5 Label assignment in training

In the training part, the target-consistent label assignment will be separated into two parts: new-born objects assignment σ_{new} and tracked objects assignment σ_{tr}. An example of the association strategy between two frames is shown in Fig. 3.7. The σ_{new} is the matching correspondence that uses the Hungarian algorithm to execute bipartite matching between the obj_{new} and its correspondent ground truth GT_{det}. The detection assignment processing is shown in the Eq. 3.9

$$\sigma^i_{new} = \arg\max \mathcal{L}_{Hungarian}(obj^i_{new}, GT^i_{det}) \tag{3.9}$$

where $\mathcal{L}_{Hungarian}$ is the loss of Hungarian algorithm determined by a combination of three components: IoU, class and center distance, and the $\mathcal{L}_{Hungarian}$ defines as

$$\begin{aligned}\mathcal{L}_{Hungarian}(obj^i_{new}, GT^i_{det}) = \lambda_{iou}\mathcal{L}_{iou}(obj^i_{new}, GT^i_{det}) + \lambda_{cls}\mathcal{L}_{cls}(obj^i_{new}, GT^i_{det}) \\ + \lambda_{cent}\mathcal{L}_{cent}(obj^i_{new}, GT^i_{det})\end{aligned} \tag{3.10}$$

where $\mathcal{L}_{iou}$ and $\mathcal{L}_{cent}$ is to calculate the IoU and center distance of detection and ground truth, respectively. $\mathcal{L}_{cls}$ is same as the class loss in DETR shown in Sec. 2.4.1.

The σ_{tr} is the matching correspondence that associating the tracked objects obj_{tr} with its correspondent tracked ground truth GT_{tr}. At the frame $t-1$, the obj^{t-1}_{tr}

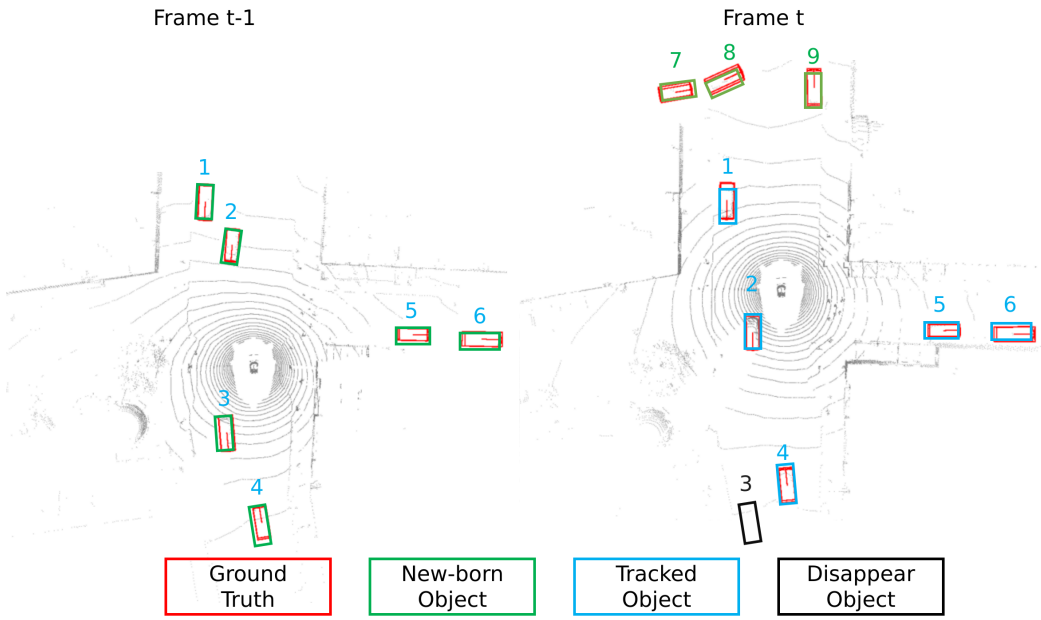


Figure 3.7: Association strategy of MOTPT Transformer. Left: Hungarian association for new-born objects obj_{new}^{t-1} assignment in frame $t-1$. The obj_{new}^{t-1} (green boxes) come from detection of model, and they are assigned with its correspondent ground truth GT_{new} by Hungarian association in frame $t-1$. Right: Tracked ID association for tracked objects obj_{tr}^{t} assignment in frame t. The obj_{tr}^{t} (blue boxes) come from the motion prediction of obj_{tr}^{t-1}. obj_{tr}^{t}, associate with its correspondent ground truth GT_{tr}^{t} based on their track IDs. The disappear objects obj_{dis}^{t}(black boxes) associated with empty object and count their disappear time.

is empty, and generate the obj_{new}^{t-1}, then, the obj_{tr}^{t} in frame t come from obj_{new}^{t-1} is assigned to GT_{tr}^{t}. In the next frame, the obj_{tr}^{t+1} in frame $t+1$ come from obj_{tr}^{t} is assigned to GT_{tr}^{t+1}. If obj_{tr} which is not matched with any ground truth GT, then it is regarded as disappear objects obj_{dis}, assigned as empty objects, and counts the disappear time T_{dis}. When T_{dis} is greater than disappear threshold τ_{dis}, remove obj_{dis} and set it as obj_{exit}.

The tracked objects assignment processing is shown as

$$\sigma_{tr}^{t} = \begin{cases} \emptyset & i = 0 \\ \sigma_{tr}^{i-1} \cup \sigma_{new}^{i-1} & 2 \leq i \leq N \end{cases} \tag{3.11}$$

where N is the length of the point cloud sequence.

3.6 Refinement for tracking prediction

The query output from Transformer serves as the original track query q_{otr}, and it is fed into the spatial-temporal module (STM) to refine the q_{otr} and filter the track query set. In STM, a spatial module and a temporal module update the $embed_{out}$

for modeling the implicit motion model, which is shown in Fig. 3.8.

The temporal attention updates the $embed_{out}$ by calculating the attention with the previous output embedding $embed_{out}^{t-1}$ and current output embedding $embed_{out}^{t}$. The key and value of self-attention are $embed_{out}^{t-1}$, and the query is $embed_{out}^{t}$, which represents the relationship of $embed_{out}$ in time dimension.

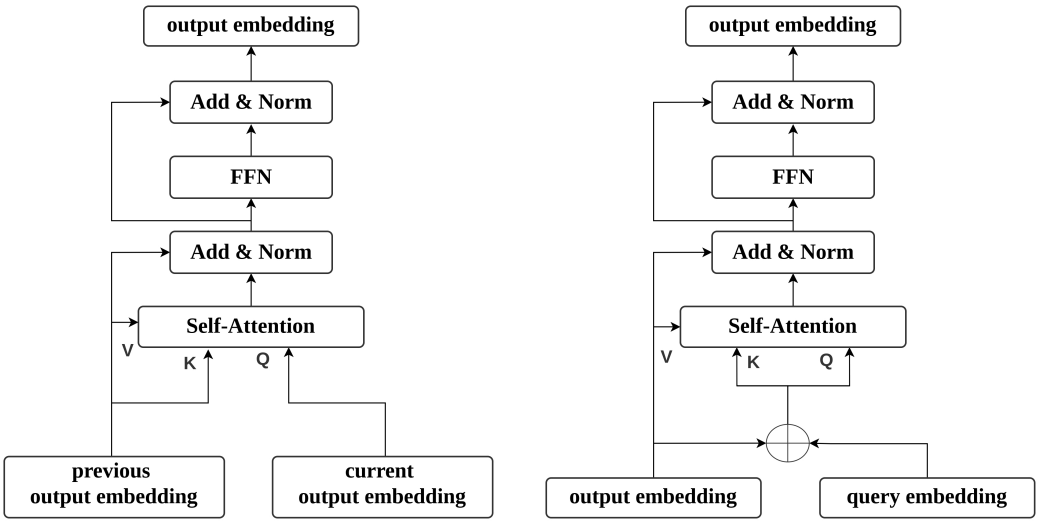
Then spatial attention updates the $embed_{out}$ by calculating the attention with the query embedding $embed_q$ and the $embed_{out}$, where query embedding defines as concatenating the detect query and the tracked query. The key and query of self-attention are concatenating $embed_q$ and the $embed_{out}$, and the value is $embed_{out}$, which combines the query embedding relationship for $embed_{out}$.

After temporal and spatial attention, the STM also predicts a tracking score s_{STM} for tracked objects in training mode and calculates the loss of tracking score with the labels in GT to evaluate whether the $embed_{out}$ updated by STM is reasonable.

In the training mode, the STM module only keeps the q_{otr} objects whose IoU with the GT is larger than the active track threshold τ_{active} and saves the filtered q_{otr} as q_{tr}, and it can be expressed as

$$s_{STM} = FC(embed_{out}) \tag{3.12}$$

where FC is fully connected layers of tracking score.



(a) temporal attention (b) spatial attention

Figure 3.8: Spatial attention and temporal attention.

3.7 End-to-end training

The whole pipeline is trained in an end-to-end fashion. The total loss $\mathcal{L}_{total}$ contains down-sample strategy loss $\mathcal{L}_{downsampling}$, centroid prediction loss $\mathcal{L}_{cent}$, multi-layers of box prediction loss for detect query and track query $\mathcal{L}_{box}$, the loss of track score $\mathcal{L}_{STM}$ for STM module, class prediction loss $\mathcal{L}_{cls}$ and "no object" loss $\mathcal{L}_{no}$

$$\mathcal{L}_{total} = \lambda_{sam}\mathcal{L}_{sample} + \lambda_{cent}\mathcal{L}_{centriod} + \mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{no}\mathcal{L}_{no} + \lambda_{tr}\mathcal{L}_{STM}. \quad (3.13)$$

Furthermore, the $\mathcal{L}_{box}$ can be separated into location loss $\mathcal{L}_{loc}$, size loss $\mathcal{L}_{size}$, angle-bin loss $\mathcal{L}_{angle-bin}$, angle-residual loss $\mathcal{L}_{angle-res}$, and corner loss $\mathcal{L}_{corner}$

$$\mathcal{L}_{box} = \lambda_{loc}\mathcal{L}_{loc} + \lambda_{size}\mathcal{L}_{size} + \lambda_{bin}\mathcal{L}_{angle-bin} + \lambda_{res}\mathcal{L}_{angle-res} + \lambda_{corner}\mathcal{L}_{corner} \quad (3.14)$$

where all the coefficients of these losses are the corresponding weights, reflecting the importance of different loss functions to the training. The corner loss helps to constrain the box prediction further. Specifically, the auxiliary loss for multi-layer box prediction is used to calculate the loss for each layer in the Transformer, which helps the model output the correct number of objects of each class.

Like 3DETR, use $L1$ loss for location and size loss, cross entropy loss for angle-bin loss and Huber loss [15] for angle-res loss. Because of the unbalanced number of ground truth and empty prediction objects, use the focal loss [19] for class prediction.

In order to improve the training efficiency and stabilization, it is better to gradually increase the number of training frames for each training clip, because train with a small number of training frames in one clip is fast and stabilizes the weight of the tracker, and then increase the number of training frames help to capture the long-term relationship.

3.8 Inference

In the inference, the Transformer module outputs the predict scores s_{pred}, predict boxes box_{pred} of query, but there is still no IDs assign for the box_{pred}. There are two methods to assign the ID for predict boxes to infer the final tracked result

- Score filter method, using the predict score to distinguish the new-born objects obj_{new}, disappear objects obj_{dis}, and exit objects obj_{exit} directly.
- Post matching method, using the Hungarian matching to associate objects.

3.8.1 Score-filter method

For inference using score-filter method, the new-born objects obj_{new} are defined as the objects of detect queries q_{det} whose predict scores s_{pred} are greater than the new-born threshold τ_{new}, we get

$$obj_{new} = \{q_k \in q_{det} |\ s_{pred,k} > \tau_{new}\} \quad (3.15)$$

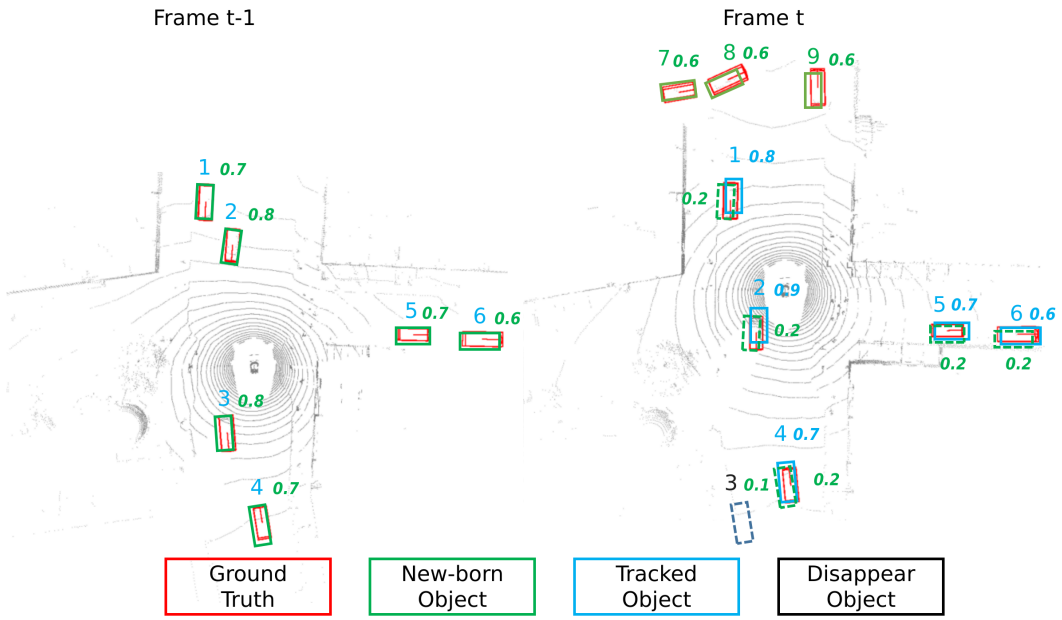


Figure 3.9: The score filter inference of MOTPT. We set $\tau_{\text{new}} = 0.6$, $\tau_{\text{dis}} = 0.5$. The new-born objects obj_{new}^{t-1} 1-6 in frame $t-1$ will be predicted in a high score (larger than 0.5). In frame t, obj_{new}^{t-1} updates as tracked objects obj_{tr}^{t}, and other score of duplicated detection on obj_{tr}^{t} will be suppressed(green boxes 1, 2, 4, 5, 6). The disappear objects obj_{dis}^{t}(black boxes 3) whose scores lower than τ_{dis} counts its disappear time.

where $s_{pred,k}$ is the predict score for the kth object in detect queries.

The disappear objects obj_{dis} are the of track objects q_{tr} whose scores s_{tr} are lower than the disappear threshold τ_{dis}. Moreover, the disappear objects will be filtered, and exit if disappear time is greater than exit threshold τ_{ex} for the consecutive N frames. It can be expressed as

$$\begin{aligned} obj_{dis} &= \{q_k \in q_{tr} |\ s_{pred,k} < \tau_{dis}\} \\ obj_{exit} &= \{s_{pred,k} > \tau_{ex}\} \end{aligned} \tag{3.16}$$

where $s_{pred,k}$ is the score of the k^{th} track query.

An example of the score filter inference result is shown in Fig 3.9. The new-born objects obj_{new}^{t-1} 1-6 in frame $t-1$ will be predicted in a high score (larger than 0.5). In frame t, obj_{new}^{t-1} updates as tracked objects obj_{tr}^{t}, and other score of duplicated detection on obj_{tr}^{t} will be suppressed(green boxes 1, 2, 4, 5, 6). The disappear objects obj_{dis}^{t}(black boxes 3) whose scores lower than τ_{dis} counts its disappear time.

3.8.2 Post-matching method

Another proposal is the post-matching method, which uses the Hungarian matching method to associate the detected objects and tracked objects from the Transformer

output. Although the bipartite matching output can suppress the duplicate detection for the same vehicle, the performance of suppression is still not stable, especially for the low-score objects near the boundary of LiDAR detection, which are easily mistaken as duplicated objects and removed. So, in order to make good use of the low score detection and improve the detection performance, it is necessary to apply NMS to remove the extra and overlapping bounding boxes to get filtered detection objects after NMS obj_{det}. At frame t, the tracked objects obj_{tr}^{t} come from motion prediction of previous detect objects obj_{det}^{t-1} or previous tracked objects obj_{tr}^{t-1}. Then apply Hungarian match between obj_{det}^{t} and obj_{tr}^{t-1} to get the associated correspondence. After Hungarian match, we need to filter some unreasonable correspondence whose distance are larger than the distance threshold τ_{match} to get filtered associated correspondence $\sigma_{det,tr}^{t}$

$$q_{\text{det\_match}}, q_{\text{tr\_match}} = \{d_i \in \sigma_{\text{det,tr}} |\ d_i < \tau_{\text{match}}\}. \tag{3.17}$$

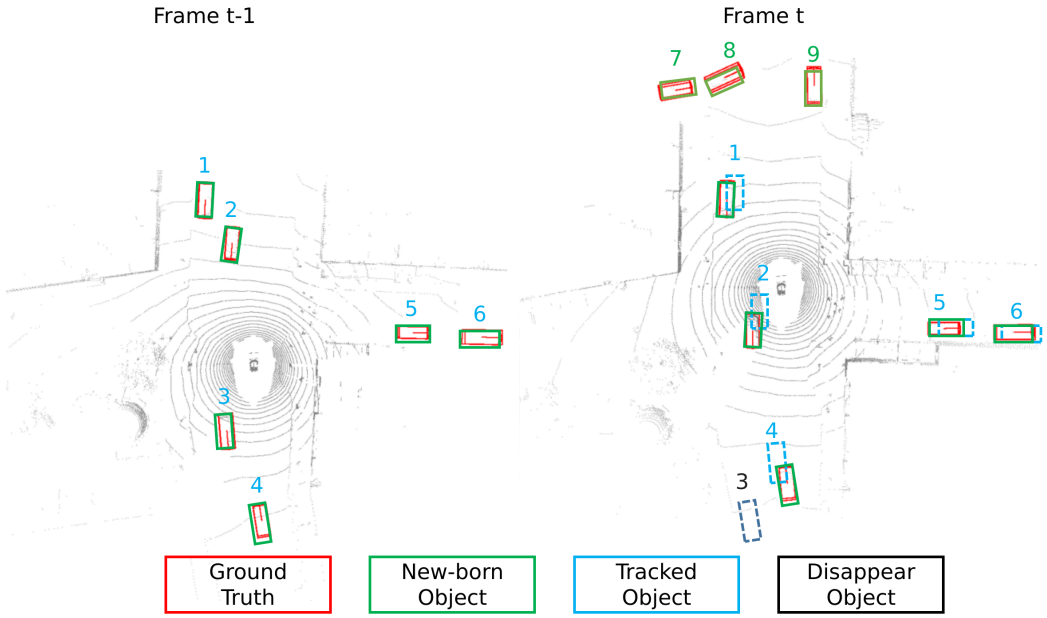


Figure 3.10: The post-process inference of MOTPT, Left: detection objects obj_{det}^{t-1} 1-6 in frame $t-1$. Right: In frame t, Tracked objects obj_{tr}^{t} come from motion prediction of obj_{det}^{t-1}. The obj_{tr}^{t} 1, 2, 4, 5, 6 have association with obj_{det}^{t-1} use Hungarian matching, and assign their track ID to their correspondent detection obj_{det}^{t} 1, 2, 4, 5, 6. Tracked objects 3 have no associated detection, which is defined as disappear object obj_{dis}^{t}, and count its disappear frame. Detect objects obj_{det}^{t} 7,8,9 have no correspondent tracked objects, which are regarded as new-born objects obj_{new}^{t}.

Based on the filtered association correspondence $\sigma_{det,tr}$, in frame t, we assign the track ID of the obj_{tr}^{t-1} to their correspondent obj_{det}^{t}, because we trust the detection compared with track prediction more. The new-born objects obj_{new}^{t} define as the unmatched correspondences in the obj_{det}^{t}, and disappear objects obj_{dis}^{t} define as the unmatched correspondence in the obj_{tr}^{t-1} and count its disappear frames. When the number of the disappear frames is larger than the exit threshold τ_{ex}, then it will

be regarded as the exit objects obj^t_{exit} and removed from the tracked objects. An example of the post-process matching inference result is shown in Fig 3.10.

3.9 Implementation detail

Our MOTPT tracker is trained in the OpenPCDet [38] framework, which is a code library for point cloud 3D target detection based on the PyTorch implementation. It adopts the top-level code framework design idea of data-model separation, allowing researchers to easily adapt various models to different point cloud 3D object detection datasets, eliminating the concern of getting lost in the 3D coordinate transformation when developing models.

Our model is trained with 8 A100 GPUs with 1 batch size, using AdamW[21] as optimizer, initial momentum $\rho = 0.9$, initial learning rate $\lambda = 0.01$, and weight decay $w_{dc} = 0.01$, which is fast-converge. Since the NuScenes contains a large number of training frames and uses interpolation to increase the frame frequency from 2 FPS to 10 FPS, it costs a lot of training time to train one epoch. Therefore, we only train the model with small epochs (30) and gradually increase the number of training continuous frames from 3 to 5 every 10 epochs.

4
Results

4.1 Detector performance

First, we implement the IA-SSD detector on the NuScenes dataset and find that it is an efficient and accurate detector. Then, we compare the feature of 256 points f_{256} of different dimensions 256 and 512 (f_{256}^{256} and f_{256}^{512}), which causes little different performances shown in Tab. 4.1.

Dimension	Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
256	car	0.677	0.176	0.144	0.131	2.010	0.505
512	car	0.678	0.176	0.145	0.126	2.026	0.503

Table 4.1: The detection performance of IA-SSD.

We train our MOTPT tracker model for several steps to improve training efficiency. First, we pre-trained a detector as initialization of MOTPT, which can dramatically speed up the overall training process. The detector model is formed by combining the IA-SSD backbone and the 3DETR Transformer, the single frame training model of MOTPT.

In order to explore the optimal parameters of the detector, we choose relatively good performance parameters as the control group, which only changes one parameter respectively in follow experiments. The parameters of the control group shows in Tab. 4.2, and its performance shows in Tab. 4.3.

Tune detector layers. In order to improve the detector performance, we explore the influence of different parameters on the Transformer module, including the number of encoder layers and decoder layers of the Transformer. Tab. 4.4 shows the detection performances of the different number of encoder layers and decoder layers. We do the ablation of the encoder and set its number of layers as 0 first, and the result shows that the encoder is vital. The module cannot work for this model without an encoder because it is necessary to use self-attention to capture the relationship of point features between different downsampling layers. Meanwhile, increasing the number of encoder layers improves the average accuracy AP of the detector, while increasing the number of decoder layers only slightly improves the detector's performance. Moreover, the detector is the heaviest part of the model, which costs the most GPU memory in training. To balance the model efficiency and accuracy, we

Transformer	
Encoder layer	6
Encoder head	4
Encoder dimension	128
Encoder FFN dimension	128
Decoder layer	2
Decoder head	4
Decoder dimension	512
Decoder FFN dimension	512
Weight of matcher	
λ_{iou}	2
λ_{cls}	1
λ_{cent}	1
Weight of total training loss	
λ_{cls}	2
λ_{no}	5
λ_{bin}	1
λ_{res}	1
λ_{loc}	10
λ_{size}	0.1
λ_{corner}	0.05

Table 4.2: Control group parameters consist of three components: the Transformer parameter, matcher weights, and training weights.

Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
car	0.577	0.353	0.284	0.201	2.119	0.505

Table 4.3

decide to use 6 encoder layers and 2 decoder layers in the detector model.

N	M	Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
0	2	car	0.038	0.739	0.298	1.557	1.873	0.425
2	2	car	0.573	0.342	0.282	0.197	2.108	0.505
4	2	car	0.574	0.350	0.284	0.202	2.097	0.503
6	2	car	0.577	0.353	0.284	0.201	2.119	0.505
6	4	car	0.577	0.352	0.284	0.208	2.153	0.509
6	6	car	0.578	0.349	0.284	0.200	2.141	0.508

Table 4.4: The detector performances of the different numbers of encoder layers and decoder layers, where N and M represents the number of encoder layers and decoder layers, respectively.

Tune loss weights. We also tune the optimal weight of the detector losses. From

the experiments shown in Tab. 4.5, it can be seen that the low λno decreases the detector's performance because it will greatly impact the prediction score and further influence the average precision. While changing single λ_{loc} or λ_{corner} has no great impact as λ_{no}, because $\mathcal{L}_{loc}$ and $\mathcal{L}_{corner}$ both are parameter of the prediction box, which can complement each other. Therefore, it is better to choose a larger λ_{no} and λ_{loc}, and a small λ_{corner}.

	Object class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
$\lambda_{no} = 0$	car	0.571	0.354	0.284	0.204	2.125	0.508
$\lambda_{no} = 1$	car	0.577	0.355	0.284	0.205	2.106	0.502
$\lambda_{loc} = 0$	car	0.575	0.355	0.284	0.205	2.120	0.504
$\lambda_{loc} = 1$	car	0.577	0.355	0.284	0.207	2.131	0.504
$\lambda_{corner} = 0$	car	0.573	0.352	0.284	0.203	2.130	0.508
$\lambda_{corner} = 1$	car	0.577	0.357	0.284	0.206	2.129	0.506

Table 4.5: The comparison of the importance of box loss weights to the detector performance.

Tune matcher weights. We explore the influence of weights of the Hungarian matcher on the detection performance shown in Tab. 4.6. The experiment's result shows that the $\lambda_{cls} = 0$ is much more important for the detector, which deserved to set a higher weight. $\lambda_{cent} = 0$ and $\lambda_{iou} = 0$ have almost the same impact because the most matched prediction of the vehicle also has the largest IoU and closest distance. Therefore, a reasonable choice of matcher weight is $\lambda_{cent} = 1$, $\lambda_{cls} = 2$, $\lambda_{iou} = 1$.

	Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
$\lambda_{cent} = 0$	car	0.575	0.354	0.284	0.202	2.120	0.506
$\lambda_{iou} = 0$	car	0.576	0.354	0.284	0.204	2.123	0.505
$\lambda_{cls} = 0$	car	0.571	0.352	0.284	0.205	2.120	0.506

Table 4.6: The comparison of the importance of Hungarian matching weights to the detector performance.

Feature dimension. Like implementing IA-SSD, we compare the feature of 256 points f_{256} of different dimensions 256 and 512 (f_{256}^{256} and f_{256}^{512}). The comparison results are shown in 4.7. The lower dimension of f_{256} in Transformer causes the main decrease in performance because the dimension of the backbone has little impact on the detector. Therefore, we choose to use 512 dimensional f_{256}^{512} because the higher dimension expresses a more complex feature.

Data augmentation. Moreover, We consider the impact of data augmentation on the detector effect. The augmented 3D point clouds commonly improve the ground truth density in training to increase the training speed and avoid overfitting. For the augmentation of the data in the detection task, we sample the ground truth data and balance the number of ground truth objects GT_{num} in all frames when GT_{num} is less

Dimension	Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
256	car	0.428	0.602	0.329	0.243	2.182	0.504
512	car	0.577	0.353	0.284	0.201	2.119	0.505

Table 4.7: The detection performance of different dimensional point feature.

than a threshold $\tau_{balance}$, and also add small perturbations, including random axis flips with 50% probability, random rotation around z-axis with an angle in the range $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ and random scaling of the scenes with a factor in the range $[0.95, 1.05]$. The experiment result for data augmentation with different $\tau_{balance}$ shows in Tab. 4.8.

$\tau_{balance}$	Object Class	AP↑	ATE↓	ASE↓	AOE↓	AVE↓	AAE↓
5	car	0.577	0.354	0.284	0.202	2.118	0.507
10	car	0.577	0.351	0.284	0.204	2.133	0.507
15	car	0.573	0.352	0.284	0.200	2.124	0.507

Table 4.8: The comparison of detection performance of different $\tau_{balance}$.

It can be seen that when $\tau_{balance}$ is a small number like 5, 10, then the AP of the detector is almost the same. However, the giant $\tau_{balance}$ degrades the detector's performance since this data augmentation strategy creates some irrational ground truth instances in the point cloud. At the same time, NuScenes is a sizable dataset, ensuring sufficient data to prevent overfitting. Therefore, balancing the number of ground truths in NuScenes detector training is unnecessary.

In brief, our detector performance is worse than the IA-SSD detector. The potential reason is that IA-SSD uses the one-to-many matching method, and our detector uses the one-to-one matching to train. However, due to insufficient supervision due to unbalanced foreground information, the performance of one-to-one matching methods is inferior to that of traditional detectors trained with one-to-many label [56].

4.2 Tracker training

To speed up the MOTPT tracker training, we freeze the IA-SSD point-based backbone and use the pre-trained detector model to preliminary detect the vehicle for a single frame.

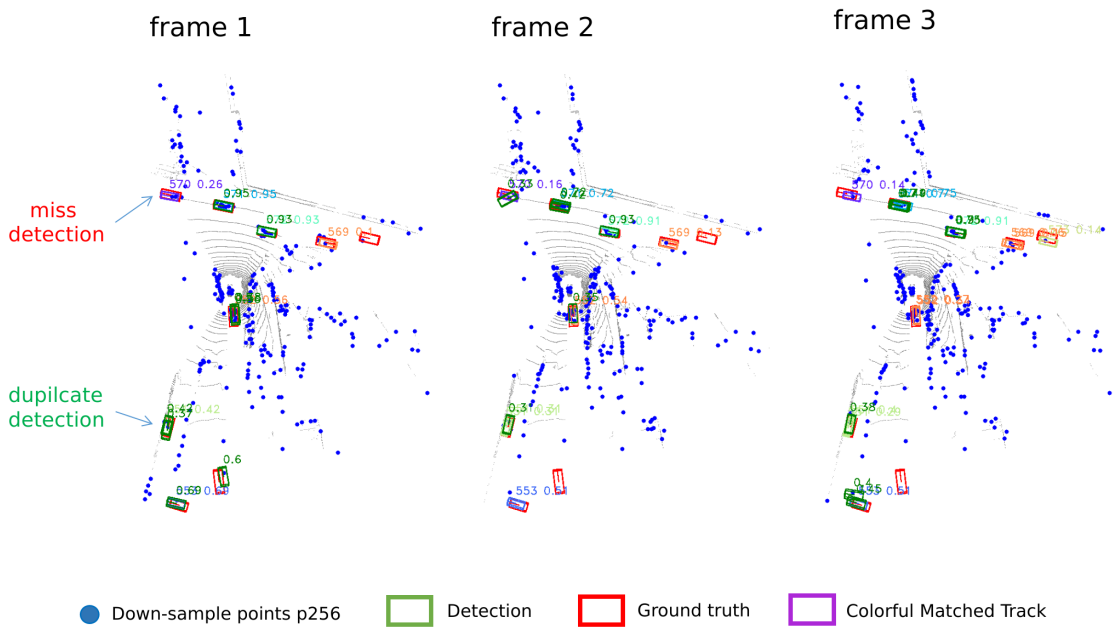


Figure 4.1: Three consecutive frames in the 10 FPS tracking training. Some objects, only have red boxes but no green boxes, which means that the detection is missed. While some objects, they have more than one green detection box, that's the duplicate detection.

In the training mode, in order to predict track offset d_{tr} correctly, we set a maximum τ_{to} to boundary it in x, y, z dimensions between two frames. Because an object cannot move too far between two frames, this offset will be considered with the frame rate. In 2 FPS tracking, we set $\tau_{to} = (10, 10, 2)$, and 10 FPS tracking, $\tau_{to} = (4, 4, 1)$, because the vehicles move longer between in the lower frames. In addition, there might be several detection bounding boxes for the same object, and a detection score threshold τ_d of 0.3 is set to filter low score boxes. However, duplicate detection still occurs since there might be more than one high-scoring box for one object, and it may cause missing detection since the only detection for the object has poor quality with a score lower than 0.3, as shown in Fig. 4.1. This problem mainly affects the tracking continuity, making it easy to cause ID switches. Meanwhile, the duplication is more likely to happen for the detection on the point cloud boundary. Therefore, to remove the duplicated detection, NMS is needed to solve this problem in inference.

4.3 Tracker inference

4.3.1 Score-filter method

In practice, we find that the tracking performance of inference using the score-filter method is terrible. The reason is that it is hard for the Transformer to suppress the score duplicate detection, although we assign the duplicate detection to the no object class and punish it with a high no object loss λ_{no}. In this case, when there is duplicate detection, there are duplicated tracks, which dramatically increases the

ID switch and false positive. Meanwhile, the new objects far from LiDAR will be detected with relatively low scores, so they will not be regarded as new objects. This situation increases the false negative. Hence, the score-filter method is sensitive to the score threshold τ_{new} and τ_{dis}.

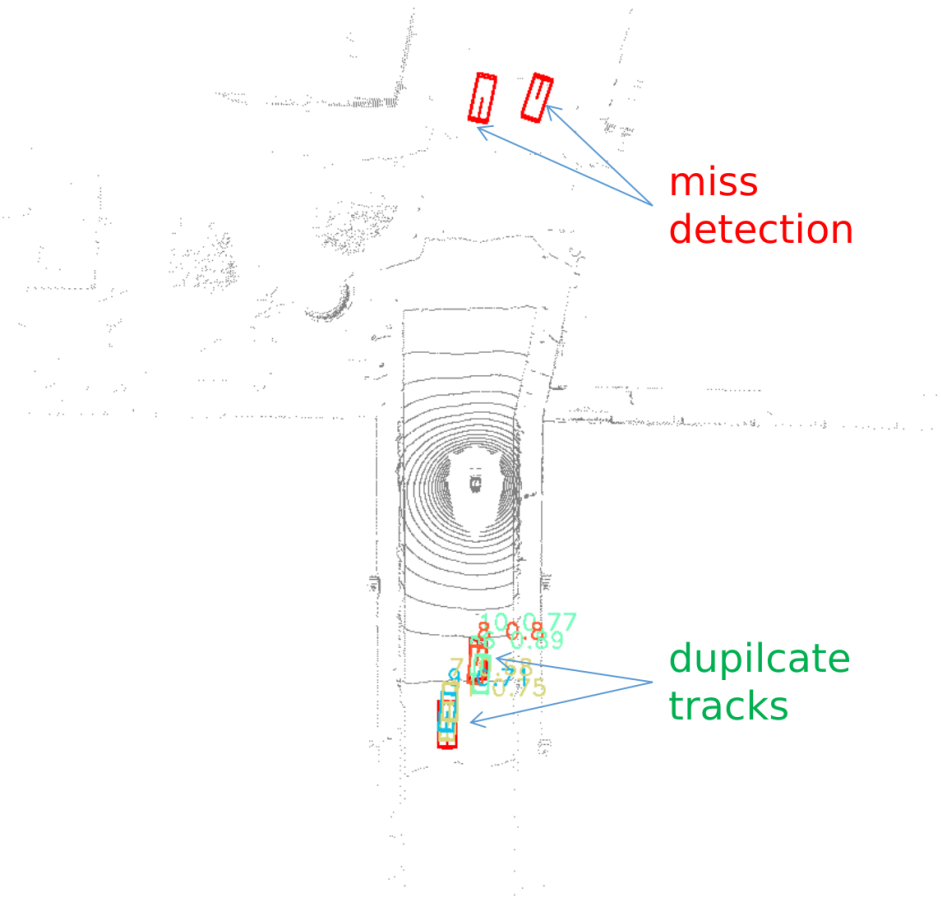


Figure 4.2: An example of inference using the score filter method in 10 FPS scene without ego-motion. $\tau_{new} = 0.6$, $\tau_{dis} = 0.5$. The duplicate tracks occur because of the similar detection score, and the ground truth far from LiDAR predicts a low score and causes miss detection.

4.3.2 Post-matching method

Post-matching is a method that rely on two key parameters: τ_{match} and τ_{ex}. In addition, the frame rate of the point cloud, whether using STM and whether removing ego-motion impact the tracking performance. In order to explore the impact of these elements, we also set a control group. Tab. 4.9 control group's condition and its tracking performance in both 2 FPS and 10 FPS scenes.

Tune parameters. The post-matching method thresholds τ_{match} and τ_{ex} greatly impact the tracking performance. For τ_{match}, if it is too small, the situation that some tracked objects occluded by others for several frames are terminated in advance, and some tracked objects may be regarded as new-born objects more often.

FPS	Ego	STM	τ_{match}	τ_{ex}	AMOTA↑	AMOTP↓	RECALL↑	FP↓	FN↓	IDS↓
2	×	√	10	4	0.391	1.315	0.512	5156	28432	3309
10	×	√	5	4	0.478	1.206	0.546	5790	26501	1296

Table 4.9: The tracking performance of two cases in 2 FPS and 10 FPS. For 2 FPS, τ_{match} is set to 10 and τ_{ex} is set to 4. For 10 FPS, τ_{match} is set to 5 and τ_{ex} is set to 4.

Meanwhile, τ_{match} also can not be tremendous. Otherwise, disappear objects easily produce false matching with other detection. Both of these situations lead to a high-level ID switch. The comparison of different τ_{match} in 2 FPS and 10 FPS scenes shows in Tab. 4.10. According to the results, we set $\tau_{match} = 15$ in 2 FPS and $\tau_{match} = 5$ in 10 FPS.

	τ_{match}	AMOTA↑	AMOTP↓	RECALL↑	FP↓	FN↓	IDS↓
2 FPS	5	0.344	1.254	0.508	9421	28715	3060
	10	0.391	1.315	0.512	5156	28432	3309
	15	0.392	1.315	0.513	5475	28402	3358
10 FPS	2	0.314	1.398	0.471	6101	30839	4542
	5	0.447	1.276	0.530	4493	27391	1674
	8	0.447	1.273	0.531	4560	27350	1734

Table 4.10: The comparison of different τ_{match} in 2 FPS and 10 FPS scenes.

For τ_{ex}, it depends on the stabilization of detection. Small τ_{ex} causes the disappear objects to exit easily, and when these disappear objects re-appear, they will be regarded as new objects, which increases the ID switch. While large τ_{ex} leads to the exit objects still being kept in some frames, which increases the false positive. The comparison of different τ_{ex} in 2 FPS and 10 FPS scenes shows in Tab. 4.11. According to the results, we set $\tau_{ex} = 4$ in 2 FPS and $\tau_{ex} = 6$ in 10 FPS.

	τ_{ex}	AMOTA↑	AMOTP↓	RECALL↑	FP↓	FN↓	IDS↓
2 FPS	2	0.349	1.252	0.534	10705	27170	3278
	4	0.391	1.315	0.512	5156	28432	3309
	6	0.273	1.222	0.506	13094	28799	3010
10 FPS	2	0.446	1.274	0.502	3328	29034	1287
	4	0.478	1.206	0.546	5790	26501	1296
	6	0.479	1.206	0.546	5692	26480	1289

Table 4.11: The comparison of different τ_{ex} in 2 FPS and 10 FPS scenes.

Ego-motion. In the tracking processing based on the LiDAR data, the ego-vehicle can have a long-distance journey between two LiDAR frames, especially in a low frame rate case. Since all the information in one frame will be encoded to the implicit features, converting it using the recognized pose change is difficult. Furthermore,

the implicit ego-motion parameter included in the detection bounding box will cause difficulty estimating a static vehicle's motion prediction. Therefore, the easiest way to solve this problem is to convert the input LiDAR data into a certain global coordinate system. The comparison of whether removing ego-motion is shown in Tab. 4.12, we can find that both in 2 FPS and 10 FPS, removing the ego-motion can slightly improve the tracking performance.

	Ego	AMOTA↑	AMOTP↓	RECALL↑	FP↓	FN↓	IDS↓
2 FPS	×	0.374	1.285	0.508	7169	28686	3084
	√	0.312	1.412	0.468	8269	289896	3642
10 FPS	×	0.485	1.171	0.515	5129	28269	822
	√	0.430	1.184	0.479	5556	30365	1184

Table 4.12: The comparison of whether having ego-motion in 2 FPS and 10 FPS scenes.

Ablation on STM. The STM module, consisting of temporal and spatial attention, aims to capture the long-term relationship of all objects across the point cloud sequence and help to improve the accuracy of track prediction. In order to further verify the function of STM, we remove this module and evaluate it in 2 FPS and 10 FPS scenes without ego-motion.

We can find that the STM module can play a more effective role in 10 FPS tracking than 2 FPS tracking because the effect of the STM module is to improve the accuracy of track prediction. For 10 FPS tracking, the motion pattern is much easier to learn for the STM module. While for 2 FPS, the motion pattern is harder to distinguish and has more complex spatial and temporal relationships, which STM can help a lot. Therefore, the STM module has a significant impact on 2 FPS tracking.

	STM	AMOTA↑	AMOTP↓	RECALL↑	FP↓	FN↓	IDS↓
2 FPS	×	0.343	1.255	0.510	9832	28600	3163
	√	0.391	1.315	0.512	5156	28432	3309
10 FPS	×	0.474	1.273	0.541	5129	28269	822
	√	0.478	1.206	0.546	5790	26501	1296

Table 4.13: The comparison of whether having STM in 2 FPS and 10 FPS scenes.

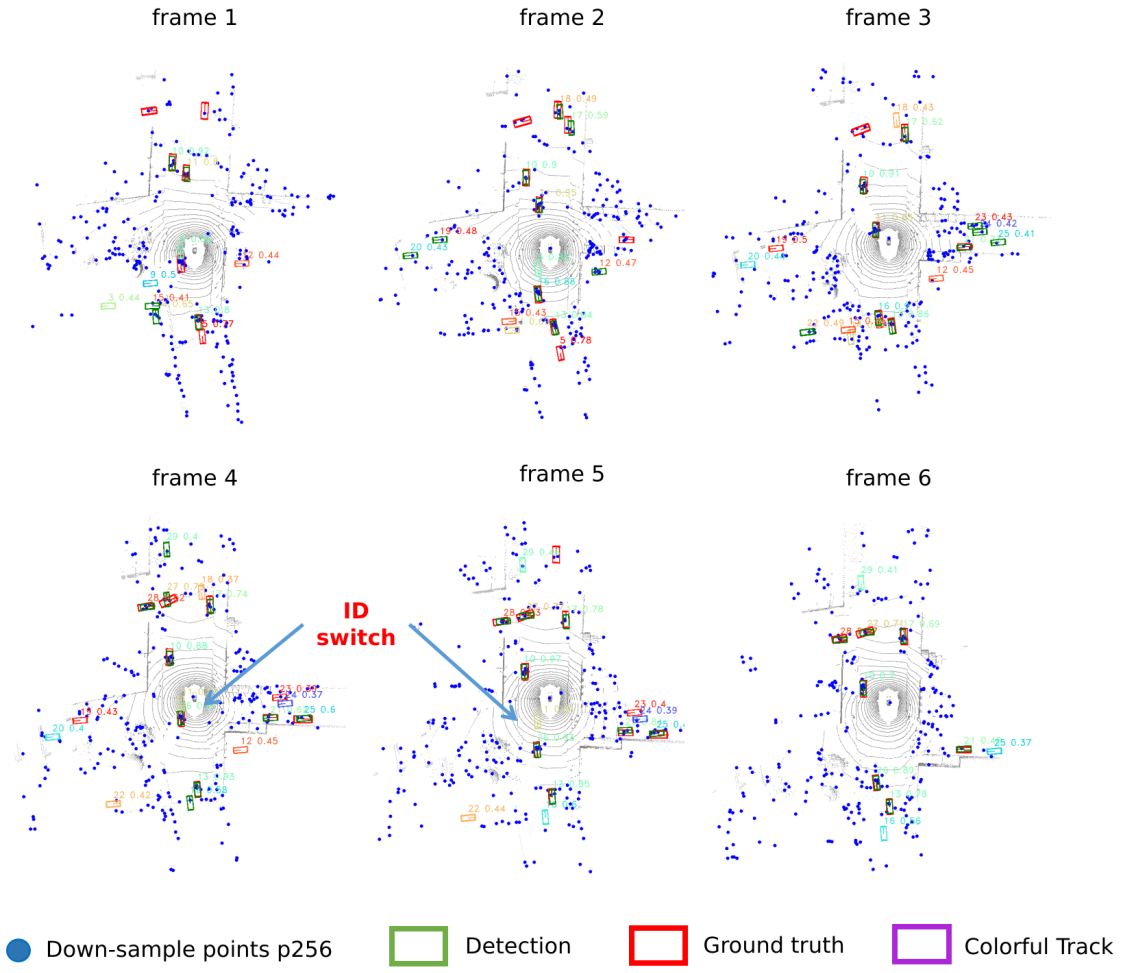


Figure 4.3: Consecutive frames in the 2 FPS tracking inference. *track* 11 at frame 4 and frame 5 causes an ID switch because of the large displacement, and there is no IoU between frames.

Frame rate. The frame rate has a significant impact on the final tracking result. There is large motion displacement over two consecutive frames in the low frame rate tracking. However, most classical tracking algorithms are based on the continuity assumption of the target state, that is, the motion and appearance of the target object are assumed to change little between successive image frames. Therefore, it is easier for the tracker to track the objects in the high frame rate sequence.

In all comparisons above, we can find that the low-frequency tracking brings worse performance, significantly increasing the FP, FN, and ID switch. An inference example of MOTPT in 2 FPS is shown in Fig. 4.3 which illustrates why a low frame rate leads to difficulty tracking. We note that in frame 4 and frame 5, the *track* 11 cannot track for the vehicle and create a new *track* 26 because the vehicles have a large displacement possibly between two adjacent frames, and there is no IoU of the same vehicle in these two frames. Furthermore, the large displacement also increases the risk of causing the mismatch for different tracks, which consequently causes an ID switch. Therefore, the large displacement causes a huge difficulty for our model

in learning the spatial-temporal relations for the objects at a low frame rate.

After comparing the influence of different elements above, we choose the best configuration of our model shown in Tab. 4.14. The evaluation result of the best configuration is shown in Tab. 4.15, and Tab. 4.16 shows the performance comparison with the other 4 3D trackers. The tracking accuracy and precision are not very high, while the false alarms per frame and ID switches are at a high level. Fig. 4.4 shows the main problems of post matching method. The post-matching method relies on the stability of detection and track prediction. When some false positives occur, they may cause the miss matching for the tracked objects. When the track prediction is inaccurate, the new detection may not match the correspondent track prediction. In this case, the new detection will be regarded as new-born objects instead of tracked objects, and it causes an ID switch.

Frame rate	Ego-motion	STM	τ_{match}	τ_{ex}
10 FPS	×	√	5	6

Table 4.14: The best configuration of our model.

Object Class	AMOTA↑	AMOTP↓	RECALL↑	MOTAR↑	GT
car	0.479	1.206	0.546	0.814	58317
MOTA↑	MOTP↓	MT↑	ML↓	FAF↓	TP↑
0.426	0.549	850	1392	97.7	30548
FP↓	FN↓	IDS↓	FRAG↓	TID↓	LGD↓
5692	26480	1289	859	1.45	2.43

Table 4.15: The tracking performance of the best configuration.

Method	AMOTA↑	AMOTP↓	MOTAR↑	MOTA↑	MOTP↓	FP ↓	FN↓	IDS↓
ImmortalTracker [41]	0.68	0.60	0.80	0.57	0.28	18012	21661	320
SimpleTrack [28]	0.67	0.55	0.81	0.57	0.29	17514	23451	575
CenterPoint [50]	0.65	0.53	0.78	0.54	0.29	17355	24557	684
AB3DMOT [44]	0.54	0.83	0.81	0.47	0.30	17886	29190	948
MOPTP(Ours)	0.48	1.21	0.81	0.43	0.55	5692	26480	1289

Table 4.16: Comparisons for 3D MOT on NuScenes dataset for vehicles.

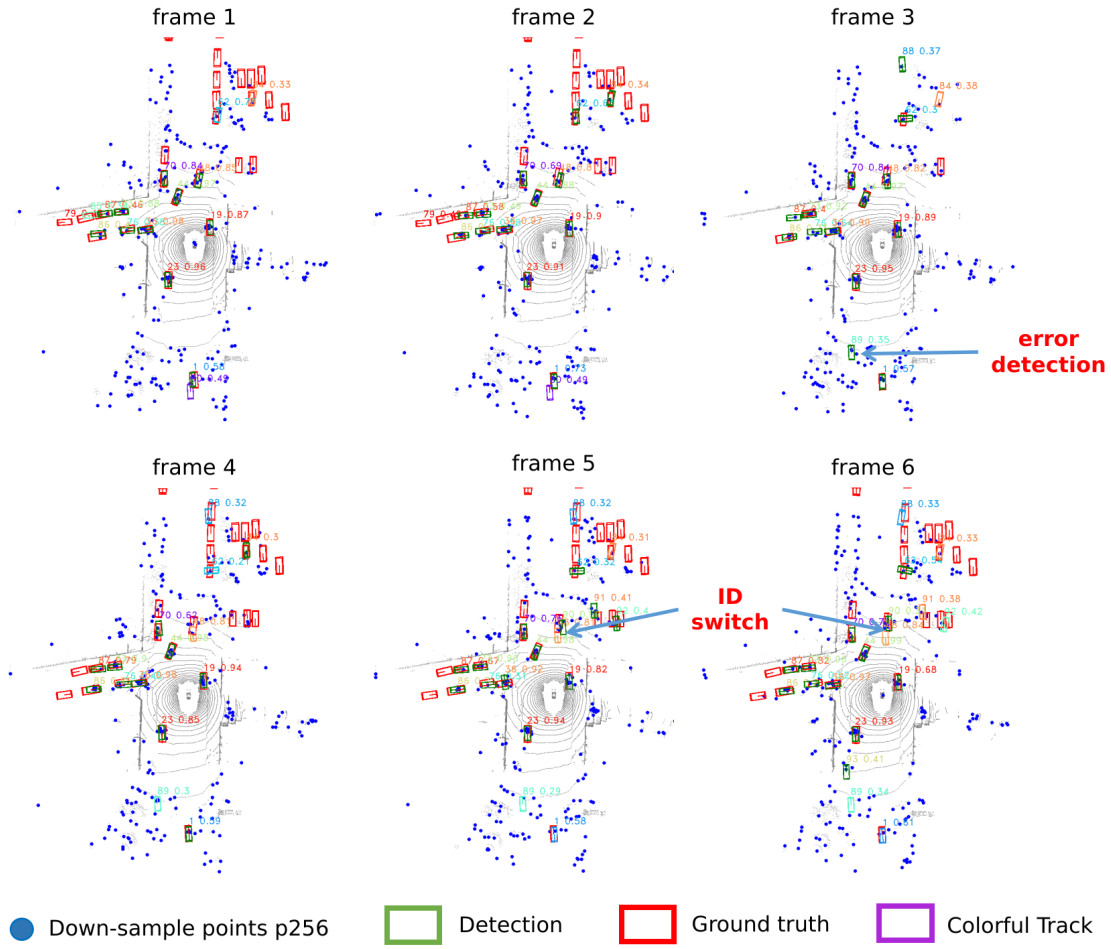


Figure 4.4: An example of inference using post-matching method in 10 FPS scene without ego-motion. The detection is not very stable as some false detections lead to high levels of FP, generating many ID switches.

5

Discussion

5.1 MOTPT and MOTR

MOTPT is a 3D point cloud tracker, which has a huge difference from the image tracker MOTR. The special of point cloud compared with images:

- The point cloud is disordered. Affected by the acquisition device and the coordinate system, the same object is scanned using different devices or positions, and the arrangement order of 3D points is very different. Such data is difficult to process directly through the end-to-end model.
- The point cloud is sparse. In autonomous driving, the sampling point coverage of LiDAR has strong sparseness relative to the scale of the scene. In the KITTI dataset, if the original LiDAR point cloud is projected onto the corresponding color image, only about 3% of the pixels have corresponding radar points. This extreme sparsity makes point cloud-based high-level semantic perception particularly difficult.
- The amount of point cloud information is limited. The data structure of a point cloud is a set of points composed of point coordinates in some three-dimensional space. It is essentially a low-resolution resampling of the geometric shape of the three-dimensional world, so it can only provide one-sided geometric information.

Therefore, the original framework of MOTR can not track the 3D point cloud. In this case, fully end-to-end training is still hard for the 3D tracker. So the original score-filter method in MOTR can not work well for MOTPT in 3D.

5.2 Point query and learnable query

DETR[6] maintains the learnable queries to represent the feature of objects. This method is suitable for 2D image detection because the feature is much rich compared with 3D LiDAR. However, in the 3D point cloud, the feature is not obvious when there are only a few points and the learnable query is slow to converge and sensitive to the initialization. Inspire by 3DETR [27], point query is fast-converge, because down-sample points query input provides a good prior of center and feature. Therefore, it is necessary to use a point query in the MOTPT model.

5.3 Center direct prediction and center offset prediction

The most difficult part of the vehicle prediction is the location prediction because unlike the object prediction in the image, the feature information is dense. **Center direct prediction** predicts the center of bounding boxes c_d from output embedding directly. The advantage is that it updates the center into the Transformer pipeline frame to frame implicitly, which can be expressed as

$$c_d = MLP_{\text{center}}(embed_{out}) \tag{5.1}$$

However, the Transformer DETR-based method prediction is hard to detect the small objects in 2D images. Therefore, center direct prediction can not get a reasonable prediction, because the point cloud feature is so sparse in the 3D point cloud, which is similar to the small objects. **Center offset prediction** predicts the offset from sample points to box centers. It decreases the difficulty of prediction because the main location accuracy lies in the sample points of the backbone, and the offset prediction has a limited and easily converged boundary.

6
Conclusion

In this thesis, we present MOTPT, end-to-end architecture for 3D LiDAR multi-object tracker evaluated on the NuScenes dataset. First, We evaluate a detector as the pre-train model of MOTPT and explore the impact of different parameters. The experiments show that the higher dimension of the point feature causes better detection performance, and it is necessary to use encoder layers in the detector model. However, the performance of the detector model is worse than IA-SSD performance because of the limitation of the one-to-one matching strategy of the Hungarian matcher. Furthermore, for the MOTPT tracker, the score-filter and post-matching methods in the inference mode are compared. The score-filter method is terrible because the model is sensitive to prediction scores and causes many duplicated tracks. While the post-matching method has reasonable performance, its accuracy is still worse than baseline since it is hard to capture the spatial relationship of long-term relationships in the 3D point cloud. We still find that the higher frame rate improves the tracking performance dramatically and removes the ego-motion, also helps to decrease the difficulty of tracking. Meanwhile, the STM module also positively impacts tracking performance at a high frame rate.

Bibliography

[1] Keni Bernardin and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the clear mot metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[2] Alex Bewley et al. "Alextrac: Affinity learning by exploring temporal reinforcement within association chains". In: *2016 IEEE International conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 2212–2218.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[4] Jason Brownlee. "A gentle introduction to the rectified linear unit (ReLU)". In: *Machine learning mastery* 6 (2019).

[5] Holger Caesar et al. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[6] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[7] Kyunghyun Cho et al. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

[8] Kaiwen Duan et al. "Centernet: Keypoint triplets for object detection". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6569–6578.

[9] Peng Gao et al. "Fast convergence of detr with spatially modulated co-attention". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3621–3630.

[10] Zheng Ge et al. "Yolox: Exceeding yolo series in 2021". In: *arXiv preprint arXiv:2107.08430* (2021).

[11] Jun Han and Claudio Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *International workshop on artificial neural networks*. Springer. 1995, pp. 195–201.

[12] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[13] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[14] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[15] Peter J Huber. “Robust estimation of a location parameter”. In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[16] Kyung Hun Kim, Jun Ho Heo, and Suk-Ju Kang. “Towards real-time multi-object tracking in cpu environment”. In: *Journal of Broadcast Engineering* 25.2 (2020), pp. 192–199.

[17] H Kuang Chiu et al. “Probabilistic 3d multi-object tracking for autonomous driving”. In: *ArXiv, vol. abs/2001.05673* (2020).

[18] Harold W Kuhn. “The Hungarian method for the assignment problem”. In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[19] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[20] Shilong Liu et al. “DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=oMI9PjOb9Jl.

[21] Ilya Loshchilov and Frank Hutter. “Fixing Weight Decay Regularization in Adam”. In: *ArXiv* abs/1711.05101 (2017).

[22] Qianhui Luo et al. “3d-ssd: Learning hierarchical features from rgb-d images for amodal 3d object detection”. In: *Neurocomputing* 378 (2020), pp. 364–374.

[23] Wenhan Luo et al. “Multiple object tracking: A literature review”. In: *Artificial Intelligence* 293 (2021), p. 103448.

[24] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. Citeseer. 2013, p. 3.

[25] Nima Mahmoudi, Seyed Mohammad Ahadi, and Mohammad Rahmati. “Multi-target tracking using CNN-based features: CNNMTT”. In: *Multimedia Tools and Applications* 78.6 (2019), pp. 7077–7096.

[26] Tim Meinhardt et al. “Trackformer: Multi-object tracking with transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8844–8854.

[27] Ishan Misra, Rohit Girdhar, and Armand Joulin. “An end-to-end transformer model for 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2906–2917.

[28] Ziqi Pang, Zhichao Li, and Naiyan Wang. “Simpletrack: Understanding and rethinking 3d multi-object tracking”. In: *arXiv preprint arXiv:2111.09621* (2021).

[29] Jinlong Peng et al. “Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking”. In: *European conference on computer vision*. Springer. 2020, pp. 145–161.

[30] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[31] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).

[32] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[33] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[34] Felicia Ruppel et al. “Transformers for Multi-Object Tracking on Point Clouds”. In: *arXiv preprint arXiv:2205.15730* (2022).

[35] Alexey Stakhov and Boris Rozin. “On a new class of hyperbolic functions”. In: *Chaos, Solitons & Fractals* 23.2 (2005), pp. 379–389.

[36] Peize Sun et al. “Transtrack: Multiple object tracking with transformer”. In: *arXiv preprint arXiv:2012.15460* (2020).

[37] Matthew Tancik et al. “Fourier features let networks learn high frequency functions in low dimensional domains”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7537–7547.

[38] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. `https://github.com/open-mmlab/OpenPCDet`. 2020.

[39] Pavel Tokmakov et al. “Learning to track with object permanence”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10860–10869.

[40] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[41] Qitai Wang et al. “Immortal Tracker: Tracklet Never Dies”. In: *arXiv preprint arXiv:2111.13672* (2021).

[42] Yingming Wang et al. “Anchor detr: Query design for transformer-based detector”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 3. 2022, pp. 2567–2575.

[43] Yongxin Wang, Kris Kitani, and Xinshuo Weng. “Joint object detection and multi-object tracking with graph neural networks”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13708–13715.

[44] Xinshuo Weng et al. “3D Multi-Object Tracking: A Baseline and New Evaluation Metrics”. In: *IROS* (2020).

[45] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[46] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[47] Jialian Wu et al. “Track to detect and segment: An online multi-object tracker”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12352–12361.

[48] Zhenbo Xu et al. “Continuous Copy-Paste for One-stage Multi-object Tracking and Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15323–15332.

[49] Min Yang and Yunde Jia. “Temporal dynamic appearance modeling for online multi-person tracking”. In: *Computer Vision and Image Understanding* 153 (2016), pp. 16–28.

[50] 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*. 2021, pp. 11784–11793.

[51] Fangao Zeng et al. “MOTR: End-to-End Multiple-Object Tracking with TRansformer”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[52] Yifan Zhang et al. “Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18953–18962.

[53] Yifu Zhang et al. “Bytetrack: Multi-object tracking by associating every detection box”. In: *arXiv preprint arXiv:2110.06864* (2021).

[54] Yifu Zhang et al. “Fairmot: On the fairness of detection and re-identification in multiple object tracking”. In: *International Journal of Computer Vision* 129.11 (2021), pp. 3069–3087.

[55] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. “Tracking objects as points”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 474–490.

[56] Li Zhu et al. “Bridging the gap between one-to-many and one-to-one label assignment via NMS-aware alignment module”. In: *Neurocomputing* 494 (2022), pp. 346–355.

[57] Xizhou Zhu et al. “Deformable {DETR}: Deformable Transformers for End-to-End Object Detection”. In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=gZ9hCDWe6ke`.

A

Appendix 1

A.1 Pseudo code of MOTR3D

Algorithm 1 Pseudo code of MOTR3D

Input: A point cloud sequence Seq, groud truth GT if training
output: Tracks τ of the sequence

Initialization: track query q_{tr}
for point cloud frame p_t in Seq **do**

$/ * Detect\ single\ frame * /$
$f_{1024}^{128}, p_{1024}, f_{256}^{512}, p_{256} \leftarrow Backbone(p_t)$
Detect queries $q_{det} \leftarrow f_{256}^{512} + PE(p_{256})$
Query embedding $embed_q$ *gets* concatenate(q_{det}, q_{tr})
Output embedding $embed_{out} \leftarrow Transformer$(key=$f_{1024}^{128} PE(p_{1024})$, query=$embed_q$)
Original detect objects $obj_{odet} \leftarrow MLP_{det}(embed_{out}) + p_{256}$
Original track objects $obj_{otr} \leftarrow MLP_{tr}(embed_{out}) + p_{256}$

$/ * STM\ update\ output\ embedding * /$
$embed_{out}^{t} \leftarrow$ temporal attention($embed_{out}^{t}$, $embed_{out}^{t-1}$)
$embed_{out}^{t} \leftarrow$ spatial attention($embed_{out}^{t}$, $embed_{q}^{t}$)

$/ * Track\ single\ frame * /$
if training **then**
Matched tracked indices $\sigma_{tr} \leftarrow$ ID Match(obj_{otr}, GT_{tr})
Matched detection indices $\sigma_{det} \leftarrow$ Hungarian Matcher(obj_{odet}, GT_{det})
Loss $\leftarrow \mathrm{L}_{total}$(obj$_{otr}$, obj_{odet}, σ_{tr}, σ_{det})
else
$/ * postmatchingmethod * /$
Tracked objects $obj_{tr} \leftarrow obj_{odet} \cup obj_{otr}$
New-born objects $obj_{new} \leftarrow obj_{odet} \backslash obj_{otr}$
Disappear objects $obj_{dis} \leftarrow obj_{otr} \backslash obj_{odet}$
Exit objects $obj_{exit} \leftarrow \{t_{dis} \in obj_{dis} \mid t_{dis} \geq \tau_{ex}\}$
end if
end for

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY